Key agreement: security / division

Daniel R. L. Brown[1]

September 8, 2021

Work-in-progress

[1] danibrown@blackberry.com

Abstract

Some key agreement schemes, such as Diffie–Hellman key agreement, reduce to Rabi–Sherman key agreement, in which Alice sends ab to Charlie, Charlie sends bc to Alice, they agree on key $a(bc) = (ab)c$, where multiplicative notation here indicates some specialized associative binary operation.

All non-interactive key agreement schemes, where each peer independently determines a single delivery to the other, reduce to this case, because the ability to agree implies the existence of an associative operation. By extending the associative operation's domain, the key agreement scheme can be enveloped into a mathematical ring, such that all cryptographic values are ring elements, and all key agreement computations are ring multiplications. (A smaller envelope, a semigroup instead of a ring, is also possible.)

Security relies on the difficulty of division: here, meaning an operator / such that $((ab)/b)b = ab$. Security also relies on the difficulty of the less familiar wedge operation $[ab, b, bc] \mapsto abc$.

When Rabi–Sherman key agreement is instantiated as Diffie–Hellman key agreement: its multiplication amounts to modular exponentiation; its division amounts to the discrete logarithm problem; the wedge operation amounts to the computational Diffie–Hellman problem.

Ring theory is well-developed and implies efficient division algorithms in some specific rings, such as matrix rings over fields. Semigroup theory, though less widely-known, also implies efficient division in specific semigroups, such as group-like semigroups.

The rarity of key agreement schemes with well-established security suggests that easy multiplication with difficult division (and wedges) is elusive.

Reduction of key agreement to ring or semigroup multiplication is not a panacea for cryptanalysis. Nonetheless, novel proposals for key agreement perhaps ought to run the gauntlet of a checklist for vulnerability to well-known division strategies that generalize across several forms of multiplication. Ambitiously applying this process of elimination to a plethora of diverse rings or semigroups might also, if only by a fluke, leave standing a few promising schemes, which might then deserve a more focused cryptanalysis.

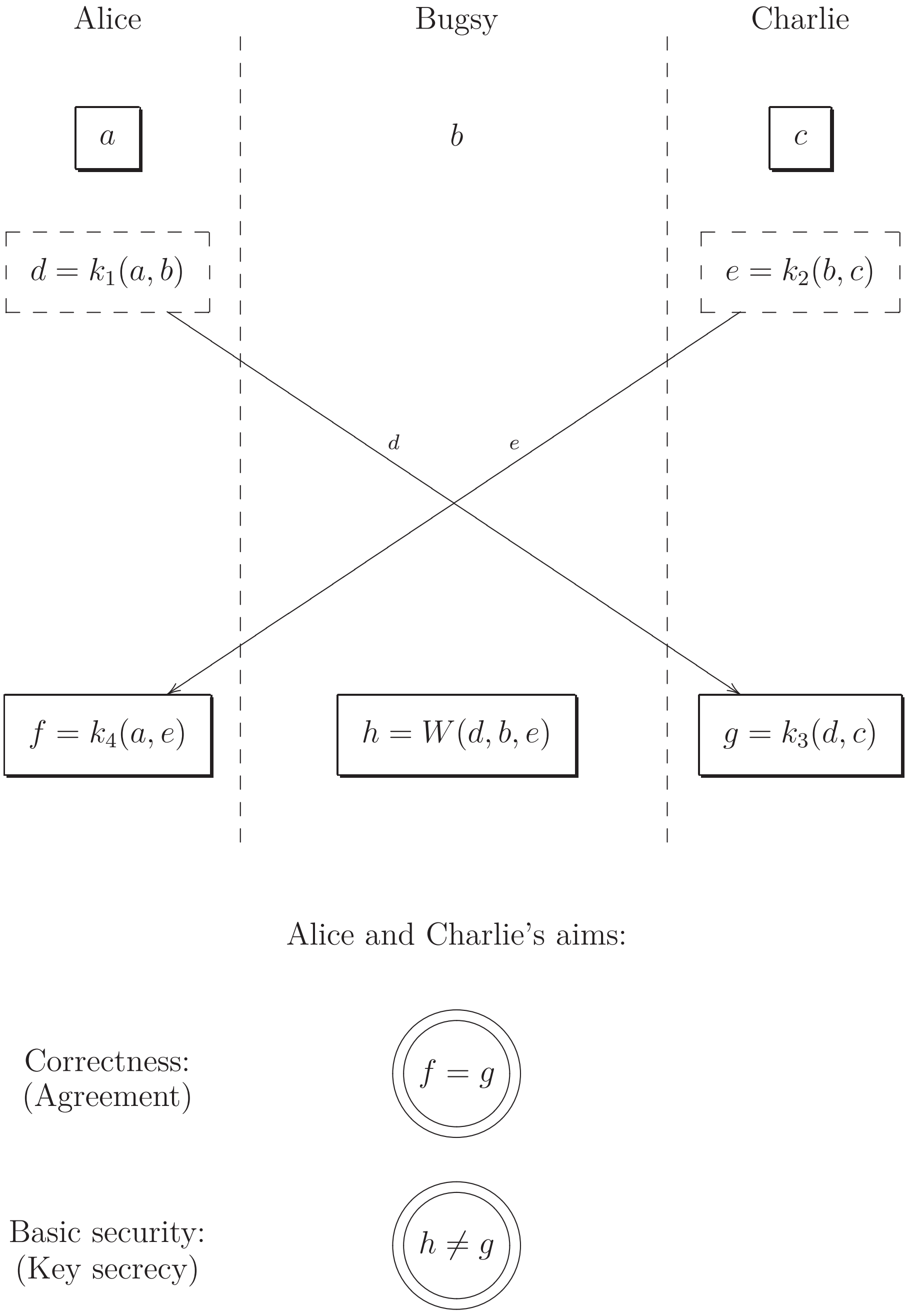

Figure 1: Key agreement between Alice and Charlie, but watched by adversary Bugsy. Informal qualifiers: $\exists[k_1, k_2, k_3, k_4]$ and $\forall[a, c, W]$, for practical functions k_i and feasible function W.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Key agreement is now used often to securely connect devices through the internet. For example, when a browser visits a secure web site, the browser and the web server often use key agreement to agree on a secret key known only to the browser and server.

In more detail, the secure Hypertext Transfer Protocol (HTTPS) uses the Transport Layer Security (TLS) security protocol, whose latest version (1.3) requires Diffie–Hellman key agreement for the initial connection. After the key agreement step, web content between the browser and server is then protected with the agreed key and some kind of symmetric-key authenticated encryption (such as AES-GCM).

Appendix A discusses applications of key agreement more generally.

1.1 Key agreement

In this report, key agreement means the idea sketched in Figure 1[1]. Formal definitions are in Chapter 2. Key agreement is a generalization of basic Diffie–Hellman key agreement, and is usually part of a larger system such as a secure handshake as a step for key distribution in a secure communication protocol.

Key agreement allows two users to agree on a key by delivering each other information. Delivery is non-interactive: each delivery can be independently

[1] Douglas Stebila suggested the crossing of arrows in the figure to distinguish from key encapsulation. I since learned of similar but non-crossing rhombic diagram in an article of De Feo [Feo17].

generated, starting from some initial joint information. If the key agreement is secure, then the agreed key will be a secret known only to the agreeing parties.

Adversaries may see the deliveries, but security depends on the adversaries not modifying the deliveries. In other words, key agreement, as we define it, is unauthenticated. Generally, to avoid an man-in-the-middle attack, some extra security techniques must be applied, to authenticate the deliveries. For example, digital signatures might be applied to the deliveries. These extra mechanisms are not part of key agreement, but a necessary part of larger system.

A few schemes for key agreement are

- Diffie and Hellman's original (1978) modular exponentiation based key agreement,
- Koblitz (1987) and Miller's (1985) elliptic curve variant of Diffie–Hellman key agreement,
- Menezes, Qu and Vanstone's (1995) double-key variant of Diffie–Hellman key agreement, and
- De Feo, Jao and Plut's (2014) super-singular isogeny Diffie–Hellman.

The few schemes above (and yet fewer variations) are widely conjectured to contribute significantly to security, at least when used correctly within a larger protocol. (Furthermore, some kinds of password-authenticated key exchange, such as SPEKE and SPAKE2, also fit this model of key agreement.)

Other than these few schemes and similar ones, secure key agreement seem elusive. The key agreement schemes with well-established security are similar to those above.[2] Obscurer key agreements do not have well-established security (although may well be secure nonetheless).

1.2 The quantum threat

Shor's algorithm and a large-enough quantum computer would break Diffie–Hellman key agreement, including variants ECDH and ECMQV. This quantum computer threat justifies the search for other key agreement schemes.

[2] To be clear, the previous two sentence do not refer interactive key encapsulation schemes.

Super-singular isogeny key exchange, for example, is a key agreement scheme not known to be vulnerable to Shor's algorithm. Other key agreement schemes, such as those based on braid groups, have been proposed too. More schemes might mitigate the quantum threat further.

(Public-key encryption and key encapsulation are alternatives to key agreement. These alternatives are interactive in the sense that one party acts as an initiator and one acts as a responder. In some applications, interactivity is not an obstacle. Some of these, such as RSA-KEM (a key encapsulation method based on factorization of integers), would are also be vulnerable to Shor's quantum computer algorithm. Others, such as code-based McEliece and lattice-based NTRU, are not known to be vulnerable to any quantum attacks.)

1.3 Semigroups and rings

Modern algebra deals often with rings, and sometimes with semigroups. This report uses both rings and semigroups. Although both ring and semigroup are standard notions in algebra, we now review them briefly.

Diversion 1.3.1. Easy examples can be better refreshers than rigorous definitions. The most fundamental ring is the set

$$\mathbb{Z} = \{0, 1, -1, 2, -2, \dots\} = \{\dots, -2, -1, 0, 1, 2, \dots\}$$

of integers under the two usual operations of addition and multiplication.

Semigroups have only a single associative operation, and can be quite a bit quirkier than rings. The set $\{0, 1, 4, 9, 16, \dots\}$ of perfect square integers form a semigroup under multiplication. The set $\{3, 4\} \cup \{n \in \mathbb{Z} : n > 5\}$ forms semigroup under addition.

A **semigroup** is a set S together with an associative binary operation. Most often the binary operation is written as multiplication, and the semigroup said to be **multiplicative**, in which case, the associative axiom is

$$a(bc) = (ab)c,$$

for all $a, b, c \in S$. In an additive semigroup S^+, the associative axiom is $a + (b + c) = (a + b) + c$. A **monoid** is a semigroup with identity element, usually written 1, such $1a = a1 = a$ for all a. (In additive notation, the identity element is written as 0.) A **group** is a monoid, in which each element b has an inverse q such that $bq = qb = 1$. An **abelian** group is a commutative group meaning $ab = ba$ for all a, b.

Some textbooks on semigroup theory are listed in the references of this report. The journal *Semigroup Forum* covers semigroup theory.

For completeness, Appendix B reviews some notation (new and old) for semigroups, and proves a few basic results. This report has a focus on division, which means that our notation emphasizes the division operation more than traditional notation does in ring theory and semigroup theory.

For concreteness, Appendix C provides some examples of semigroup (with the caveat that these should be deemed insecure for use in cryptography).

A **ring** is a set R together with two binary operations, written as addition and multiplication (unless otherwise indicated), which obey a stricter set of axioms:

$$\begin{aligned} a+(b+c) &= (a+b)+c, \\ a(bc) &= (ab)c, \\ a(b+c) &= (ab)+(ac), \\ (a+b)c &= (ac)+(bc), \\ a+b &= b+a \\ R+b &= R, \end{aligned}$$

for all $a, b, c \in R$, and we define $R+b = \{a+b : a \in R\}$. The last condition, implies existence of a binary operation $-$, called **subtraction**, such that $(a-b)+b = a$. Subtraction then implies existence of an additive identity 0, and a negation operation, so that addition forms a group.

In other words, a **ring** is set with two operations: addition forming an abelian group, multiplication forming a semigroup, and multiplication distributing over addition. Many authors also require multiplication to form a monoid, and would label the looser definition above (without identity 1) as a **rng**[3]. In the sub-discipline of commutative algebra, rings are usually assumed to have commutative multiplication. A **field** is a ring in which multiplication forms an abelian group upon exclusion of the additive identity 0.

Ring theory is a mainstream topic of modern algebra, and is taught in many textbooks and courses. For most ring theory, this report relies on the plentiful literature on ring theory. The structure of an extra operation, and the stricter axioms, gives ring theory a much richer character than semigroup theory.

[3] Not to be confused with a random number generator

Nonetheless, rings and semigroups are related, such that some part of their theory can be reduced to each other. One direction is immediate: each ring R forms two semigroups with the same set of elements as R: an additive semigroup R^+ whose operation is addition of R, and a multiplicative semigroup $R^\times$ with multiplication of R as the operation.

In the other direction, every semigroup S is a subsemigroup of the multiplicative semigroup of a ring. For example, consider the **semigroup ring** $R = C[S]$, whose elements are a finite formal linear combinations $\sum_i c_i[s_i]$ where the coefficients c_i belong to a base ring C with an identity 1, and the distinct s_i belong to S. The full set of axioms for a semigroup ring are treated later in this report. The element s of S maps into element $1[s]$ of R, such that S is embedded into $R^\times$.

The semigroup ring is typically much larger than the the semigroup. A semigroup S can have special properties and algorithms that do immediately derive from properties and algorithms holding over the larger semigroup ring $R = C[S]$. In other words, general ring theory might not solve all problems in semigroup theory.

1.4 Associative key agreement

This report studies associative agreement, which uses semigroup multiplication. Rabi and Sherman [RS93] mentioned this idea. Berenstein and Chernyak [BC04] also mention the idea.

In associative key agreement, three elements a, b, c in a ring (or a semigroup) are needed. Alice chooses element a as a secret known only to her, ideally using a cryptographically secure random number generator. Charlie chooses c similarly. The value b is pre-arranged, but is not necessarily secret: b could be fixed for all users, or b could be custom for Alice and Charlie.

Alice computes $d = ab$ and delivers d to Charlie. The value of d might be seen by other parties: it is a major aim of key agreement for Alice and Charlie to establish a shared secret without any pre-existing means of secret communication.

Charlie must somehow be sure that d is actually from Alice. The means Charlie uses to authenticate that d is from Alice is outside the scope of key agreement: Alice and Charlie must use some extra authentication mechanism[4].

[4] For example, Alice might apply a digital signature to d, or perhaps d could be delivered

Charlie computes $e = bc$ and delivers e to Alice, by similar authenticated means[5].

Charlie can compute and deliver e before receiving d from Alice. Alice and Charlie could even send each other deliveries simultaneously.

Alice computes a key $f = ae$ and Charlie computes a key $g = dc$. The keys agree because $f = ae = abc = dc = g$, due to associativity of the semigroup. The main aim is for shared key abc to be a secret known only to Alice and Charlie. For a superficial plausibility of this aim, notice that that Alice and Charlie have used their secrets a and c to compute abc, and having not directly revealed their secrets a and c to anbody else (or each other).

Associative key agreement is defined more formally in Chapter 2.

1.5 All key agreement is essentially associative

Every key agreement scheme is essentially associative, after relabeling the values appropriately, and extra elements, as shown in Chapter 2.

In short, the fact that Alice and Charlie can agree is essentially equivalent to the associative law. Extra elements can used to fill in a full multiplicative semigroup table. For example, most missing operations not defined by Alice and Charlie standard key agreement computations can be defined as zero.

As already noted, a semigroup can also be embedded into a ring. Therefore, the key agreement values can be re-labelled as ring elements.

In particular, Diffie–Hellman key agreement is associative. The associated semigroup is related to the usual Diffie–Hellman group.

1.6 Ring and semigroup security

The primary question is whether an adversary seeing the two deliveries d and e can also compute the key abc. In other words, can abc be computed from d, b and e? A secondary question is whether an adversary can deduce a from d, which could be an additional problem for Alice if she re-uses a, in the key

over an authenticated (but public) channel.

[5] In some applications, only one of the parties of authenticated, perhaps with post-agreement authentication being used for the party whose delivery was not authenticated.

agreement, or otherwise. These questions can be re-stated as computational problems for a semigroup or a ring.

The basic security of key agreement requires that a hard computational problem in the underlying semigroup, the **wedge** problem of computing abc from $[ab, b, bc]$. If the wedge problem is easy, then the associative key agreement scheme is insecure.

The division problem must also be hard. For example, if $a = d/b$, then Alice's secret a can be computed from d and b, neither of which are secret. An adversary who also sees e can compute Alice's key $f = ae$. If Alice uses the secret a for other purposes, then the division problem can be more important than the wedge problem.

In Diffie–Hellman key agreement, the underlying semigroup has division problem equivalent to the discrete logarithm problem, and has a wedge problem equivalent to the well-known Diffie–Hellman problem.

Chapter 2

Key agreement (inter-operation and reduction)

This chapter

- defines a key agreement scheme to be a quadruple of functions that inter-operate correctly, and
- reduces key agreement to semigroups.

Security definitions for key agreement are deferred to Chapter 3.

2.1 Schemes

A scheme for key agreement is specified by four inter-operating functions.

Definition 2.1.1. *A* ***key agreement scheme*** *is an array* $k = [k_1, k_2, k_3, k_4]$ *of functions such that:*

$$k_3(a, k_2(b, c)) = k_4(k_1(a, b), c) \tag{2.1.1}$$

for all a, b, c *such that* $k_1(a, b)$ *and* $k_2(b, c)$ *are defined.*

Diversion 2.1.1. Various alternative terminology has been used. Some variation of **non-interactive, one-round, unauthenticated key exchange** is typical.

Diversion 2.1.2. Figure 1 illustrates this interoperability definition plus some security requirements.

Diversion 2.1.3. Intention of security would perhaps be implied or inferred by the term **key agreement**, even though **insecure key agreement** would be properly understood.

Outside this report, confusion about intent can perhaps be avoided by calling Definition 2.1.1 **operational** key agreement, to help disclaim intention of security by emphasizing the interoperability instead, allowing us to freely discuss key agreement schemes at the level of interoperability, deferring security questions to a later point.

2.2 Diffie–Hellman

The original Diffie–Hellman (using modular exponentiation) is a key agreement scheme:

Lemma 2.2.1. *Let p be a positive integer. Let $k = [k_1, k_2, k_3, k_4]$ where*

$$\begin{aligned} k_1(a,b) &= b^a \bmod p, \\ k_2(b,c) &= b^c \bmod p, \\ k_3(a,e) &= e^a \bmod p, \\ k_4(d,c) &= d^c \bmod p, \end{aligned}$$

with the input variables non-negative integers less than p. Then k is a key agreement scheme.

Proof. Definition 2.1.1 holds because

$$\begin{aligned} k_3(a, k_2(b,c)) &= k_3(a, b^c) \\ &= (b^c)^a \\ &= b^{ca} \\ &= b^{ac} \\ &= (b^a)^c \\ &= k_4(b^a, c) \\ &= k_4(k_1(a,b), c) \end{aligned}$$

where all exponentiation is done modulo p. □

Diversion 2.2.1. Lemma 2.2.1 assumes the standard notational convention that $r = n \bmod p$ means that $n = qp + r$ for $0 \leq r < p$.

Diversion 2.2.2. Writing out all the mod-p reductions in the proof of Lemma 2.2.1, would make clear that a key step of the proof is a mini-lemma that $(z \bmod p)^e \bmod p = z^e \bmod p$ for all integers e, z with $e \geq 0$.

Diversion 2.2.3. Lemma 2.2.1 assumes that $0^0 = 1$, which is the conventional definition, although arguably somewhat arbitrary.

An alternative definition, $0^0 = 0$, is less conventional, and requires an additional proviso to Lemma 2.2.1, which is that p is a radical number (not divisible by the square of any prime). (For example, the proviso is necessary when $p = 12$ and $[a, b, c] = [0, 6, 8]$, because $0 = 0^0 = (6^8)^0 \neq (6^0)^8 = 1^8 = 1$.)

To avoid any confusion, and exceptional corner cases, it will be convenient to work with positive integers only.

Definition 2.2.1. ***Diffie-Hellman key agreement*** *is scheme k defined in Lemma 2.2.1 with all input variables a, b, c, d, e restricted to positive integers, and p restricted to prime numbers. In this case, write the scheme as $k^{DH \bmod p}$.*

Diversion 2.2.4. This definition is properly defined because if $0 < a, b < p$, then $0 < (a^b \bmod p) < p$. This works whenever p is a radical number (not divisible by a perfect square other than 1), so Diffie–Hellman can be defined reasonably for such p.

Diversion 2.2.5. Diffie–Hellman key agreement, for large primes p, is intended, and believed, to meet some of the basic security definitions of Chapter 3.

2.3 Domain and Ranges

It is often useful to study details of k that are implicit in Definition 2.1.1, such as the domains and ranges of the functions.

Diversion 2.3.1. The usual **Cartesian** product of sets is $S_1 \times S_2 = \{[s_1, s_2] : s_i \in S_i\}$, where $[s_1, s_2]$ indicates a two-element **array**. (A two-element array is also known as an (**ordered**) **pair**.)

Diversion 2.3.2. Arrays and Cartesian products can be **indexed** using subscript notation: $[s_1, s_2, \ldots, s_n]_i = s_i$ and $(S_1 \times S_2)_i = S_i$.

Diversion 2.3.3. The domain of a two-input function is really just a set of two-element arrays. In other words, $k_i(x, y) = k_i([x, y])$, with the left side considered an abbreviation.

Each function k_i has form $k_i : \mathrm{Dom}(k_i) \to \mathrm{Ran}(k_i)$, with a defined **domain** $\mathrm{Dom}(k_i)$ and **range** $\mathrm{Ran}(k_i)$. The **image** of k_i is the set $\mathrm{Im}(k_i) = k_i(\mathrm{Dom}(k_i)) = \{k_i([x, y]) : [x, y] \in \mathrm{Dom}(k_i)\}$, which is a subset of its range, so $\mathrm{Im}(k_i) \subseteq \mathrm{Ran}(k_i)$.

Diversion 2.3.4. An alternative definition of function is a **range-less** function, representing a function as a set of pairs $\{[x, f(x)] : x \in \mathrm{Dom}(k_i)\}$. Each function defines a range-less function by ignoring or forgetting the range. Each range-less function defines many functions, by choosing any range that contains the image.

This report works with ranged functions in most definitions. But, in many cases, the main concerns could also work with range-less versions of the same functions.

Range is used occasionally to model special types of security, for example, indistinguishability, where the range indicates the set of values from which the key agreement values are to be indistinguishable. It seems simpler to work with ranged functions, and ignore the ranges when they are irrelevant than to have two sets of working definitions, one ranged, one range-less.

Diversion 2.3.5. A function f is **surjective** if $\mathrm{Im}(f) = \mathrm{Ran}(f)$. Asking if a range-less function f is surjective is meaningless because $\mathrm{Ran}(f)$ is undefined. In many applications of functions, being surjective a significant condition, and in these applications range-less functions are inapplicable.

The fundamental key agreement equation (2.1.1) implies the following relations between the domains and images:

$$\begin{aligned} \mathrm{Dom}(k_1)_2 &= \mathrm{Dom}(k_2)_1 \\ \mathrm{Dom}(k_3) &\supseteq \mathrm{Dom}(k_1)_1 \times \mathrm{Im}(k_2), \\ \mathrm{Dom}(k_4) &\supseteq \mathrm{Im}(k_1) \times \mathrm{Dom}(k_2)_2. \end{aligned}$$

In other words, equation (2.1.1) holds for all $[a, b, c]$ such that $[a, b] \in \mathrm{Dom}(k_1)$ and $[b, c] \in \mathrm{Dom}(k_2)$.

2.4 Aside: rock, scissors, paper

As a clarifying example, the common game **rock, scissors, paper** can be shown to fit the operational interface of key agreement in Definition 2.1.1.

Let $a, c \in \{\mathrm{Rock}, \mathrm{Scissors}, \mathrm{Paper}\}$. Let $k_1(a, b) = a$ and $k_2(b, c) = c$, and $k_3(a, e) = k_4(d, c) \in \{\mathrm{Left}, \mathrm{Tie}, \mathrm{Right}\}$, per Table 2.1. To see that Definition 2.1.1 holds, compute:

$$k_3(a, k_2(b, c)) = k_3(a, c) = k_4(a, c) = k_4(k_1(a, b), c).$$

	Rock	Scissors	Paper
Rock	Tie	Left	Right
Scissors	Right	Tie	Left
Paper	Left	Right	Tie

Table 2.1: Rock, scissors, paper: an example of insecure key agreement

Diversion 2.4.1. Rock, scissors, paper was never intended to be a secure key agreement scheme. It is not secure under Chapter 3, because of many very feasible attacks.

2.5 Computation and communication

Alice and Charlie will use a key agreement scheme k, with the following computations and communications.

1. Alice selects a, Charlie selects c, and they both select the same b.
2. Alice computes $d = k_1(a, b)$ and communicates d to Charlie; Charlie computes and communicates $e = k_2(b, c)$ to Alice.
3. Alice computes key $f = k_3(a, e)$; Charlie computes key $g = k_4(d, c)$.

Definition 2.1.1, ensures that Alice and Charlie compute the same key, $f = g$, because

$$f = k_3(a, e) = \underbrace{k_3(a, k_2(b, c)) = k_4(k_1(a, b), c)}_{\text{Definition 2.1.1}} = k_4(d, c) = g.$$

In other words, the keys of Alice and Charlie agree, and Alice and Charlie have achieved key agreement.

Key agreement is usually just part of a larger protocol, as discussed in Appendix A.

2.6 Practical schemes

A **practical** key agreement scheme is any scheme $k = [k_1, k_2, k_3, k_4]$ where each function k_i can be implemented by a known, practical algorithm.

Diversion 2.6.1. A **practical** algorithm is one that always returns a (correct) answer with a cost convenient to the user (in terms of time, energy, and so on).

The **convenience** of user cost can be subjective. It can be relative to the benefit, in which case it depends on the security. It can relative to comparable key agreement schemes. It can be relative to other costs (such as application cost, such as user interfaces and symmetric-key encryption).

2.7 Sessions

It will be convenient to gather all the values that Alice and Charlie compute into an array.

Definition 2.7.1. *A* ***session*** *of key agreement scheme k is an array*

$$[a, b, c, d, e, f]$$

such that

$$\begin{aligned} d &= k_1(a, b), \\ e &= k_2(b, c), \\ f &= k_3(a, e). \end{aligned}$$

Diversion 2.7.1. A key agreement scheme k is almost determined by its set of sessions. If two schemes k and K have the same set of sessions, they are **sessionally identical**.

Sessionally identical schemes can differ, for example, in the ranges of the functions k_i.

2.8 Subschemes

Definition 2.8.1. *A key agreement scheme k is a* ***subscheme*** *of key agreement scheme K if every session $[a, b, c, d, e, f]$ of k is a session of K.*

The prototypical example is a subscheme of Diffie–Hellman key agreement in which b is fixed to be an element of multiplicative order q modulo p (where q is prime). Also, a and c are often taken to be integers strictly between 1 and q.

Diversion 2.8.1. More generally, k is subscheme of K if each function k_i is a domain-restriction of the function K_i (meaning $\mathrm{Dom}(k_i) \subseteq \mathrm{Dom}(K_i)$ and $k_i(x) = K_i(x)$ for all $x \in \mathrm{Dom}(k_i)$).

Diversion 2.8.2. In some applications of key agreement, the ranges $\mathrm{Ran}(k_i)$ are important to track. A **full-ranged** subscheme k of K is a subscheme k of K such that $\mathrm{Ran}(k_i) = \mathrm{Ran}(K_i)$.

2.9 Probabilistic schemes

Key agreement defined above is deterministic, but in many applications it will be run probabilistically, as defined formally below.

Definition 2.9.1. *A **probabilistic** key agreement scheme $[a, b, c, k]$ consists of three random variables a, b, c and a key agreement scheme k, such that random variables $k_1(a, b)$ and $k_2(b, c)$ are always defined.*

To use $[a, b, c, k]$, means that Alice samples random variable a, Charlie samples c, and they together sample b. In many cases, random variable b will a constant random variable (all random samples taking the same constant value).

Diversion 2.9.1. Generally, the conditional random variables $a|_b$ and $c|_b$ should be independent, so that random variable b contains the only mutual information between Alice and Charlie.

Diversion 2.9.2. Probabilistic key agreement will help model practical security in Chapter 3 by quantifying an attacker's ability to simply guess the values a or c chosen by Alice and Charlie.

One way to see why probability helps is consider efficiency. For the functions k_i to be efficiently implementable (bounded run-time for any input), their images must be finite sets. But finite images of k_1 and k_2 enable guessing adversaries with nonzero success rate. To resist guessing adversaries, Shannon's theory has made clear:

- security uses secrecy;
- secrecy is hidden information;
- information is quantified by entropy;
- entropy is defined in terms of probability.

So, resisting these real-world attackers means Alice and Charlie will need to deploy some kind of random variables in combination with deterministic functions.

Random variables affect interoperability: sampling random variables takes time, and different samples as inputs to the algorithms implementing the function k_i have different run-times. Run-time affects interoperability and usability, so random variables are also include in this chapter about interoperability.

Definition 2.9.2. *A probabilistic key agreement scheme is **uniform** if each triple $[a, b, c]$ (valid in the sense $k_1(a, b)$ and $k_2(b, c)$ are defined), is equally likely.*

Diversion 2.9.3. If k_1 and k_2 have finite domains, then there exists exactly one uniform $[a, b, c, k]$.

Diversion 2.9.4. Determining probabilities generally requires assumptions, guesswork, or statistical inference. So, though random variables have great theoretical value, they leave open some questions of rigor.

The deterministic definition of key agreement avoids these questions of rigor, yet still permits important questions to be investigated.

Definition 2.9.3. *The* ***support*** *of the probabilistic key agreement scheme $[a, b, c, k]$ is the subscheme k' of k whose sessions $[a, b, c, d, e, f]$ consists of those with nonzero probability under $[a, b, c, k]$.*

Diversion 2.9.5. If $[a, b, c, k]$ and $[A, B, C, K]$ are probabilistic key agreement schemes, then $[a, b, c, k]$ is subscheme of $[A, B, C, K]$, if k is a subscheme K and a is a restriction[1] of A, b a restriction of B, and c a restriction of C.

Generally, a probabilistic key agreement will only be practical if it is discrete, as defined below.

Definition 2.9.4. *A probabilistic key agreement scheme is* ***discrete*** *if $[a, b, c]$ is a discrete random variable*[2]*.*

Diversion 2.9.6. Non-discrete probabilistic key agreement implies that Alice and Charlie would need to communicate and compute with an uncountable set of values, which are rather impractical.

2.10 Aside: faulty schemes

Faulty key agreement is not studied in this report, but is defined in this section, for completeness and clarification. Faulty key agreement relaxes the condition on k in a probabilistic key agreement scheme.

Definition 2.10.1. *A* ***procedural key agreement*** *scheme is an array $[a, b, c, k]$ where a, b, c are random variables, and $k = [k_1, k_2, k_3, k_4]$ is array of functions such that k_i such that both*

$$k_3(a, k_2(b, c)), \qquad k_4(k_1(a, b), c)$$

are defined with probability 1.

[1] A random variable x is a restriction of random variable X if a condition like the following holds: (i) there exists a set R such that $\Pr[X \in R] \neq 1$ and (ii) for all $S \subseteq R$ such that $\Pr[X \in S]$ is defined, then $\Pr[x \in S] = \frac{\Pr[X \in S]}{\Pr[X \in R]}$.

[2] Recall that a discrete random variable v takes countably many values $v_1, v_2, \ldots$ with nonzero probabilities p_i such that $\sum_i p_i = 1$.

By contrast, a continuous random variable assigns probabilities to events, subsets of a universe, generally using a measurable space as the universe, taking an event to be any measurable subset of the space, and letting the probability of an event be the integral of a probability density function computed over the measurable subset. A continuous random variable has probability zero of taking any particular value.

Any probabilistic key agreement is a procedural key agreement, but some procedural key agreements are not probabilistic key agreement schemes.

Definition 2.10.2. *A* ***faulty key agreement*** *scheme is a procedural key agreement scheme which is not a probabilistic key agreement scheme.*

Diversion 2.10.1. Generally, what makes $[a, b, c, k]$ a faulty key agreement scheme is that k is not a key agreement scheme.

The similarity in usefulness of faulty key agreement to that of probabilistic key agreement is measured by the following probability.

Definition 2.10.3. *The* ***success rate*** σ *of a faulty key agreement scheme is the probability that* $k_3(a, k_2(b, c)) = k_4(k_1(a, b), c)$. *The* ***failure rate*** *is* $1 - \sigma$.

Diversion 2.10.2. The term **faulty** is arguably too harsh if the failure rate is negligible (or even zero), because, in this case, faulty key agreement might be quite reasonably secure.

So, the term **faulty** serves mainly to delimit of this report's scope to a focus on associative key agreement and a reduction to semigroup theory.

Diversion 2.10.3. Faulty key agreement would typically be constructed by combination of **approximate** key agreement and **error correction**.

In approximate key agreement $k' = [k'_1, k'_2, k'_3, k'_4]$, the key agreement equation is replaced by a bound on a function $\Delta : \mathrm{Ran}(k_3) \times \mathrm{Ran}(k_4) \to T$, where T an ordered set.

$$\Delta(k'_3(a, k'_2(b, c)), k'_4(k'_1(a, b), c)) \leq \delta.$$

Typically, Δ would be distance metric, where $\mathrm{Ran}(k_3) = \mathrm{Ran}(k_4)$ and T is the set of non-negative real numbers.

A faulty key agreement scheme $k = [k_1, k_2, k_3, k_4]$ is then obtained by applying an error correction function $R : T \to U$, as

$$\begin{aligned} k_3(a, e) &= R(k'_3(a, e)), \\ k_4(d, c) &= R(k'_4(d, c)). \end{aligned}$$

A typical error correction function R with continuous domain involves rounding of real-valued variables. Error correction functions R with discrete or finite domains can be more sophisticated than simple rounding.

Diversion 2.10.4. If a faulty key agreement scheme $[a, b, c, k]$ has a success rate one and a, b, c are discrete random variables, then it ought to be possible to extract a probabilistic key agreement scheme $[a', b', c', k']$ as subscheme of $[a, b, c, k]$ (with same general meaning as subschemes of probabilistic schemes).

Diversion 2.10.5. Allowing faulty key agreement may expand the scope of possibly useful schemes, but also may take extra work to handle nonzero failure rate in various contexts (the worst case being when an attacker can induce the failure case with probability higher than the failure rate).

Furthermore, the reduction of key agreement to semigroups, might become a (faulty?) reduction from faulty key agreement to faulty semigroups (partially associative magmas), a much larger category of mathematical object, which is less well-studied (and less well-behaved). Besides, the scope obtained from all semigroups is already very large, at least from the perspective of providing many targets for cryptanalysis.

For such reasons, faulty key agreement is not considered further in this report.

2.11 Multiplicative schemes

The focus of this report is the following type of key agreement (see [RS93, BC04]).

Lemma 2.11.1. *Let S be a multiplicative semigroup*[3]*. Let $k^S = [k_1, k_2, k_3, k_4]$ where*

$$k_1(a, b) = ab, \tag{2.11.1}$$
$$k_2(b, c) = bc, \tag{2.11.2}$$
$$k_3(a, e) = ae, \tag{2.11.3}$$
$$k_4(d, c) = dc, \tag{2.11.4}$$

for all $a, b, c, d, e \in S$. Then k^S is a key agreement scheme.

Proof. Check the key agreement equation:

$$\begin{aligned} k_3(a, k_2(b, c)) &= k_3(a, bc) \\ &= a(bc) \\ &= (ab)c \\ &= k_4(ab, c) \\ &= k_4(k_1(a, b), c), \end{aligned}$$

where the outer equations are from the definition, the middle equation is due to the associativity law. □

[3] A semigroup is (recall) a set S with an associative binary operation. A multiplicative semigroup uses multiplicative notation for the binary operation. In a multiplicative semigroup, the associative law is written $a(bc) = (ab)c$ for all $a, b, c \in S$ in a multiplicative semigroup. For more detail on semigroups, see Chapter B.

Definition 2.11.1. *A* ***multiplicative*** *key agreement scheme is a scheme* $k^S = [k_1, k_2, k_3, k_4]$ *from Lemma 2.11.1.*

The scheme k^S *is* ***associated*** *with semigroup* S*, and is the unique multiplicative scheme associated with* S*.*

A semigroup that uses a non-multiplicative notation is isomorphic to a semigroup with multiplicative notation, so can also be associated with a key agreement scheme.

A converse to Lemma 2.11.1 can be formed, using the next notion.

Definition 2.11.2. *A key agreement scheme* $[k_1, k_2, k_3, k_4]$ *is* ***symbiotic***

- $k_1 = k_2 = k_3 = k_4$*, and*
- $\mathrm{Dom}(k_1) = \mathrm{Ran}(k_1) \times \mathrm{Ran}(k_1)$.

Diversion 2.11.1. Saying k_i are equal means $\mathrm{Ran}(k_i)$ are equal, and $\mathrm{Dom}(k_i)$ are all equal, and of course $k_i(a, b) = k_1(a, b)$ for all i and all $[a, b] \in \mathrm{Dom}(k_1)$.

Diversion 2.11.2. The mnemonic for **symbiotic** is **same binary operation**.

A multiplicative key agreement scheme is symbiotic, because $k_i(a, b) = ab$, and $\mathrm{Dom}(k_i) = S \times S$ and $\mathrm{Ran}(k_i) = S$, for a semigroup S. Conversely, the following holds.

Lemma 2.11.2. *Let* k *be a symbiotic key agreement scheme. Let* S *be the set* $\mathrm{Ran}(k_1)$ *with a multiplication operation defined by*

$$ab = k_1(a, b).$$

Then S *is a semigroup, and* k *is the multiplicative key agreement associated with* S*.*

Proof. To prove the associative law in S, compute as follows

$$\begin{aligned} a(bc) &= k_1(a, bc) \\ &= k_1(a, k_1(b, c)) \\ &= k_3(a, k_2(b, c)) \\ &= k_4(k_1(a, b), c) \\ &= k_1(k_1(a, b), c) \\ &= k_1(ab, c) \\ &= (ab)c, \end{aligned}$$

which shows that $a(bc) = (ab)c$ for all $a, b, c \in S$.

To show that k is the multiplicative scheme associated with S, check that

$$\begin{aligned} k_1(a,b) &= k_1(a,b) = ab, \\ k_2(b,c) &= k_1(b,c) = bc, \\ k_3(a,e) &= k_1(a,e) = ae, \\ k_4(d,c) &= k_1(d,c) = dc, \end{aligned}$$

using equality of the k_i and definition of multiplication as k_1. □

Multiplicative (and symbiotic) key agreement might seem rather special, but later it will be shown that it is actually quite general.

2.12 Aside: reflective and chiral schemes

Definition 2.12.1. *A key agreement scheme $k = [k_1, k_2, k_3, k_4]$ is* ***reflective*** *if*

$$\begin{aligned} k_1(a,b) &= k_2(b,a) \\ k_3(a,b) &= k_4(b,a) \end{aligned}$$

for all a, b in the appropriate domains. Otherwise, k is ***chiral****.*

Diffie–Hellman key agreement (Definition 2.2.1) is reflective. If S is a commutative semigroup, then the multiplicative key agreement scheme k^S (Definition 2.11.1) is reflective, otherwise k^S is chiral.

Diversion 2.12.1. In reflective key agreement, the roles of Alice and Charlie are interchangeable.

In chiral key agreement, Pat must use Charlie's functions k_2 and k_4 to talk Alice. Similarly, Pat must use Alice's functions k_1 and k_3 to talk to Charlie. The roles of Alice and Charlie can be renamed **left** and **right**, when more than two parties are involved. Chiral key agreement does not permit agreement between two parties playing the same role, such as left and left.

2.13 Associative schemes

Definition 2.13.1. *An* ***associative*** *key agreement k is a subscheme of a multiplicative key agreement scheme K.*

If K is associated with semigroup S, then say that k is ***associated*** *with S.*

Every session of an associative key agreement scheme takes the form:

$$[a, b, c, d, e, f] = [a, b, c, ab, bc, abc].$$

In particular, choose three subsets $A, B, C \subseteq S$ of a multiplicative semigroup S. Let $k^{A,B,C,S}$ be an associative key agreement scheme that is a subscheme of the multiplicative scheme k^S with all sessions $[a, b, c, d, e, f]$ having $[a, b, c] \in A \times B \times C$. More precisely, $k^{A,B,C,S} = [k_1, k_2, k_3, k_4]$ where

$$\begin{aligned} k_1 &: A \times B \to S : [a, b] \mapsto ab, \\ k_2 &: B \times C \to S : [b, c] \mapsto bc, \\ k_3 &: A \times S \to S : [a, e] \mapsto ae, \\ k_4 &: S \times C \to S : [d, c] \mapsto dc. \end{aligned}$$

Call $k^{A,B,C,S}$ an **induced** associative key agreement scheme (induced by $[A, B, C]$).

2.14 Aside: non-associative scheme

A **non-associative** key agreement scheme is any scheme k that is not associative.

In particular, Diffie–Hellman key agreement k (Definition 2.2.1) is not associative. To see this, first $k_1(a, b) = k_3(a, b) = b^a \neq a^b = k_2(b, a) = k_4(b, a)$, for most a and b. But if k were associative, then it would be the subscheme of a multiplicative scheme K, and would have $k_1(a, b) = K_1(a, b) = K_2(a, b) = k_2(a, b)$, a contradiction.

For a smaller non-associative scheme, consider rock, scissors, paper key agreement again. Represent it as $k_1(a, b) = a$ and $k_2(b, c) = c$ and $k_3(a, b) = k_4(a, b) = (a - b) \bmod 3$, where all $a, b, c \in \mathbb{Z}/3 = \{0, 1, 2\}$, making each k_i a binary operation on $\mathbb{Z}/3$. Recall that this is a key agreement scheme because $k_3(a, k_2(b, c)) = k_3(a, c) = (a - c) \bmod 3 = k_4(a, c) = k_4(k_1(a, b), c)$. Suppose that k is a subscheme of a multiplicative scheme K. Then all the sessions of k must be sessions of K, so $k_1(a, b) = K_1(a, b)$ and $k_2(b, c) = K_2(b, c)$. But since K is multiplicative, K is symbiotic with $K_1 = K_2$. Clearly, $k_1 \neq k_2$, so k cannot be a subscheme of K.

2.15 Equivalent schemes

Key agreement schemes k and K are **isomorphic** if they differ only by a relabeling of the values Alice and Charlie use. In more detail, a key agreement scheme k is isomorphic to a key agreement scheme K if there is an array $[\alpha, \beta, \gamma, \delta, \eta, \phi]$ of bijections, such that,

- If $[a, b, c, d, e, f]$ is a session k, then $[\alpha(a), \beta(b), \gamma(c), \delta(d), \eta(e), \phi(f)]$ is a session of K.

- Every session of K has the form $[\alpha(a), \beta(b), \gamma(c), \delta(d), \eta(e), \phi(f)]$ for a session $[a, b, c, d, e, f]$ of k.

Two schemes k and K are **equivalent** if they are isomorphic and the relabeling maps are efficient functions in both directions (so, $a \mapsto \alpha(a)$ and $\alpha(a) \mapsto a$, and similar).

Diversion 2.15.1. A slightly stricter type of isomorphism is range-isomorphism. In this case, the bijections cover the entire domains and ranges of the functions k_i and K_i. To be completed.

2.16 Aside: derived schemes

Given a key agreement scheme k, another scheme can be formed by applying appropriate functions to the inputs and outputs of the functions k_i.

Lemma 2.16.1. *Let $k = [k_1, k_2, k_3, k_4]$ be a key agreement scheme. Let $\alpha, \beta, \gamma, \delta, \delta', \eta, \eta', \phi$ be functions, such that*

$$\begin{aligned} K_1(a, b) &= \delta(k_1(\alpha(a), \beta(b))) \\ K_2(b, c) &= \eta(k_2(\beta(b), \gamma(c))) \\ K_3(a, e) &= \phi(k_3(\alpha(a), \eta'(e))) \\ K_4(d, c) &= \phi(k_4(\delta'(d), \gamma(c))). \end{aligned}$$

are well-defined. If $\delta'(\delta(x)) = x$ and $\eta'(\eta(x)) = x$ for all x in the appropriate domains, then $K = [K_1, K_2, K_3, K_4]$ is a key agreement scheme.

Proof. The key agreement equation for K can be checked as follows:

$$\begin{aligned}
K_3(a, K_2(b,c)) &= \phi(k_3(\alpha(a), \eta'(K_2(b,c))))) \\
&= \phi(k_3(\alpha(a), \eta'(\eta(k_2(\beta(b), \gamma(c)))))) \\
&= \phi(k_3(\alpha(a), k_2(\beta(b), \gamma(c)))) \\
&= \phi(k_4(k_1(\alpha(a), \beta(b)), \gamma(c))) \\
&= \phi(k_4(\delta'(\delta(k_1(\alpha(a), \beta(b)))), \gamma(c))) \\
&= \phi(k_4(\delta'(K_1(a,b)), \gamma(c))) \\
&= K_4(K_1(a), c).
\end{aligned}$$

In other words, when Alice selects a in K, Alice is effectively selecting $\alpha(a)$ in k. For Charlie, using c in K amounts to using $\gamma(c)$ in k. When Alice delivers d in K, Alice effectively delivering $\delta'(d)$ in k. And so on. □

Diversion 2.16.1. The scheme K will be well-defined if each of

$$\begin{aligned}
\operatorname{Im}(\alpha) \times \operatorname{Im}(\beta) &\subseteq \operatorname{Dom}(k_1), \\
\operatorname{Im}(\beta) \times \operatorname{Im}(\gamma) &\subseteq \operatorname{Dom}(k_2), \\
\operatorname{Dom}(k_1) \times \operatorname{Im}(\eta) &\subseteq \operatorname{Dom}(k_3), \\
\operatorname{Im}(\eta) \times \operatorname{Dom}(k_1) &\subseteq \operatorname{Dom}(k_4),
\end{aligned}$$

holds, for example.

Write $\mu = [\alpha, \beta, \gamma, \delta, \eta, \phi]$ and $K = \mu(k)$, and say that K is derived from k.

Diversion 2.16.2. Cryptographers might appreciate examples of derived schemes with:

- α and γ some kind of pseudorandom domain expansion, such as a password-based key derivation function, or a pseudorandom random number generator mapping a raw entropy source into a uniformly distributed value, or domain-adjustment operation mapping one uniform distribution to another nearly uniform distribution over a different domain;
- δ and η some kind of encoding or formatting functions, such as leading-zero-bit-padding, radix expansion (8-bit, 32-bit, ...), endianness-switching, or syntax formatting (ASN.1, JSON, XML);
- ϕ some kind of key derivation function, such as a hash function to remove bias as bit string, or a labeled one-way function to produce multiple effectively independent sub-keys, or even a stream cipher to produce a key stream to encrypt a message.

Intuition might suggest that less derived schemes might be easier to break, so might be the proper target of cryptanalysis.

The maps $\mu = [\alpha, \beta, \gamma, \delta, \eta, \phi]$ define morphisms between key agreement schemes, forming a mathematical category of key agreement. Maps can be composed, map composition is associative, and the identity map 1_k from k to itself is the array of identity functions on appropriate domains.

Two schemes k and K are isomorphic if there are maps μ and ν such that

$$\begin{aligned} k &= \mu(k) \\ K &= \nu(K) \\ \mu \circ \nu &= 1_K \\ \nu \circ \mu &= 1_k \end{aligned}$$

In other words, isomorphic schemes k and K differ only by bijective coordinate re-labelings of the sessions $[a, b, c, d, e, f]$.

A subscheme k of K is derived from K by making each re-labeling function an embedding function, so that $\alpha(a) = a$ for all $a \in \mathrm{Dom}(k_1)_1$, and so on.

2.17 Essentially associative schemes

Equivalence extends associativity to the following larger class of key agreement schemes.

Definition 2.17.1. *A key agreement scheme is **essentially associative** if it is equivalent to an associative scheme.*

2.18 Diffie–Hellman is essentially associative

Lemma 2.18.1. *The Diffie–Hellman key agreement scheme $k^{DH \bmod p}$ from Definition 2.2.1 is essentially associative.*

Proof. Form a semigroup on set

$$S = \{[1], \ldots, [p-1], [p+1], [p+2], \ldots [2p-1]\} \tag{2.18.1}$$

with multiplication defined as

$$[a][b] = \begin{cases} [(ab - 1 \bmod (p-1)) + 1] & \text{if } a < p, \quad b < p, \\ [(a^b \bmod p) + p] & \text{if } a > p, \quad b < p, \\ [(b^a \bmod p) + p] & \text{if } a < p, \quad b > p, \\ [p+1] & \text{if } a > p, \quad b > p. \end{cases} \tag{2.18.2}$$

In other words, for $a, b \in \{[1], \dots, [p-1]\}$, use multiplication modulo $p-1$, except that the result is put into the range $[1, p-1]$ (instead of the usual range $[0, p-2]$ computed using remainders). If exactly one of a or b is in $\{[p+1], \dots, [2p-1]\}$, then use exponentiation modulo p, except the result is put into the range $[p+1, 2p-1]$ (instead of the usual $[0, p-2]$ range computed using remainders). If both a and b are in $\{[p+1], \dots, [2p-1]\}$, then default to returning $[p+1]$.

Associativity of S can be proved by looking at how many of the relevant variables exceed p.

If none of the variables a, b, c exceed p, so $a, b, c < p$, then

$$[a]([b][c]) = [((abc) - 1 \bmod p-1)) + 1] = ([a][b])[c]. \tag{2.18.3}$$

To see this, consider that the inner parenthesized product on each side, has a result less than p, so all computations are essentially multiplications modulo $p-1$.

If one of the variables a, b, c exceeds p, say, one of a, b, c has value e with $e > p$, and the others have values $x, y < p$, then

$$[a]([b][c]) = [e^{xy} \bmod p] = ([a][b])[c]. \tag{2.18.4}$$

To see this, consider that the inner parenthesized product on each side is either a power computed modulo p, or a product modulo $p-1$, and then the outer product on each side is a power computed modulo p.

If two or more of the variables a, b, c exceed p, then

$$[a]([b][c]) = [p+1] = ([a][b])[c]. \tag{2.18.5}$$

To see this, consider that the inner parenthesized product on each side has a result greater than p, and is actually $p+1$, if the the unused variable is less than p. The outer product is then of two values exceeding p, which results in $p+1$, or involves $p+1$ and a number smaller than p, resulting an exponentiation, but that exponentiation leads back to $p+1$ because the base of exponentiation is $p+1$.

Let $A = C = \{[1], \dots, [p-1]\}$ and $B = \{[p+1], \dots, [2p-1]\}$. Form the induced associative key agreement scheme $K = k^{A,B,C,S}$. (See §2.13.)

Then K is equivalent to the given Diffie–Hellman key agreement $k = k^{\mathrm{DH} \bmod p}$, with session $[a, b, c, d, e, f]$ of k mapping to the session

$$[[a], [b+p], [c], [d+p], [e+p], [f+p]]$$

of K. □

Table 2.2 lists the multiplication table of $p = 5$ version of the semigroup used in the proof, which is a small but typical example.

	[1]	[2]	[3]	[4]	[6]	[7]	[8]	[9]
[1]	[1]	[2]	[3]	[4]	[6]	[7]	[8]	[9]
[2]	[2]	[4]	[2]	[4]	[6]	[9]	[9]	[6]
[3]	[3]	[2]	[1]	[4]	[6]	[8]	[7]	[9]
[4]	[4]	[4]	[4]	[4]	[6]	[6]	[6]	[6]
[6]	[6]	[6]	[6]	[6]	[6]	[6]	[6]	[6]
[7]	[7]	[9]	[8]	[6]	[6]	[6]	[6]	[6]
[8]	[8]	[9]	[7]	[6]	[6]	[6]	[6]	[6]
[9]	[9]	[6]	[9]	[6]	[6]	[6]	[6]	[6]

Table 2.2: Multiplication table for a Diffie–Hellman semigroup, with $p = 5$

Diversion 2.18.1. Lemma 2.2.1 shows that a similar key agreement scheme allowing input variables with value 0. Adding an element $[p]$ to S in the proof can be done under a similar multiplication rule, lumping $[p]$ into the cases with $a > p$, but does not give an associative multiplication. (For example, $([5][2])[2] = [5][2] = [5]$ and $[5]([2][2]) = [5][4] = [6]$.)

Finding an equivalent associative key agreement is still possible (see §2.20).

Diversion 2.18.2. The proof makes uses of the requirement that p is prime. In particular, multiplication in the semigroup uses reduction modulo $p - 1$, which arises because of Fermat's little theorem $a^b \mod p = a^{(b \bmod p-1)} \bmod p$, for all a with $0 < a < p$.

Diversion 2.18.3. Earlier it was shown that Diffie–Hellman is not associative. Therefore equivalence does not preserve the associativity of key agreement.

2.19 Aside: a more general Diffie–Hellman realm

A **realm** is set with two binary operations, written as addition and multiplication (see §B.12). The **Diffie–Hellman realm** $R_{\mathrm{DH}}[S]$ of a multiplicative semigroup S consists of the following.

- Non-negative integers, written as usual, $0, 1, 2, \ldots,$, with the usual (standard) addition and multiplication.
- Infinity, written as ∞, with operations defined by the usual conventions $\infty + r = r + \infty = r\infty = \infty r = \infty$, for all other elements r (those above

and those below).

- Copies of elements S, written as $[s]$ for $s \in S$. Elements of $[S]$ operate among themselves as:

$$[s] + [t] = [st]$$
$$[s][t] = \infty$$

Elements of $[S]$ operate with positive (finite) integers as:

$$[s] + n = n + [s] = \infty,$$
$$[s]n = n[s] = [s^n].$$

Diversion 2.19.1. Addition and multiplication in $R_{\mathrm{DH}}[S]$ are each associative. (To be verified.) (Write $R_{\mathrm{DH}}[S]^+$ for its an additive semigroup (magma), $R_{\mathrm{DH}}[S]^\times$ for its multiplicative semigroup (magma).)

If S is commutative (or power-commutative, meaning that $(st)^n = s^n t^n$ for all $s, t \in G$ and all positive integer n), than multiplication in $R_{\mathrm{DH}}[S]$ is distributive over addition, making $R_{\mathrm{DH}}[S]$ a semiring. To see this, calculate $n([s]+[t]) = n([st]) = [(st)^n] = [s^n t^n] = [s^n] + [t^n] = n[s] + n[t]$, for example.

Diversion 2.19.2. Suppose S is a group. Suppose also that there exists a positive integer m, such that every element of S has an order dividing a m. We can then define a realm congruence on $R_{\mathrm{DH}}[S]$ in which the integers are reduced modulo m, which we call the **reduced Diffie–Hellman realm** of G.

Diversion 2.19.3. Associative key agreement using the multiplicative semigroup $R_{\mathrm{DH}}[S]^\times$ is essentially Diffie–Hellman (exponentiative) key agreement over the multiplicative semigroup S, as explained below.

The secrets a and c (of Alice and Charlie) will be integers. Values $b, d, e, f, g \in [S]$, say $b = [B]$, and $d = [D]$ and so on. So, $d = ab$ means that $[D] = a[B] = [B^a]$, for example.

2.20 Key agreement is essentially associative

All key agreement, not just Diffie–Hellman, is essentially associative:

Lemma 2.20.1. *Every key agreement scheme is essentially associative.*

Proof. Let $k = [k_1, k_2, k_3, k_4]$ be a key agreement scheme. The aim is to construct a semigroup, $S = S_k$, and subsets $A, B, C \subseteq S$ such that k is equivalent to the induced associative scheme $k^{A,B,C,S}$.

The high-level strategy to define semigroup S is to use the key agreement functions k_i as the multiplication operation wherever possible, and otherwise to make multiplication zero-valued (for a formal value 0).

Because the domains of the function k_i may overlap, with k_i and k_j taking inconsistent values on the overlaps, disjoint copies of the domains are needed to make multiplication well-defined.

Define a set $S = S_k$, with elements of the seven forms:

$$0, [1, a], [2, b], [3, c], [4, d], [5, e], [6, f],$$

where $[a, b, c, d, e, f]$ are sessions of k. Equivalently,

$$\begin{aligned} a &\in \mathrm{Dom}(k_1)_1, \\ b &\in \mathrm{Dom}(k_1)_2, \\ c &\in \mathrm{Dom}(k_3)_2, \\ d &\in \mathrm{Ran}(k_1), \\ e &\in \mathrm{Ran}(k_2), \\ f &\in \mathrm{Ran}(k_3) \cup \mathrm{Ran}(k_4). \end{aligned}$$

Define multiplication in S by

$$\begin{aligned} [1, a][2, b] &= [4, k_1(a, b)], \\ [2, b][3, c] &= [5, k_2(b, c)], \\ [1, a][5, e] &= [6, k_3(a, e)], \\ [4, d][3, c] &= [6, k_4(d, c)], \end{aligned}$$

whenever the inputs to the functions k_i on the right side of each equation belong to the correct domain; and otherwise

$$st = 0,$$

for all $s, t \in S$ (except for inputs s, t as listed above). Table 2.3 summarizes multiplication in S. Multiplication in S is associative because, if $[s, t, u] =$

	0	$[1,a]$	$[2,b]$	$[3,c]$	$[4,d]$	$[5,e]$	$[6,f]$
0	0	0	0	0	0	0	0
$[1,a]$	0	0	$[4,k_1(a,b)]$	0	0	$[6,k_3(a,e)]$	0
$[2,b]$	0	0	0	$[5,k_2(b,c)]$	0	0	0
$[3,c]$	0	0	0	0	0	0	0
$[4,d]$	0	0	0	$[6,k_4(d,c)]$	0	0	0
$[5,e]$	0	0	0	0	0	0	0
$[6,f]$	0	0	0	0	0	0	0

Table 2.3: Multiplication for S_k

$[[1,a],[2,b],[3,c]]$, then

$$\begin{aligned} s(tu) &= [1,a]([2,b][3,c]) \\ &= [1,a][5,k_2(b,c)] \\ &= [6,k_3(a,k_2(b,c))] \\ &= [6,k_4(k_1(a,b),c)] \\ &= [4,k_1(a,b)][3,c] \\ &= ([1,a][2,b])[3,c] \\ &= (st)u, \end{aligned}$$

and, otherwise, $s(tu) = 0 = (st)u$. (To see this, suppose $s(tu) \neq 0$. Then, $s = [i,a]$ for some a and some $i \in \{1,2,4\}$, by looking at the rows of Table 2.3 non-zero entries. Similarly, $tu = [j,e]$, for some e and $j \in \{2,3,5\}$, by looking at columns with non-zero entries. But $tu \neq 0$, so looking at the form all non-zero entries in the table, we get $tu = [j,e]$ with $j \in \{4,5,6\}$. Therefore $j \in \{2,3,5\} \cap \{4,5,6\}$, so $j = 5$. This forces $i = 1$ and $t = [2,b]$ and $u = [3,c]$ for some b and c. The argument for $(st)u$ is similar.)

Let

$$\begin{aligned} A &= \{[1,a] : a \in \mathrm{Dom}(k_1)_1\}, \\ B &= \{[2,b] : b \in \mathrm{Dom}(k_1)_2\}, \\ C &= \{[3,c] : c \in \mathrm{Dom}(k_2)_2\}. \end{aligned}$$

The associative key agreement scheme $K = k^{A,B,C,S}$ is equivalent to k, with session $[a,b,c,d,e,f]$ of k mapping to session

$$[[1,a],[2,b],[3,c],[4,d],[5,e],[6,f]].$$

of K. This gives an isomorphism, with efficient maps in both directions. □

2.21 Aside: packed associated semigroups

If key agreement k is associated with semigroup S, define a subset $S_k \subseteq S$, the set of **sessional** elements, by including each entry of each session $[a, b, c, d, e, f]$ of k, as mapped to elements of an associated semigroup S, via the equivalence of k to a subscheme of the multiplicative scheme k_S. The **sessionality** of S associated with k is the cardinality $|S_k|$ of the set of sessional elements. A semigroup S is a **packed** associated semigroup of k, if its sessionality is minimal among all associated semigroups of k.

For example, for Diffie–Hellman agreement mod p, the associated semigroup from the proof Lemma 2.20.1 has sessionality of $6p - 6$, while the associated semigroup from the proof of Lemma 2.18.1 has sessionality $2p-2$. Therefore, the former semigroup is not a packed associated semigroup of k, because the latter has lower sessionality.

Diversion 2.21.1. For key agreement schemes with an infinite number of sessions, the cardinality of the set of sessional elements is likely not very informative, but a more refined approach based on functions into the semigroup may be more informative.

Let S and T be two associated semigroup of S. Say S is at least as packed as T if there exists a function $p : T_k \to S_k$, such that for any entry x of any session $[a, b, c, d, e, f]$, such that x maps to $t \in T_k$ and to $s \in S_k$, then $p(t) = s$. Say that S is relatively packed if for all T at least as packed as S it holds that S is at least as packed as T.

For finite sessionality semigroups, being packed implies being relatively packed.

2.22 Aside: reduction to a category

Category theory is an abstraction that unifies many notions across modern algebra. Perhaps, it is then unsurprising that key agreement can be abstracted into category theory.

Recall that category has objects and morphisms between objects. Morphisms compose associatively. (There is also a unique identity morphism form each object to itself, but we shall not need that.) Write $\mathrm{Mor}(O, P)$, for the set of morphisms between two objects O, P. Write $f \circ g \in \mathrm{Mor}(O, P)$ for the composition of morphisms, $f \in \mathrm{Mor}(P, Q)$ and $g \in \mathrm{Mor}(O, Q)$. (This convention means the objects have opposite left-to-right ordering from the morphisms in a product.)

First, we construct a key agreement scheme from any category, and any four objects O_1, O_2, O_3, O_4 in the category. Define a key agreement scheme k as follows:

$$\begin{aligned}
k_1 &: \mathrm{Mor}(O_4, O_3) \times \mathrm{Mor}(O_3, O_2) \to \mathrm{Mor}(O_4, O_2) : [a, b] \mapsto a \circ b \\
k_2 &: \mathrm{Mor}(O_3, O_2) \times \mathrm{Mor}(O_2, O_1) \to \mathrm{Mor}(O_3, O_1) : [b, c] \mapsto b \circ c \\
k_3 &: \mathrm{Mor}(O_4, O_3) \times \mathrm{Mor}(O_3, O_1) \to \mathrm{Mor}(O_4, O_1) : [a, e] \mapsto a \circ e \\
k_4 &: \mathrm{Mor}(O_4, O_2) \times \mathrm{Mor}(O_2, O_1) \to \mathrm{Mor}(O_4, O_1) : [d, c] \mapsto d \circ c
\end{aligned}$$

Second, we construct a category with four objects O_1, O_2, O_3, O_4 from any key agreement scheme k.

Let $\mathrm{Mor}(O_i, O_i) = \{1_{O_i}\}$. Let $\mathrm{Mor}(O_i, O_j) = \{\}$ if $i < j$. Let $\mathrm{Mor}(O_4, O_3) = \mathrm{Dom}(k_1)_1$, meaning all a such that k_1 is defined for some input of the form $[a, b]$ (for some value of b). Similarly, let $\mathrm{Mor}(O_3, O_2) = \mathrm{Dom}(k_1)_2$ and $\mathrm{Mor}(O_2, O_1) = \mathrm{Dom}(k_2)_2$. Let $\mathrm{Mor}(O_4, O_2) = \mathrm{Im}(k_1)$ and $\mathrm{Mor}(O_3, O_1) = \mathrm{Im}(k_2)$ and $\mathrm{Mor}(O_4, O_1) = \mathrm{Im}(k_3)$. Define morphism composition as

$$\begin{aligned}
\circ &: \mathrm{Mor}(O_4, O_3) \times \mathrm{Mor}(O_3, O_2) \to \mathrm{Mor}(O_4, O_2) : [a, b] \mapsto k_1(a, b) \\
\circ &: \mathrm{Mor}(O_3, O_2) \times \mathrm{Mor}(O_2, O_1) \to \mathrm{Mor}(O_3, O_1) : [b, c] \mapsto k_2(b, c) \\
\circ &: \mathrm{Mor}(O_4, O_3) \times \mathrm{Mor}(O_3, O_1) \to \mathrm{Mor}(O_4, O_1) : [a, e] \mapsto k_3(a, e) \\
\circ &: \mathrm{Mor}(O_4, O_2) \times \mathrm{Mor}(O_2, O_1) \to \mathrm{Mor}(O_4, O_1) : [d, c] \mapsto k_4(d, c)
\end{aligned}$$

together with the usual category axiom that identity morphisms compose without effect.

Although category theory is important in modern algebra, it is not often viewed algorithmically.

2.23 Reduction to a ring

Any multiplicative semigroup is a subsemigroup of the multiplicative semigroup of a ring. For example, the ring can be taken as a semigroup ring of the semigroup. The semigroup ring contains a copy of the given semigroup.

Therefore, the session values of a key agreement can be represented as ring elements, and the key agreement functions as ring multiplications.

Such rings can be constructed that are larger than the given semigroup, sometimes much larger. This ring view has the potential risk that division in the very small subsemigroup is much easier than in the ring at large.

Diversion 2.23.1. Cash, Kiltz and Shoup [CKS08] introduced twinned Diffie–Hellman. This can be interpreted using the semigroup ring construction above. Essentially, what they propose amounts to taken a formal sum of two elements in the semigroup associated with Diffie–Hellman.

2.24 Reductionism

Because of Lemmas 2.11.1 and 2.20.1, a question about a key agreement can be translated into a question about semigroups, and a question about semigroups can be translated into a question about key agreement. In other words, key agreement theory is almost reduced to semigroup theory. The reduction is not perfect, by any means, for at least the following two major reasons.

- For each semigroup S, there are many associative key agreement schemes $k^{A,B,C,S}$.
- For each key agreement scheme k, there may be many different semigroups S, such that k is equivalent to an associative key agreement scheme $k^{A,B,C,S}$.

Diversion 2.24.1. A minor imperfection is that the main formal notions for equivalence of schemes ignore the overall range of the functions k_i, but in several applications of key agreement the ranges are important.

Diversion 2.24.2. The reduction is not claimed to be a functor in the mathematical sense of category theory.

Semigroup theory is older than key agreement. Optimism leads to a hope that semigroup theory is re-usable to understand key agreement. Pessimism leads to a doubt that most existing semigroup theory is not useful to understand key agreement. For example, existing semigroup theory likely

- addresses only mathematically nicer semigroups that would be insecure if used for associative key agreement,
- examines the underlying structure of semigroups existentially, without much focus on critical practical issues such as algorithms.

Diversion 2.24.3. The semigroup S constructed in the proof of Lemma 2.20.1 is arguably unnatural:

- its multiplication table has a patchwork appearance of mostly zeros with a four disconnected rectangular patches derived from the four key agreement functions, and
- the semigroup is 4-nilpotent, meaning $s^4 = 0$ for all $s \in S$.

The capacity for an unnatural semigroup to hold a nearly arbitrary patchwork of functions strongly suggests that a general theory of semigroups cannot prove useful to study such arbitrary key agreement.

Nonetheless, the general results suggests a hope that the following approach will be more useful than the generic approach of using Lemma 2.20.1.

Given key agreement scheme k, find the most natural semigroup S associated with k. The natural semigroup S, especially if it is group-like, may be more amenable to known semigroup theory. Similarly, form a ring R from S, and perhaps the ring theory will be useful to understanding R, then S, and then k.

Diversion 2.24.4. As far as naming the directions of translation can help, perhaps the terms below are most appropriate:

- to **descend** is to translate from key agreement to semigroups, from practice to theory, from applications to basics, from concrete to abstract, from engineering to mathematics, from divulgers to dividers, from focus on $[A, B, C, S]$ to focus on S;
- to **ascend** is to translate from semigroups to key agreement, from theory to practice, from basics to applications, from abstract to concrete, from mathematics to engineering, from dividers to divulgers, from focus on S to focus on $[A, B, C, S]$.

Under these terms, this report aims to descend from key agreement to the basics of semigroup theory and ring theory.

Chapter 3

Security aims of key agreement

Security aims for key agreement

- generalize well-known security aims of Diffie–Hellman key agreement, and its derived variants,
- specialize well-known security aims of more general definitions of key exchange,
- are formalized here for completeness (but coined new names to avoid the overloading old terminology, whether specific to Diffie–Hellman or generic to wider classes of key exchange), and
- can usually be reduced into the difficulty of certain semigroup problems (some of which, such as division, are well-known and much older than public-key cryptography).

Attack strategies against some of the basic security aims are deferred to Chapters 4 and 5 (and expressed in terms of semigroups).

3.1 Watchers (generalized DHP)

The main security aim for any key agreement scheme is to resist the next type of attack:

Definition 3.1.1. *A* ***watcher*** *of a key agreement scheme $k = [k_1, k_2, k_3, k_4]$ is a function W such that*

$$W(k_1(a, b), b, k_2(b, c)) = k_3(a, k_2(b, c)), \tag{3.1.1}$$

for all $[a, b, c]$ *with* $[a, b] \in \mathrm{Dom}(k_1)$ *and* $[b, c] \in \mathrm{Dom}(k_2)$.

Diversion 3.1.1. In the case of the Diffie–Hellman key agreement scheme (Definition 2.2.1), computing the watcher function is equivalent to the (variable-base) standard computational Diffie–Hellman problem, because in this case

$$W(d, b, e) = W((b^a \bmod p), b, (b^c \bmod p)) = (b^{ac} \bmod p).$$

3.1.1 Real world impacts of a watcher

The real-world impact of a watcher depends on how key agreement is used. Almost all key agreement usage makes the agreed key f a critical secret. An attacker who knows f would defeat security of the entire application of key agreement, in a non-recoverable, devastating way.

If Alice represents a user of k_1 and k_3 and Charlie represents a user of k_2 and k_4, then let Bugsy represent a user of W. Bugsy aims to compute $f = g$, using only b, $d = k_1(a)$ and $e = k_2(c)$.

3.1.1.1 Fake delivery (man-in-middle) attacks

The impact of a practical watcher algorithm should be compared to the impact of a simpler alternative attack, the **man-in-the-middle** attack, or more accurately speaking, the **fake delivery** attack, described below.

In this attack, Bugsy does not use a watcher W at all, but instead replaces Alice's or Charlie's delivery with his own. For example, Bugsy replaces Alice's delivery d by his own delivery d'. Bugsy has his own secret a', distinct from Alice's secret. Bugsy computes $d' = k_1(a', b)$. Bugsy computes agreed key $f' = k_3(a', e)$. Charlie's agree key is $g' = f'$. Of course, Charlie's key f' is different from the key f he would have computed, had Charlie been given Alice's actual delivery d.

Nonetheless, Charlie might think that he has agreed on a key g' with Alice, and that g' is known only to Alice and Charlie. Unfortunately for Charlie, it actually Bugsy who knows g', not Alice. To make Charlie think g' is known only to Alice and Charlie, Bugsy needs to deceive Charlie to think that the fake delivery d' is from Alice.

Bugsy can also work on both sides, making a **double fake delivery** attack. As above, Bugsy replaces d by d', but Bugsy also replace e by e'. Bugsy computes two distinct keys: f' and g'. Key f' is agreed with Alice, and key g' is agree with Charlie. Alice and Charlie's keys do not agree. At this point, Bugsy may run a **relay** attack. Bugsy intercepts encrypted traffic that Alice or Charlie sends. He decrypts with the appropriate key to get the plaintext. Then he re-encrypts, and re-authenticated to the other victim. All traffic between Alice and Charlie is relayed as if they had agreed on the same key $f = g$. Alice and Charlie are not likely to notice this, except perhaps for the subtle delays, or if they have a second layer of cryptographic defense, beyond just key agreement.

Bugsy's cost in this man-in-the-middle attack is not much more than the cost of the users Alice and Charlie. His computations are just the functions k_i. Perhaps his most difficult to task is to stop all the intermediate traffic, and replace it with this own version.

Thwarting these fake delivery attacks (in other words, man-in-the-middle attacks) is usually necessary in most applications of key agreement. A typical protection against these attacks is to apply a digital signature to the delivery of Alice or Charlie, or both their deliveries. (The signature verification key must be authentic for this to be effective.)

In many cases, only one of the deliveries is authenticated. The other delivery is consider anonymous, and is vulnerable to a fake delivery attack.

A common example is Alice using key agreement through TLS 1.3 inside a web browser. Alice is typically anonymous, with her (elliptic curve) Diffie–Hellman key agreement delivery not digitally signed. Charlie runs a web server, and his (elliptic curve) Diffie–Hellman key agreement delivery is authenticated, using a digital signature (such as ECDSA or RSA). Alice knows, cryptographically speaking, that she is talking to Charlie, due to Charlie's digital signature. On other hand, Alice is cryptographically anonymous. As far as Charlie knows, he could be taking to Bugsy, at least until some extra authentication steps are taken.

Many web server will have Alice authenticate herself to Charlie, but only after key agreement has been used to provide a private connection. For example, Alice sending Charlie a secret token, such as a (slow) hash of a password, or a cookie previously stored in Alice's web browser. Key agreement, Charlie's digital signature, and encryption together assure Alice that her secret token is only seen by Charlie, and not some adversary.

Instead of Charlie signing his own delivery, he may have a third party sign it, in which case the delivery is part of a certificate. Since the third party would likely only sign the delivery once, the certified delivery is static, and would re-used across many sessions. In this case, Charlie should worry about forward secrecy. To recover forward secrecy, he might use two sessions key agreement with Alice. The first, with the static certified delivery, is used only to authenticated, the ephemeral delivery.

The Menezes–Qu–Vanstone (MQV) key agreement is an instance can be considered as a merger of two agreement sessions, one ephemeral and one static. Instead of the static session authenticating the ephemeral session, only a single key is computed, derived as a combination of the various inputs. An alternative viewpoint is that that each MQV delivery has two parts, one part certified for authentication, the other part ephemeral, for forward secrecy.

Yet another mitigation against man-in-the-middle attacks is to use non-block-able deliveries. In this case, the authentication of the deliveries is not due to cryptography, but due to some physical means. For example, Alice and Charlie might presume that Bugsy might be able to send them fake deliveries, but is not able to block their own deliveries from arriving. If Alice and Charlie are expecting each other's deliveries, and only receive one delivery each, then then can deduce that the deliveries are not fake. If Alice or Charlie receives two deliveries, then one must be fake, so an ongoing attack can detected, and an appropriate alert raised.

Some systems overtly rely on similar forms of non-cryptographic authentication. A common form authentication is known as **trust-on-first-use** (TOFU), based on the notion that the Bugsy will not be ready to block deliveries from the initial communications between Alice and Charlie, perhaps because many there are too many pairs of peers initiating communications.

3.1.1.2 Privacy violation

If Alice (or Charlie) uses the agreed key f to encrypt a message to Charlie (or Alice), then Bugsy using a watcher can compute f and then decrypt the ciphertexts encrypted with f, and read the encrypted messages that were meant to be private.

Diversion 3.1.2. Encryption using an agreed key is a common application of variants of Diffie–Hellman key agreement. For example, the Transport Layer Security (TLS) version 1.3, requires use of elliptic curve Diffie–Hellman key agreement, and derives a encryption key from the agreed key, using the derivation method that Bugsy can replicate easily.

In this mode of attack, Bugsy is an entirely passive observer. Bugsy does not need to send any data to Alice or Charlie. Bugsy does not need to block or delay any traffic between Alice and Charlie. Bugsy need only observe d, b, e and the ciphertexts that to be decrypted with f.

In particular, this mode of attack is not thwarted by any level of authentication on the deliveries.

In this sense, it is more severely damaging than a man-in-the-middle attack. In other words, the extra impact of an efficient watcher function (over man-in-the-middle) is that it can break authenticated forms of key agreement, and it is arguably stealthier, because the attacker can maintain total network silence.

3.1.1.3 Impersonation

If Alice or Charlie uses the agreed key f to authenticate a message to the other, then using a watcher, Bugsy can compute f and forge any message. In other words, Bugsy can impersonate the sender.

Diversion 3.1.3. Authentication using agreed keys is a common application of variants of the Diffie–Hellman key agreement. For example, the Transport Layer Security (TLS) version 1.3, requires use of elliptic curve Diffie–Hellman key agreement, and derives an authentication key from the agreed key, using the derivation method that Bugsy can replicate easily.

In this mode of attack, after passively observing the key agreement values d, b, e, Bugsy begins to send out data to Alice or Charlie.

In some applications, Charlie will only trust the agreed key f once. In other words, Charlie will verify one single message from Alice per key agreement session.

In this case, for Bugsy to impersonate Alice, he must also block Alice's authenticated message to Charlie. Otherwise, Charlie will receive two authenticated messages, when he only expects one. Charlie will either be alerted to an ongoing impersonation effort, or will just reject one of the authenticated messages, perhaps Bugsy's forgery (which is likely if it arrives after Alice's message). If Charlie is careless in the sense of not looking for a second message, then Bugsy need merely arrange for the forgery to be delivered to Charlie before the truly authentic message. If the watcher algorithm is slow, then Bugsy may need to delay Alice's authenticated message, to give him time to evaluate the watcher function and produce an impersonated message.

The advantage over of this impersonation attack using a watcher, over the fake delivery (man-in-the-middle) attacks is that this impersonation attacks works even if the deliveries d and e are authenticated.

So, if Alice and Charlie authenticate their deliveries d and e, they can then safely the agree key f to authenticate messages, provided that Bugsy has no efficient watcher algorithm. In other words, key agreement enables the authentication of the initial deliveries to be transferred to the authentication of later messages.

With a watcher, Bugsy can defeat this transfer of authentication.

Diversion 3.1.4. Such authentication transfer is a common application of variants of Diffie–Hellman key agreement. The transport layer security protocol TLS 1.3 works this way. A server's Diffie–Hellman delivery is digitally signed, but then all payload message data is authenticated in the record layer using the agree keys.

3.1.2 Seclusive key agreement

Definition 3.1.2. *A key agreement scheme is* ***seclusive*** *if its watcher is infeasible.*

Diversion 3.1.5. A function f is **feasible** if it can be evaluated correctly using a feasible algorithm A.

A **feasible** algorithm means that A always runs and returns an answer within cost that its user (which is the attacker Bugsy in the case of a watcher function) can afford; **correct** evaluation means that every input x in the domain of the function, the algorithm returns the correct output $f(x)$.

These definitions have zero tolerance for exceptions. An algorithm is incorrect even if it is only wrong on one single output. An algorithm is infeasible even if its cost only exceeds the attacker's budget on one single input. The strictly deterministic setting does not allow for partially correct algorithms and partially feasible algorithms. These are allowed in the probabilistic setting, where meaning is formally attached how probable certain events are, and incorrectness rates and average costs can be quantified.

A function is **infeasible** if it is not feasible. Equivalently: every correct algorithm is infeasible; or, every feasible algorithm is incorrect. Due to the zero tolerance of exceptions, a function might be infeasible if just a single one of its inputs has an output that is not feasible to compute. For example, if one single output length has 2^{1000} bytes, then it is infeasible to produce this output, just due to its sheer length.

Such trivially infeasible functions will either not arise for watchers of efficient key agreement schemes, or will be quantifiably accounted for in the probabilistic setting.

Diversion 3.1.6. If no feasible algorithm to implement the watcher is yet known, then the key agreement can be called either **currently** or **conjecturally** seclusive, depending on the amount of effort spent trying to discover a feasible watcher algorithm.

Definition 3.1.3. *The* ***seclusiveness*** *of a key agreement scheme is the set of costs of algorithms to implement a watcher, especially the minimal costs.*

Diversion 3.1.7. Reasonable measures of algorithm cost include total energy consumption, monetary expenditures, custom hardware, memory usage, distributed communication, CPU cycles, and total time.

Diversion 3.1.8. The strict formalization of seclusiveness does not guarantee effective resistance to probabilistic watchers §3.1.5, where a feasible algorithm finds the agreed key for most (or many) choices of watcher input $[d, b, e]$.

Nonetheless, seclusiveness is generally a necessary condition for secure key agreement, a bare minimum bar to pass. Because seclusiveness can be defined and studied without reference of probability, its logical simplicity makes it a sensible and worthwhile security aim to isolate.

Furthermore, in some cases, the best known probabilistic watchers actually turn out to be only slightly better than deterministic watchers, anyway.

3.1.3 Watcher existence and uniqueness

For convenience, let

$$\begin{aligned} A(k) &= \{[a,b,c] : [a,b] \in \mathrm{Dom}(k_1), [b,c] \in \mathrm{Dom}(k_2)\}, \\ D(k) &= \{[k_1(a,b), b, k_2(b,c)] : [a,b,c] \in A(k)\}, \end{aligned}$$

so that the definition of a watcher W is an equation about the values W takes on inputs from $D(k)$.

Lemma 3.1.1. *For each key agreement scheme k, there exists a watcher W. If $R \supseteq \mathrm{Ran}(k_3)$, there exists at most one watcher of the form $W : D(k) \to R$.*

Proof. To show W exists, construct a watcher $W : D(k) \to \mathrm{Ran}(k_3)$ as follows. For each $[d, b, e] \in D(k)$, choose some $[a', b, c'] \in A(k)$ such that $[d, e] = [k_1(a', b), k_2(b, c')]$, which exists by the definition of $D(k)$. Let $W(d, b, e) = k_3(a', k_2(b, c'))$.

To show that W defined above is a watcher, consider any $[a, b, c] \in A(k)$, and consider $[d, b, e] = [k_1(a, b), b, k_2(b, c]]$ as an input to W. By definition of W,

$$W(d, b, e) = k_3(a', k_2(b, c'))$$

for the chosen a' and c', which may be distinct from a and c, but do satisfy

$k_1(a', b) = d = k_1(a, b)$ and $k_2(b, c') = e = k_2(b, c)$. Calculating

$$\begin{aligned} W(d, b, e) &= W(k_1(a, b), b, k_2(b, c)) \\ &= k_3(a', k_2(b, c')) \\ &= k_3(a', k_2(b, c)) \\ &= k_4(k_1(a', b), c) \\ &= k_4(k_1(a, b), c) \\ &= k_3(a, k_2(b, c)), \end{aligned}$$

so, W meets the definition of a watcher: it is correct for $[a, b, c]$, even though it was defined another choice $[a', b, c']$.

To see the uniqueness part, consider two functions $W, W' : D(k) \to R$. Let $[d, b, e] \in D(k)$. By definition of $D(k)$, $d = k_1(a, b)$ and $e = k_2(b, c)$ for some a and c. By definition of watchers, $W(d, e) = k_3(a, k_2(b, c))$, and $W'(d, e) = k_3(a, k_2(b, c))$. So, $W(d, e) = W'(d, e)$ for all $[d, b, e] \in D(k)$. In other words, $W = W'$. □

Because of this lemma, it is usually unambiguous to refer to **the watcher** of a key agreement scheme.

Diversion 3.1.9. Mathematicians might care and know whether existence of the watcher requires the axiom of choice for infinite-domain key agreement schemes, and perhaps even if existence of a watcher is equivalent to the axiom of choice.

3.1.4 Reduction of watchers to wedges

In defining key agreement, notions of equivalence and subschemes helped provide a reduction of key agreement to semigroups. This section shows that watchers undergo the same reduction.

3.1.4.1 Watchers of equivalent schemes

Let k and K be isomorphic key agreement schemes, as described in §2.15. The sessions of k and K are one-to-one correspondence: $[a, b, c, d, e, f]$ in k mapping to $[\alpha(a), \beta(b), \gamma(c), \delta(d), \eta(e), \phi(f)]$ in K, where $\alpha, \beta, \gamma, \delta, \eta, \phi$ are bijections.

Let w be a watcher for k. Define a watcher W for K by

$$W(D, B, E) = \phi(w(\delta^{-1}(D), \beta^{-1}(B), \eta^{-1}(E)).$$

Similarly, if W is a watcher for K, then

$$w(d, b, e) = \phi^{-1}(W(\delta(d), \beta(b), \eta(e)),$$

is a watcher for k.

If k and K are also equivalent, meaning the relevant bijections and their inverses are efficient, then the w is feasible if and only if W is. Therefore K is seclusive if and only if k is seclusive.

Diversion 3.1.10. Actually, if k and K are only **weakly equivalent**, meaning that bijection $\delta, \beta, \eta, \phi$ and their inverses are only feasible (not necessarily efficient), then the seclusiveness of k and K are basically equivalent, under the loose principle that the cost of running five feasible algorithms is still feasible (there being no sharp line between feasible and infeasible).

The arguments above and the previous lemmas imply the following.

Lemma 3.1.2. *Every seclusive key agreement scheme is equivalent to a seclusive associative key agreement scheme.*

Proof. Let k be a seclusive key agreement scheme. Lemma 2.20.1 means that k is equivalent to an associative scheme K.

If K were not seclusive, then a feasible watcher W for K would exist. By the arguments above, this would imply a feasible watcher w for k. That would mean that k is not seclusive, a contradiction. So, K must be seclusive. □

3.1.4.2 Watchers of subschemes

Subschemes were defined in §2.8.

Lemma 3.1.3. *If k is a subscheme of K, and W is a watcher of K, then w defined by*

$$w(d, b, e) = W(d, b, e)$$

for all $[d, b, e] \in \mathrm{Im}(k_1) \times \mathrm{Dom}(k_1)_2 \times \mathrm{Im}(k_2)$, is watcher of k.

Proof. When inputs to functions k_i are restricted to the appropriate values from sessions of k, each function k_i is a restriction of function K_i, so that $k_i(x, y) = K_i(x, y)$, since the output value of k_i is another value in the session, and every session of k is a session of K.

Therefore, the watcher condition on W transfers to w. Formally,

$$\begin{aligned} w(k_1(a,b),b,k_3(b,c) &= W(k_1(a,b),b,k_3(b,c)) \\ &= W(K_1(a,b),b,K_3(b,c)) \\ &= K_3(a,K_2(b,c)) \\ &= K_3(a,k_2(b,c)) \\ &= k_3(a,k_2(b,c)), \end{aligned}$$

which (repeating the arguments above) uses the fact that if $[a,b,c]$ generates a session $[a,b,c,d,e,f]$ of k, then this is also a session of K. □

If W is feasible, then w is feasible, using the same implementation algorithm for the function. If k is seclusive, then K is seclusive.

Diversion 3.1.11. The converse can fail: a seclusive key agreement scheme might have an inseclusive subscheme.

In particular, every seclusive associative key agreement scheme is a subscheme of a seclusive multiplicative key agreement scheme. Combining this with Lemma 3.1.2 gives the next result.

Lemma 3.1.4. *If k is seclusive, then there exists a semigroup S such that key agreement scheme $K = k^S$ is seclusive, and k equivalent to a subscheme of k^S.*

3.1.4.3 Watcher by wedges

In multiplicative key agreement, a watcher is exactly the wedge operation $[ab, b, bc] \mapsto abc$ of the associated semigroup. (The wedge operation is defined more formally in §B.9.)

This equivalence is formalized as the next two lemmas, one for each direction:

Lemma 3.1.5. *If S is a semigroup, and k^S is its associated multiplicative key agreement scheme, and W is a watcher for k^S, then*

$$d \wedge_b e = \begin{cases} W(d,b,e) & \text{if } W(d,b,e) \text{ is defined} \\ b & \text{if } W(d,b,e) \text{ is not defined} \end{cases}$$

defines a wedge operator for S.

Lemma 3.1.6. *If S is a semigroup, and k^S is its the associated multiplicative key agreement scheme, and $\wedge$ is a wedge operator for S, then*

$$W(d, b, e) = d \wedge_b e$$

defines a watcher for k^S.

Combining with Lemma 3.1.4, gives the following:

Lemma 3.1.7. *If k is a seclusive key agreement scheme, then there exists a semigroup S with no feasible wedge operator, and k is equivalent to a subscheme of k^S.*

This shows that the reduction of key agreement to semigroups also reduces seclusiveness to infeasible wedge operators.

Diversion 3.1.12. The converse can fail: even if a semigroup S has no feasible wedge operator, the multiplicative key agreement scheme k^S can have subschemes that are not seclusive. For example, the subscheme $k^{\{a\},\{b\},\{c\},S}$ has a constant functions $w(d, b, e) = abc$ for a watcher.

Instead, a partial converse holds. A watcher of a scheme implies an incomplete wedge (§B.9.3) against an associated semigroup.

Chapter 4 lists some strategies to compute wedges.

3.1.4.4 Faulty watchers

Although a watcher W always exist for any k, it might be infeasible. Instead one may wish to relax the condition on W, and define a **faulty watcher** W' as some arbitrary function. Faulty watchers should be measured somehow, and it would be best to measure their success against probabilistic key agreements.

This would be a straightforward to do, but it seems better to consider a more general definition of a probabilistic watcher, defined in the next section.

3.1.5 Probabilistic watchers

Probabilistic watchers are modified watchers, defined formally as:

Definition 3.1.4. *A* ***probabilistic watcher*** *a pair $[p, W]$, where p is a random variable, and W is a function.*

Diversion 3.1.13. Generally, while a deterministic watcher is a ternary function, the function W in a probabilistic watcher is quaternary function, with the first input being taken as a sample of the random variable.

Diversion 3.1.14. Given that deterministic watchers exist, it is natural to question the need for probabilistic watchers at all. One reason is that through randomization, greater efficiency is possible.

Probabilistic watchers can target both probabilistic and deterministic key agreement schemes.

3.1.5.1 Success rate against probabilistic key agreement

A probabilistic watcher can target probabilistic key agreement scheme, with success rate defined:

Definition 3.1.5. *The* ***success rate*** *of* $[p, W]$ *against* $[a, b, c, k]$ *is the probability*

$$\sigma_{[p,W]}([a, b, c, k]) = \Pr[W(p, k_1(a, b), b, k_2(b, c)) = k_3(a, k_2(b, c))], \qquad (3.1.2)$$

taken over the random variables a, b, c, p.

Diversion 3.1.15. Implicit in the definition of success rate is that function W might not even be defined for some sessions of k. Undefined cases of W, where the inputs to W are not part of the domain of W, contribute nothing to the success rate. In other words, the success rate accounts only for these inputs to W with defined outputs, measuring only the rate of correct outputs.

3.1.5.2 Success rate against deterministic key agreement

A probabilistic watcher can target deterministic key agreement scheme k, with success rate defined:

$$\sigma_{[p,W]}(k) = \inf_{[a,b,c]} \Pr[W(p, k_1(a, b), b, k_2(b, c)) = k_3(a, k_2(b, c))], \qquad (3.1.3)$$

where the $[a, b, c]$ range over all $[a, b, c]$ possible in sessions of k.

Diversion 3.1.16. Success rate against probabilistic key agreement is at least the success rate against any deterministic key agreement

$$\sigma_{[p,W]}([a, b, c, k]) \geq \sigma_{[p,W]}(k) \qquad (3.1.4)$$

due to the deterministic success rate being defined with an infimum.

Diversion 3.1.17. Using a supremum instead of an infimum leads to a vacuous definition, with success rate always equal to 1. Choose p to be a constant random variable $[p_1, p_2, p_3] = [a_0, b_0, c_0]$, and define $W(q, d, b, e) = k_3(q_1, k_2(q_2, q_3))$. At $[a, b, c] = [a_0, b_0, c_0]$, the probability on the right of the defining equation is 1. With a supremum-based definition, the success rate would be one.

3.1.5.3 Faulty watcher as a probabilistic watcher

A **faulty watcher** w of k is any ternary function that is not a watcher. Each faulty watcher induces a probabilistic watcher $[p, W]$ defined by $W(q, d, b, e) = w(d, b, e)$, where p is any random variable.

Conversely, if a probabilistic watcher $[p, W]$ has the property that $W(q, d, b, e) = W(r, d, b, e)$ for all q and r, then say that $[p, W]$ is **deterministic**. Letting $w(d, b, e) = W(q, d, b, e)$, then w is either a watcher or a faulty watcher.

Essentially, a faulty watcher is a probabilistic watcher, which does not use its own randomness. A faulty watcher can have success rate against probabilistic key agreement. Because a faulty watcher is not a watcher, the success rate of its induced probabilistic watcher against deterministic key agreement is zero.

3.1.5.4 External probabilistic watchers

Probabilistic watcher $[p, W]$ is **separated** from probabilistic key agreement scheme $[a, b, c, k]$ if random variable p is independent of random variables a, c.

A probabilistic watcher separated from a probabilistic key agreement scheme models the ideal situation that no information leaks about Alice and Charlie's secret random variables a and c leaks to the adversary.

3.1.5.5 Probabilistic seclusiveness

Definition 3.1.6. *A probabilistic scheme $[a, b, c, k]$ is π-**seclusive** if no feasible probabilistic watcher $[p, W]$ separated from $[a, b, c, k]$ has success rate $\sigma \geq \pi$ against $[a, b, c, k]$.*

Diversion 3.1.18. A probabilistic watcher is feasible if both p and W are feasible. As always, a function, in this case W, is feasible if it can be implemented correctly by an algorithm running with a cost affordable to the attacker.

A random variable, in this case p, is feasible if the attacker has affordable means to sample p. This might rule out a random variable p that depends so much on the secret random variables a and c that its value reveals one of the vales a or c.

Diversion 3.1.19. To avoid a technically vacuous definition of seclusiveness, we have restricted to probabilistic watchers that are separated from the target scheme. If we had not made this restriction, then we could have chosen p as the random variable $[p_1, p_2, p_3] = [a, b, c]$, and the function W as $W(p, d, b, e) = k_3(p_1, k_2(p_2, p_3))$.

Consequently, our definition of seclusive of probabilistic key agreement does not at all address adversaries that somehow obtain leakage of information about a and c.

Because there is no sharp line between feasible and infeasible, there may well be no sharp line marking π-seclusiveness.

Diversion 3.1.20. A π-seclusive deterministic key agreement scheme k can be defined similarly, using the success of probabilistic watchers $[p, W]$ against k.

It seem simpler use seclusiveness for deterministic key agreement, and π-seclusiveness of probabilistic key agreement, without mixing the probabilistic and deterministic cases.

3.1.5.6 Costs of probabilistic watchers

To be completed.

The average cost of probabilistic watcher $[p, W]$ means the average total cost of sampling p and implementing W, where the average is computed over $[a, b, c, p]$.

The worst-case cost of probabilistic watcher $[p, W]$ means supremum cost of sampling p and implementing W, where the maximum is taken over all values of $[a, b, c, p]$.

Diversion 3.1.21. The π-seclusiveness of a probabilistic key agreement scheme can be defined as the set of costs of all probabilistic watchers of the scheme with success $\sigma \geq \pi$.

3.1.5.7 Success rate 1 probabilistic watchers

Watchers and probabilistic watchers of success rate 1 are closely related, as one might expect.

Lemma 3.1.8. *If W is a watcher of k, then $[p, w]$ for w defined by*

$$w(q, d, b, e) = W(d, b, e),$$

is a probabilistic watcher with success rate 1 against all probabilistic key agreement of the form $[a, b, c, k]$.

Proof. Suppose W is a watcher of k, and w is defined as $w(q, d, b, e) = W(d, b, e)$. Then, $\sigma_{[p,w]} = 1$, by inspection of the definitions. □

3.1.5.8 De-randomization

A probabilistic watcher $[p, W]$ of $[a, b, c, k]$ is **de-randomizable** if there is exists a function Q such that:

$$w(d, b, e) = W(Q(d, b, e), d, b, e)$$

gives a watcher w of k. In this case, we call Q a **de-randomizer**.

Lemma 3.1.9. *If $[p, W]$ is a probabilistic watcher of $[a, b, c, k]$ with success rate 1, and the support of $[a, b, c, k]$ is k, then there exists a de-randomizer.*

If p is also independent of $[a, b, c]$, then there exists a constant de-randomizer.

Proof. Let Q be a function with an everywhere-nonzero conditional probability $\Pr[p = Q(d, b, e)|k_1(a, b) = d, b, k_2(b, c) = e]$. Such a Q exists by taking the value with highest conditional probability. Because the support of $[a, b, c, k]$ is k, the function Q is well-defined, because condition (defining the conditional probability) has nonzero probability.

If w defined by $w(d, b, e) = W(Q(d, b, e), d, b, e)$ fails to be a watcher, then for some $[a, b, c]$, with $d = k_1(a, b)$ and $e = k_2(b, e)$ and $f = k_3(a, e)$, we have $w(d, b, e) \neq f$.

Because the support of $[a, b, c, k]$ is k, the session $[a, b, c, d, e, f]$ has nonzero probability, say ϵ. The conditional probability that $p = Q(d, b, e)$ is nonzero too, say ϕ. This implies that the success rate of W is at most $1 - \epsilon\phi$, which is less than 1.

If p is independent of $[a, b, c]$, then all the conditional probabilities for each output $Q(d, b, e)$ do not depend on the input $[d, b, e]$, so Q can be chosen as constant. □

3.1.5.9 Repetition (to be corrected)

To be corrected and completed.

A probabilistic watcher $[p, W]$ with success rate σ can be iterated n times by taking n independent copies of p, say $[p_1, \ldots, p_n]$.

For input $[d, b, e]$, this gives n different outputs $f_i = W(p_i, d, b, e)$. For some $[p, W]$, the probability that one of f_i is correct might be $1 - (1 - \sigma)^n$. If testing the correctness of f_i is easy (such as by decrypting a message encrypted with f), then $[p, W]$ with small success rate σ could have its success rate increased, through iteration, to approximately $1 - e^{-n\sigma}$.

3.1.5.10 Probabilistic watchers of equivalent schemes

To be completed.

If key agreement schemes k and K are (weakly) equivalent, then a probabilistic watcher $[p, w]$ with success rate σ against k, can be converted into a probabilistic watcher $[p, W]$ of K of the same success rate, by applying the suitable session transformation functions to the inputs and outputs of w.

3.1.5.11 Probabilistic watchers of subschemes

To be completed.

Recall that a seclusive key agreement scheme K can have a non-seclusive subscheme k. The same failure can happen for π-seclusive schemes. A subschemes of a π-seclusive scheme can be fail to be π-seclusive.

Recall that if scheme K has a seclusive subscheme k, then K is seclusive. In other words, seclusiveness is inherited from a subscheme. But π-seclusiveness is not inherited from a subscheme. A scheme K can have a subscheme k that is π-seclusive, even though K is not π-seclusive. For example, K could have the property that, outside of sessions drawn from k, all other sessions have the same non-secret agreed key, and furthermore, the number of these non-k session vastly outnumbers those k, with a probability of session of K being a non-k close to one.

This observation about drastic differences between π-seclusiveness of a scheme and its subscheme weakens the strength of our reduction of key agreement to semigroups. In some sense, the observation shows how the difficulty of attacks hinges not so much on the whole semigroup, but rather special sets of inputs. However, as long as we keep this mind for the later, we can first attempt to solve semigroup problems as though the average difficulty did not vary with the inputs, and then later account the varying average difficulty as needed.

3.2 Divulgers (generalized DLP)

The discrete logarithm problem (DLP) must be difficult for Diffie–Hellman security. The DLP can be generalized as follows.

Definition 3.2.1. *A **(right) divulger** is a function $L = L_1$ such that*

$$k_1(L_1(k_1(a, b), b), b) = k_1(a, b) \tag{3.2.1}$$

for all $[a, b] \in \mathrm{Dom}(k_1)$. *A* ***(left) divulger*** *is a function* L_2 *such that*

$$k_2(b, L_2(b, k_2(b, c))) = k_2(b, c) \tag{3.2.2}$$

for all $[b, c] \in \mathrm{Dom}(k_2)$.

Diversion 3.2.1. In Diffie–Hellman key agreement, a discrete logarithm solver provides a divulger:

$$L_1(d, b) = \log_b(d) = \log_b^{\mathrm{mod}\ p}(d)$$

because $k_1(a, b) = b^a \bmod p$, so

$$k_1(L_1(k_1(a, b), b), b) = b^{\log_b(b^a)} = b^a = k_1(a, b).$$

Conversely, L_1 provides a discrete logarithm solver.

Diversion 3.2.2. A mnemonic for this report's use of the jargon **divulger** is division (to be seen later), and logarithm.

The divulgers L_1 and L_2 make use of the input b, so when further clarification is needed, we call L_1 and L_2 **basic** divulgers.

3.2.1 Variant divulgers

Various less important types of divulgers can be defined similarly. Despite being less important, their similarity to basic divulgers, means that they share the relationship to division (and logarithms).

For example, a **right keyed divulger** L_3 such that

$$k_3(L_3(k_3(a, e), e), e) = k_3(a, e), \tag{3.2.3}$$

which takes as its first input the agreed key $k_3(a, e)$, which would normally be a secret. A function L_4 can be defined similarly. A reason the keyed divulgers are less important for most of applications of key agreement is that their inputs include values that the key agreement users would normally keep secret, and an attacker who has these secrets is often considered to have won.

Each of the function definitions for L_i can be reversed, allowing **reverse divulgers** to be defined

$$k_1(a, L'_1(a, k_1(a, b))) = k_1(a, b) \tag{3.2.4}$$

$$k_2(L'_2(k_2(b, c), c), c) = k_2(b, c) \tag{3.2.5}$$

$$k_3(a, L'_3(a, k_3(a, e))) = k_3(a, e) \tag{3.2.6}$$

$$k_4(L'_4(k_4(d, c), c), c) = k_4(d, c) \tag{3.2.7}$$

The reverse divulgers take as input one of the private keys a or c, which is even more unlikely and severe than an attacker who gains access to an agreed key. The reverse divulgers L'_1 and L'_2 recover a value b' which is effectively equivalent to b. In most cases, this is harm, since b is public. In a few cases, b is secret, but an alternative b' is harmless, unless $b' = b$. The reverse divulger L'_3 finds an alternative e' equivalent to e. This seems harmless if e is public or if $e' \neq e$ for secret e.

3.2.2 Watchers from divulgers

If L_1 is a divulger, then a watcher W can be constructed as:

$$W(d, b, e) = k_3(L_1(d, b), e). \tag{3.2.8}$$

To see that this is a watcher, let $a' = L_1(k_1(a, b), b)$ and compute

$$\begin{aligned} W(k_1(a, b), b, k_2(b, c)) &= k_3(L_1(k_1(a_1, b), b), k_2(b, c)) \\ &= k_3(a', k_2(b, c)) \\ &= k_4(k_1(a', b), c) \\ &= k_4(k_1(L_1(k_1(a, b), b), b), c) \\ &= k_4(k_1(a, b), c) \\ &= k_3(a, k_2(b, c)) \end{aligned}$$

by the properties of L_1 and k.

Diversion 3.2.3. Resisting watchers requires resisting to divulgers.

Diversion 3.2.4. For some key agreement schemes k, the best known way to construct a watchers W is the construction above of using a divulger. For example, this is essentially the case for (elliptic curve) Diffie–Hellman key agreement.

3.2.3 Extra impact of divulgers

Suppose that Alice re-uses the same a in more just then one single session of k. Perhaps Alice uses a in multiple different session of k. Perhaps, Alice uses a in systems related to k, in their use of k_1. For an example of the latter for Diffie–Hellman, Alice may opt to use her Diffie–Hellman secret a as a digital signature key.

A divulger jeopardizes the re-usage of a, worse than a watcher alone would jeopardize re-usage of a.

In attacks against re-use of a and k, a divulger has a more impact than a watcher, because the divulger need only be evaluated once, whereas a watcher must be evaluated for each e.

In attacks against re-use of a outside of k, finding a (or an equivalent) should undermine any benefit that Alice gets from the secrecy of a. For example, if a is used a digital signature key, and attacker could use divulger to find a (or an equivalent), then forge signatures of Alice. By comparison, a watcher alone might not have such an impact.

3.2.4 Extra assurance against divulgers

In applications where divulger does more damage than a watcher, it is may be desirable to have better security assurance against divulgers. Such security assurance is generally achieved by studying divulgers more intensely than studying watchers.

In the case of Diffie–Hellman, the divulgers are discrete logarithm solvers, we arguably have more confidence in the difficulty of divulger than in the difficulty of watchers, which are computational Diffie–Hellman problem solvers.

In order to discuss this clearly, we can name resistance to divulgers as **exclusiveness**.

Diversion 3.2.5. Exclusiveness is essentially equivalent to the function $a \mapsto k_1(a, b)$ being **one-way**, or more formally, **second pre-image resistant**.

Many one-way functions are known in cryptography, where very few secure key agreement schemes are known. This makes second pre-image resistance seem something easier to achieve. This generally makes exclusiveness more plausible than seclusiveness.

Diversion 3.2.6. A mnemonic for the jargon **exclusiveness** is seclusiveness and exponentiation (as in the opposite of a logarithm, as in a divulger).

Seclusiveness requires exclusiveness, but sometimes:

- exclusiveness can be more critical than seclusiveness, because extra impact of divulgers;

- exclusiveness is better understood than seclusiveness, resulting in greater confidence and security assurance in exclusiveness.

3.2.5 Aside: instance-verifiability of divulgers

A divulger L_1 is **instance-verifiable** meaning that the correctness of any valid instance of the output $L_1(d, b)$ can be verified without knowledge of any secrets, by checking that $k_1(L_1(d, b)) = d$.

Diversion 3.2.7. A watcher W is not generally instance-verifiable. An exception is if a distinguisher (discussed) is efficient.

A watcher can be empirically tested by choosing inputs $[d, b, e]$ from sessions whose secrets a and c chosen by the tester.

Diversion 3.2.8. Instance-verifiability helps somewhat in creating un-cheatable challenges, for example, by selecting d and b as pseudorandom hashes of short strings.

However, instance-verifiability alone is not enough in general to generate such challenges, because for random d and b, there might not exist any a at all such that $d = k_1(a, b)$.

3.2.6 Reduction of divulgers to division

In multiplicative key agreement k^S, for multiplicative semigroup S, a divider in S (an operator $/$ such that $((ab)/b)b = ab$ for all $a, b \in S$, see §B.8 for formal definitions) is equivalent to a divulger:

$$L_1(d, b) = d/b.$$

Left divulgers are equivalent to left dividers:

$$L_2(b, e) = b\backslash e.$$

Chapter 5 gives various strategies to divide in semigroups.

3.2.7 Divulgers by exhaustive search

Divulgers exist, for reasons similar to the existence of watchers.

Given a well-ordering of the set $\mathrm{Dom}(k_1)_1$, where the divulger output should lie, there exists a minimal divulger defined

$$L_1(d, b) = \min\{a : k_1(a, b) = d\}$$

If $\mathrm{Dom}(k_1)_1$ is countable, and there is an algorithm to enumerate its elements as $a_0, a_1, \ldots,$ then the minimal divulger can be implemented by an algorithm. Such an algorithm is called **exhaustive search**.

Exhaustive search can only be feasible if $\mathrm{Dom}(k_1)_1$ is finite, with a size n small enough that the adversary can afford to do n computations of k_1. This finiteness requirement is because applying exhaustive search to deterministic key agreement, we drop all notions of probability, and insist the algorithms must always run in a feasible time.

3.2.8 Divulgers by trial search

Probabilistic divulgers can also be defined, similarly to probabilistic watchers.

Probabilistic divulgers can be applied to probabilistic key agreement. Probabilistic divulgers can greatly expand what is considered as feasible attacks. By using a notion of success rate, cases take a long time for an implementation can be discounted if they are somehow rare.

Deterministic exhaustive search generalizes to probabilistic trial search. Rather than searching a fixed list, the search list can be generated probabilistically. A notion of success rate, incorporating the choice of this list and probabilistic key agreement.

Trial search, also well-known, is reviewed in Chapter D.

3.2.9 Divesters

A **divester** is a divulger L_1 such that

$$L_1(k_1(a, b), b) = a, \tag{3.2.9}$$

for all $[a, b] \in \mathrm{Dom}(k_1)$. A left divester is defined similarly from a left divulger.

A divester has potentially more impact than a general divulger because it can find the exact value of a, instead of an effectively equivalent value a'. This means that a could be attacked even if a used in systems totally unrelated to k.

Ideally, the secret value a should not be used outside of the system using k. With this ideal practice, a divester should have no more impact than a divulger. Nonetheless, in complicated systems, this is ideal might be difficult to achieve.

In multiplicative key agreement, a divester corresponds to a post-divider (see §B.8.9).

A **set-divester** is the a function L'_1 such that

$$a \in L'_1(k_1(a, b), b), \tag{3.2.10}$$

In other words, a set-divester is like a divester except that it outputs a set containing the target value a. A set-divester always exists.

Diversion 3.2.9. A divulger can essentially be characterized by the condition that $L_1(k_1(a, b), b) \in L'_1(k_1(a, b), b)$, where L'_1 is a set-divester.

The impact of the set-divester on a key agreement depends on both the size of the output set $L'_1(k_1(a, b), b)$, and the ability of the attacker to enumerate through the set to test and confirm the target a. If a was used for some other purpose – such as another session of the key agreement scheme but with a different value of b or such as to derive a key for a completely different kind of cryptography – an attacker may be able to confirm the value of a and thereby compromise the other cryptographic application of a.

A set-divester is essentially equivalent to compute a division-set, see §B.8.11.1.

3.2.10 Divergers

A **diverger** is a divulger L_1 such that

$$k_1(L_1(d, b), b) = d, \tag{3.2.11}$$

for any $d \in \mathrm{Ran}(k_1)$. A left diverger is defined similarly from a left divulger.

Divergers generally do not have more impact than a general divulger. An exception might be if the key agreement scheme is used in a specialized way. For example, an application might somehow arrange to embed meaningful information into d, as an extra feature. In this case, a diverger might be able to attack this extra feature, by embedding a false d' with some arbitrarily malicious information, and then using a diverger to find an associated value a'.

In multiplicative key agreement, a diverger corresponds to a pre-divider (see §B.8.10).

3.2.11 Weak-input divulgers

Weaker versions divulgers L_i can be defined, where the success domain for L_i is made smaller than the domain of k_i.

For example, we can narrow the definition of L_1 by restricting b to a smaller set B of **weak bases**. So,

$$k_1(L_1(k_1(a,b),b),b) = k_1(a,b) \tag{3.2.12}$$

for all $[a,b] \in \mathrm{Dom}(k_1)$ with $b \in B$. A weak base divulger is not as powerful as a general divulger L_1, because it only succeeds if $b \in B$. But there might exist B such that a weak-base divulger is much more efficient than than a general basic divulger.

Diversion 3.2.10. In the case of Diffie–Hellman key agreement, examples of weak base divulger include using a base b that has low multiplicative order.

Weak-base divesters can also be defined.

Diversion 3.2.11. In some cases, user Alice re-uses a many times. This might lead to an **iterative** weak-base divulger. (See section of esoteric security aims).

In many applications of key agreement, the base value b must be authenticated, and is perhaps even fixed. Forcing b into a weak base might be infeasible for the attacker.

A **weak-delivery** divulger is like the keyed divulger L_3, except that success is only required when input d belongs to a set D of **weak delivery** values.

3.3 Distinguishers (generalized DDHP)

Some applications of key agreement need the agreed key to be indistinguishable from agreed key of any other session. This is one way to generalize the decision Diffie–Hellman problem.

Definition 3.3.1. *A (**left**) **distinguisher** for k is a function U such that*

$$U(d,b,e,\{f,f'\}) = f \tag{3.3.1}$$

for all $[a,a',b,c,d,d',e,f,f']$ such that both $[a,b,c,d,e,f]$ and $[a',b,c,d',e,f']$ are sessions of k.

Diversion 3.3.1. To clarify, the last input of the distinguisher is a set, using standard mathematical notation, which is not ordered: in particular, $\{f,f'\} = \{f',f\}$.

(By contrast, several computer languages, including C, use braces to specify an ordered array or other ordered data structures, rather than for an unordered set.)

Diversion 3.3.2. As was the case for watchers, many applications wish to avoid probabilistic version of adversaries, at least those with sufficiently large success rates.

So, a **probabilistic distinguisher** could be formalized to address this. Informally, one probabilistic distinguisher outputs one element of $\{f, f'\}$ uniformly at random. Its probability of being correct is 1/2. In most applications, such as probability 1/2 is not considered a successful, because the threat model already accommodates this risk. So, instead success rate would measured by advantage over the base correctness probability.

3.3.1 Impact of distinguishers

To be clarified!

The impact of a distinguisher depends on how the agreed key f is used.

Some applications use f as a key in a second cryptographic scheme E, such as an authenticated encryption algorithm. If this second scheme is robust enough, then a distinguisher U against k does not immediately result in an attack against the combination of k and E. For example: E might tolerate biased keys, such as by hashing its keys before using them; E have good security provided that its key f is sufficient secrecy (entropy unknown to the attacker).

Some applications might use f as in a second cryptographic scheme E' that is less robust. An example of E' is the Vernam cipher, also know as one-time-pad. In this case, E' treats f as a key-stream. However, E' generally does not tolerate bias in the key-stream f. Key-stream bias in the candidate key-stream can be used to confirm the correctness of the candidate message. A distinguisher U can be used detect the bias, and confirm the message.

Diversion 3.3.3. In the Vernam cipher, the cipher z for message m is computed as $z = m + f$, for some form of addition (not necessarily associative). Decryption with f is defined by $m = z - f$. So, subtraction should obey the axiom $(m + f) - f = m$ (for all relevant m and f needed by users).

Suppose z is the observed ciphertext, encrypted with f from a key agreement session $[a, b, c, d, e, f]$. Suppose that an attack knows a set $\{m, m'\}$, meaning the message sent is narrowed down to one of two possible messages.

Suppose the addition is also equipped with a left (post)-subtraction operator, say $\sim$, such that $x \sim (x+y) = y$ for all x, y. If addition is commutative, then just let $x \sim y = y-x$, using the same right subtraction operation that decryption uses.

With left subtraction, the attacker can recover two guesses for the agreed key $f = m \sim z$ and $f' = m' \sim z$.

If both candidate agreed keys f and f' are compatible with sessions of k, then a distinguisher U can then be used to determine which of f or f' matches the observed d, b, e values of the session.

The match of f or f' determines the match m or m'.

Diversion 3.3.4. Another way to use a distinguisher, continuing from the previous note, consists of testing whether an observed ciphertext z is for a guessed message m. One call to the distinguisher is used for each guessed message.

Given z and message guess m, that attacker chooses some a', computes his own $f' = k_3(a', e)$, effectively generating an alternative hypothetical session $[a', b, c, d', e, f']$ (computing d' is unnecessary for this attack). The attacker then runs the distinguisher $U(d, b, e, \{m \sim z, f'\})$. If the guessed message m is correct, then $f = m \sim f$, and U should return f, by definition.

If guess m is incorrect, then $f \neq z - m$, so distinguisher U is not guaranteed to produce valid output. It seems reasonable to suspect that U in this setting might return a pseudorandom element of $\{m \sim z, f'\}$. If so, then a correct guess is more likely to return $m \sim z$, so U returning $m \sim f$ means that m is more likely correct than not.

Key derivation is a well-known, general-purpose transformation that is used, among other things, to try to produce unbiased output from biased inputs. Generally, key derivation functions are one-way and pseudorandom, and so on.

Diversion 3.3.5. A stream cipher also does a comparable task, taking a small secret key, and producing a key-stream that is difficult to distinguish from a uniformly distributed a stream of the same length.

Given a key agreement scheme k, and key derivation function F (or a stream cipher), a second key agreement scheme k' can be obtained by replacing each agreed key f of k, by $f' = F(f)$. Due to the properties of a strong key derivation F, this seems to defeat most distinguishers.

3.3.2 Distinguishers from watchers

A distinguisher can be constructed from a watcher, by letting:

$$U(d, b, e, \{f, f'\}) = W(d, b, e). \tag{3.3.2}$$

This distinguisher U has the same cost to evaluate as the watcher W.

3.3.3 Variant distinguishers

Some minor variations on the definition of distinguishers are natural.

3.3.3.1 Right distinguishers

A **right** distinguisher U is defined similarly by the same equation

$$U(d, b, e, \{f, f'\}) = f$$

but the equations needs to hold whenever $[a, b, c, d, e, f]$ and $[a, b, c', d, e', f']$ are sessions of k.

Diversion 3.3.6. Several obvious extensions are possible. For example, a **left-and-right** distinguisher

$$U(d, b, e, \{f, f'\}) = f$$

whenever $[a, b, c, d, e, f]$ and $[a', b, c', d', e', f']$ are sessions. For another example, a **left-with-right** distinguisher:

$$U(d, b, e, \{f, f', f''\}) = f$$

whenever there exist sessions $[a, b, c, d, e, f]$ and $[a', b, c, d', e, f']$ and $[a, b, c'', d, d'', f'']$.

3.3.3.2 Replay distinguishers

A **replay** distinguisher U gets an extra input, the variable d'.

For the previous distinguishers, the alternative values a', d', f' are only hypothetical values that Alice could have used, but did not use. For the replay distinguisher, a', d', f' are a second set of values Alice actually used.

The replay distinguisher has replayed Charlie's delivery e (hence the name). (It might be infeasible for Alice to test Charlie's deliveries for repeats.)

The right replay distinguisher gets input e' instead of d'.

3.3.3.3 Session distinguishers

A **session** distinguisher U_2 attacks two independent sessions, meaning

$$U_2(d, b, e, d', b', e', \{f, f'\}) = f \qquad (3.3.3)$$

whenever $[a, b, c, d, e, f]$ and $[a', b', c', d', e', f']$ are sessions of k.

A **baseless** session distinguisher needs only to be correct (or to generate output) when $b = b'$.

Diversion 3.3.7. Is a baseless sessional distinguisher U_2 easily constructible from a distinguisher U?

3.3.3.4 Boolean distinguishers

A **boolean** distinguisher is function U_1 such that

$$U_1(d, b, e, f) \in \{0, 1\} \tag{3.3.4}$$

with $U_1(d, b, e, f) = 1$ if and only if there exists a session $[a, b, c, d, e, f]$ of k.

A distinguisher can be constructed from boolean distinguisher by letting $U(d, b, e, \{f, f'\}) = f$ if $U_1(d, b, e, f) = 1$.

Diversion 3.3.8. The definitional downside of a boolean distinguisher is the difficulty extending it to the probabilistic setting, because it becomes vacuous. An attacker could always output 0 and be correct most of the time. The distinguisher taking input $\{f, f'\}$ has a more natural success when generalized to the probabilistic setting.

To be completed.

3.3.4 Detectors

To be completed.

Definition 3.3.2. *A role-i **detector** is a function $T_i : \mathrm{Ran}(k_i) \to \{0, 1\}$ such that $T_i(r) = 1$ if and only if $r \in \mathrm{Im}(k_i)$.*

Diversion 3.3.9. A **session** detector T_i is variant of the detector such that $T_i(r) = 1$ if and only if there exists a session $[a, b, c, d, e, f]$ of k such that k_i in the session has produced output r.

3.3.4.1 Impact of detectors

To be completed.

Diversion 3.3.10. Suppose that the agreed key f will be used directly as a key-stream, as in a Vernam cipher. Suppose also that the message space is much larger than the space of triples $[a, b, c]$.

In this setting, it makes sense to choose the range for f (i.e. the range for k_3 and k_4) to be all possible streams that could be added to the message. In this case, the number of actual f that could arise in sessions, which is at most the number of triples $[a, b, c]$, is much smaller than the range. For example, perhaps $|\,\mathrm{Im}(k_3)| \ll |\,\mathrm{Ran}(k_3)|$.

A detector T_3 models the ability to determine whether a given stream is actually real key-stream that can be generated by key agreement scheme.

The attacker who sees a ciphertext z and has a guess at the message can recover a guess at the key-stream by inverting the Vernam cipher, computing a candidate key-stream as $f' = m' \sim z$, via left subtraction as explained in previous notes. (When commutative addition is used, just take $m' \sim z = z - m'$.) A detector may then reveal if the message is incorrect, as follows. If the attacker knows that z was computed using $z = f + m$ for f an agreed key of session, but the detector says f' is not an agreed key of any session, then the attacker can reject the guessed message m'.

Diversion 3.3.11. If key agreement is used to generate Vernam cipher key-streams, a detector and a distinguisher can be used together to confirm message guesses.

If the sessional detector reports that f' is an agreed key for at least one session, then the attacker can try to use a distinguisher to see whether it matches the observed values $[d, b, e]$ from a specific session.

3.3.4.2 Detectors in general semigroups

To be completed.

Detectors are defined as range-dependent, depending very much on the ranges of the functions k_i, whereas other attackers, such as watchers and distinguishers, are only concerned with images of the functions k_i.

The main definitions of subscheme and equivalent schemes were given as range-independent notions. Therefore, detectors do not translate as cleanly as other attackers, when translating to subschemes or between equivalent schemes. In particular, in the reduction of key agreement to semigroups, the issues of the range have been mostly ignored.

Nonetheless, a range-aware reduction may be possible, by letting the semigroup have elements belonging to the ranges of the functions k_i. These ranges are subsets of the semigroups, but there might not be a clean way to describe a detector without calling out these subsets.

One artificial approach is to set the ranges to be the whole semigroup, and to also set the domains to the whole the semigroup. (Extending the domain should only the detector's task more difficult). In this case, detectors T_2 and T_3, can be defined $T_i : S \to \{0, 1\} : a_1 a_2 \dots a_i \mapsto 1$, and otherwise $T_i(a) = 0$. (Also, let $T_1 = T_2$ and $T_4 = T_3$.)

In other words, a detector tests if a semigroup element can be factored into 2 or 3 elements.

3.4 Esoteric security aims

The **basic** security aims of key agreement are resistance to a watcher, divulger or distinguisher.

Any other security aims of key agreement should be considered **esoteric**. This report does not focus on esoteric security. This chapter just briefly covers these definitions.

Diversion 3.4.1. Previously published correctness definitions for (implicitly-authenticated) key exchange might cover key agreement, at the level of interoperability (but perhaps with the addition of authentication).

If so, then one should expect that security aims previously published for (implicitly-authenticated) key exchange, such as the extended Canetti–Krawczyk model, might well rather thoroughly cover the set of attacks worth resisting.

As such, the esoteric security aims are likely then just a subset of the aims listed in previous publications.

3.4.1 Impact of esoteric security

Application systems sometimes apply (Diffie–Hellman) key agreement in such a way that the system's security subtly relies upon some esoteric security property of the key agreement. In other words, esoteric security of key agreement is sometimes necessary for an application's security.

In the "provable security" research area, sometimes an esoteric security property of key agreement is proved to imply a security property of the application system. Gaps between necessary and sufficient security properties arise. The esoteric security properties sufficient for proofs are not always necessary to avoid known attacks. Partly, this is due to the esoteric security properties involving hypothetical oracles. Resistance to such oracle-based security properties helps to prove things. Failure to resist to security properties does not always lead to attacks, because in some case, even the attacks do not have such hypothetical oracles.

In summary, applications can rely on esoteric security of key agreement, either to avoid potential attacks, or to help acquire assurance from proofs.

Diversion 3.4.2. Ideally, an application system should only rely upon the main security aim of key agreement, resistance to a watcher (seclusiveness). Failing that, the application should only rely on basic security aims.

In reality, it often happens that the only known practical solution relies on esoteric security of key agreement, forcing the study of esoteric security aims.

Diversion 3.4.3. Some key agreement schemes can be interpreted as applications of simpler key agreement schemes, by means of enhancing the simpler key agreement into a more sophisticated key agreement.

Key agreement schemes that enhance simple Diffie–Hellman key agreement include

- Hashed Diffie–Hellman key agreement,
- Menezes–Qu–Vanstone key agreement (and variants such as HMQV),

- EKE and SPEKE password-authenticated key agreement,
- Cash–Kiltz–Shoup twinned Diffie–Hellman.

One purpose of enhancing a simpler scheme is to boost security, often from basic security to esoteric security.

For hashed Diffie–Hellman, one security boost is to avoid a detector attack. Another boost is to combine with random-oracle approximations of the hash function for security proofs (against hash-generic adversaries).

Enhancing key agreement schemes can be sometimes be reduced to the enhancing semigroups: modifying a simpler semigroup into one with better security.

To be completed.

3.4.2 Corrupted-session attacks

Some attackers have an extra capability. They can **corrupt** a key agreement session $[a, b, c, d, e, f]$, weakening it in some way. Session corruption can make that task of a watcher, finding f, easier than it would otherwise be. (Session corruption might also help divulger, distinguishers, and detectors, too.)

Various forms of session corruption are discussed in this section.

3.4.2.1 Leaky attackers

A **leaky** attacker of a key agreement scheme gets to see some information about a or c. This is formalized below.

Diversion 3.4.4. Probabilistic watchers, defined previously, already account for this possible form of session corruption, which could arise the random variable p depended on the random variables a or c.

In this section, we formalize this leakage differently, making things more deterministic.

Let L be a function, called the **leakage** function. An L-**leaky** watcher is a function W_L such that $W_L(d, b, e, L(a, c, f)) = f$ for any session $[a, b, c, d, e, f]$.

Every watcher W makes a leaky watcher W_L by letting $W_L(d, b, e, l) = W(d, b, e)$. Conversely, every leaky watcher that ignores its fourth input (is constant on it), provides a watcher.

A leaky watcher is **lazy** if $L(a, c, f) \in \{a, c, f\}$. Lazy leaky watcher are easy to implement. If $l = L(a, c, f) = f$, then $W_L(d, b, e, l) = l$. If $L(a, c, f) = a$, then $W_L(d, b, e, l) = k_3(l, e)$, and so on.

The difficult question is whether there is a non-lazy leaky watcher more efficient than a watcher. (More precisely, by non-lazy, we mean that the leakage function should not immediately most information about a or c or f.)

Leaky divulgers, distinguishers and detectors can be defined similarly.

3.4.2.2 Bossy attackers

Let T be a function, called the **tampering** function. A T-**bossy** watcher is a function W_T such that $W_T(d, b, e) = f'$ for any session $[a, b, c, d, e, f]$ and a tampered session $[a', b', c', d', e', f']$, in which $[d', b', e'] = T(d, b, e)$q.

Every watcher W makes a bossy watcher W_T by letting $W_T(d, b, e) = W(d', b', e')$. A bossy watcher is **left controlling** if $T(d, b, e) = [k_1(a', b), b, e]$, for some a' (perhaps a function of $[d, b, e]$). A left controlling bossy watcher is easy: let $W_T(d, b, e) = k_3(a', e)$, because the tampered session is $[a', b, c, d', e, k_3(a', e)]$. A right controlling bossy watcher can be defined similarly, and is also found easily.

The interesting question is to find a bossy watcher more efficient than a watcher, for a non-controlling tampering function. (More precisely, a non-controlling bossy watcher lacks a large degree.)

Bossy divulgers, distinguishers and detectors can also be defined. Watchers that are both bossy and leaky can be defined too.

Diversion 3.4.5. The Menezes–Qu–Vanstone key agreement tries to resist bossy attackers, for the tampering function that permits total control the ephemeral Diffie–Hellman public keys, but leaves the static Diffie–Hellman public keys untouched.

Diversion 3.4.6. Signed Diffie–Hellman key agreement is used in some protocols, such as TLS 1.3.

One of the deliveries contains ephemeral Diffie–Hellman values, its digital signature, and a signature verification keys. The signature verification key is authenticated via a certificate. The certificate makes the signature verification key immune to tampering (in the context of key agreement). The ephemeral and Diffie–Hellman value are targets for tampering.

The hope of signed Diffie–Hellman is that a bossy watcher fails to tamper, because he cannot forge a valid signature. So, he must use the honest Diffie–Hellman values, and then cannot compute the Diffie–Hellman agreed key.

3.4.2.3 Invalid session attackers

It might be useful to broaden the definition of a bossy watcher by allowing the tampered session $[a', b', c', d', e', f']$ to not correspond to any valid session. Instead, just ask that $f' = k_3(a, e')$ or $f' = k_4(d', c)$.

In this case, we have an **invalid-session** bossy watcher.

To be completed.

3.4.3 Base-stealing

In some key agreement applications, the base value b in a session $[a, b, c, d, e, f]$ is a weak secret. Most typically, the value b is derived from a password pre-shared between Alice and Charlie.

Several password-authenticated key exchange schemes fall into this class.

To be completed.

3.4.3.1 Base-faker

A **base-faker** against k is a pair $[\kappa_2, \kappa_4]$ of functions such that

$$\kappa_4(k_1(a, b)) = k_3(a, \kappa_2(k_1(a, b)))$$

whenever $[a, b]$ is in the domain of k_1.

To be completed.

3.4.3.2 Base-buster

A **base-buster** against k is a function β such that

$$\beta(d, f, e) = b$$

whenever $[a, b, c, d, e, f]$ is a session of k.

To be completed.

Diversion 3.4.7. In associative key agreement, a base-buster is essentially a function $\beta : [ab, abc, bc] \mapsto b$.

Diversion 3.4.8. In some semigroups, the function define $\beta(d, f, e) = e \wedge_f d$ computes $[ab, abc, bc] \mapsto b$. In such semigroups, a watcher implies a base-buster.

For example, if the semigroup is a group (with an identity and inverses), then $e \wedge_f d = ef^{-1}d = (bc)(abc)^{-1}(ab) = bcc^{-1}b^{-1}a^{-1}ab = bb^{-1}b = b$. Furthermore, in a group, the converse holds, so watchers and base-busters are equivalent.

3.4.3.3 Double-base-tester (simultaneous DHP)

A **double-base-tester** of k is a function τ with the following properties. For all a, a', b, b', such that $d = k_1(a, b) = k_1(a', b')$, then

$$\tau(d, b, b') = [e, f, f'],$$

where

$$\begin{aligned} f &= k_3(a, e), \\ f' &= k_3(a', e). \end{aligned}$$

Suppose that Alice shares a password with Charlie. Adversary Pester knows that Alice has one of two possible passwords, p and p', from which Alice derived one of two bases b or b'. Pester sees Alice deliver d to Charlie. Pester uses a double-base-tester to generate an e. Pester sends e to Alice, as though from Charlie. Alice computes either f or f'. Pester can easily determines which Alice computes by observing her subsequent actions, or reactions. (Alice might encrypt or authenticate a message with one of f or f', or Alice might decrypt or verify a message with one of f or f', and react accordingly).

Diversion 3.4.9. Left double-base-testers can be defined similarly, as functions $[e, b, b'] \mapsto [d, f, f']$.

Diversion 3.4.10. Multiple base-tests can also be defined, replacing $\{b, b'\}$ by a larger set.

Diversion 3.4.11. In the context of Diffie–Hellman, a double-base-tester has sometimes been called the **simultaneous** Diffie–Hellman problem.

Diversion 3.4.12. Double-base testing in an associative key agreement scheme can be implemented using strict cross-multiplication §B.8.13.3.

To be completed.

3.4.4 Multi-session attacks

A key agreement scheme is likely to be used in more than one session, and perhaps by more than one user. Consequently, it makes sense to define multi-session attacks and multi-user attacks.

3.4.4.1 Independent sessions

Let s and t be positive integers. An $[s, t]$-watcher is a function $W_{s,t}$ such that

$$W_{s,t}([d_1, b_1, e_1], \ldots, [d_t, b_t, e_t]) = [[i_1, f_{i_1}], \ldots, [i_s, f_{i_s}]] \tag{3.4.1}$$

with positive integers $i_1 < \cdots < i_s$, whenever $[a_i, b_i, c_i, d_i, e_i, f_i]$ are sessions of k.

An $[s, t]$-watcher can be constructed by applying a watcher s times, by choosing any $i_1 < \cdots < i_s$ arbitrarily, and setting $f_{i_j} = W(d_{i_j}, b_{i_j}, e_{i_j})$.

If $s > 0$, then a watcher can be constructed by applying an $[s, t]$-watcher once, by letting $[d_i, b_i, e_i] = [d, b, e]$ for all i, running $W_{s,t}$ and letting $W(d, b, e) = f_{i_1}$.

Diversion 3.4.13. It is possible to define probabilistic $[s, t]$-watcher, similarly to how probabilistic watchers are defined. By default, one would define the success rate against t independent sessions.

A probabilistic $[s, t]$-watcher can be constructed from probabilistic watcher by the same construction as for the deterministic versions described above. If the probabilistic watcher has success rate σ, then the probabilistic watcher would have success rate σ^s.

A probabilistic watcher could be constructed from a probabilistic $[s, t]$-watcher by the same construction as for the deterministic versions described above. However, if $t > 1$ and the probabilistic watcher has success rate less than one, then the resulting probabilistic watcher might have success rate zero, because the $[s, t]$-watcher might fail when all its inputs are identical.

An alternative construction of a probabilistic watcher from a probabilistic $[s, t]$-watcher is follows. It makes $u = t/s$ calls to the $[s, t]$-watcher. In each, of the u calls to $W_{s,t}$, randomly select one of the t inputs to be the input $[d, b, e]$ to the probabilistic watcher. Choose the other $t - 1$ inputs by generating fresh sessions. Each call to the probabilistic $[s, t]$-watcher has probability s/t of solving the input $[d, b, e]$. After $u = t/s$ calls, the success rate becomes non-negligible.

Diversion 3.4.14. Diffie–Hellman key agreement has the property, usually sometimes random self-reducibility, that a single call to probabilistic $[1, t]$-watcher efficiently implies a watcher.

If $s < u$, an $[s, t]$-watcher can be constructed by applying a $[u, t]$-watcher once by taking any s-element sub-array of the u-element array output of the $[u, t]$-watcher.

Diversion 3.4.15. The derivation of the definition $[s, t]$-watcher from the definition watcher can be applied to almost any problem, solving s of t independent instances of a given a problem.

In particular, this suggests it should be possible to defined an $[s, t]$-divulger, an $[s, t]$-distinguisher, and so on.

3.4.4.2 Re-usage attackers

Sometimes Alice will use the same a in more than one session, or even more than one scheme.

Diversion 3.4.16. Alice may do this for a few reasons.

A weak reason is to lower cost. For example, Alice may compute $d = k_1(a, b)$ just once, and then re-use d in multiple sessions, and even in other schemes. (Alice must still compute $f = k_4(a, e)$ separately for each session.)

A stronger reason is if Alice uses $d = k_1(a, b)$ as a trusted public key, whether it is certificated, or pre-distributed.

A natural question is whether multiple uses of the secret a makes any of the previous attacks easier.

A **re-usage** $[s, t]$-watcher has the same definition as an $[s, t]$-watcher except that its input domain are restricted, so that the underlying sequence of sessions satisfy:

$$a_1 = a_2 = \cdots = a_t. \tag{3.4.2}$$

Any $[s, t]$-watcher gives a re-usage $[s, t]$-watcher, by restricting the domain.

Diversion 3.4.17. As before, a probabilistic re-usage $[s, t]$-watcher could be defined. In this case, there seems not to be a generic conversion between either way between re-usage $[s, t]$-watcher and non-re-usage $[s, t]$-watchers, except perhaps through the probabilistic watcher, by viewing $[s, t]$-watcher and re-usage $[s, t]$-watcher as two different ways to generalize a watcher (by varying a different set of inputs).

To be completed.

To be completed.

3.4.5 Oracle attacks (generalized Gap-DHP, etc.)

Sometimes a function f has an input g which is itself a function. Call such f an **oracle function**, and call the function input g an **input oracle** of f. An **oracle algorithm** implements an oracle function f. The oracle algorithm treating its input oracles as externally defined subroutine, that can be called outside of the oracle algorithm.

Typically, when determining the cost of an oracle algorithm, counts each call to the input oracle as a single unit of efficient cost. It might be case that in the real world, the only known algorithm to implement the input oracle is infeasible. Nevertheless, the oracle algorithm could be efficient or feasible, if it uses a sufficiently low amount computational resources.

Diversion 3.4.18. An oracle algorithm might have variable cost, meaning its as runtime or memory usage, depend on the outputs of its input oracles. If no feasible algorithm for the input oracles is known, then it might be too difficult to determine the cost of an variable-cost oracle algorithm. So, caution is warranted to detect this case.

An **oracle attack** is an attack using an oracle algorithm. An oracle attack only works given some input oracles.

3.4.5.1 User-oracle attacks

A fairly realistic class of oracle attacks is where the user implements the input oracle for the attacker.

Some general notation helps indicate concisely what input oracles that users provide to certain user-oracle attacks.

Let f be a function with n inputs. Let $X_1, \ldots, X_n$ be sets. Let $f|_{S_1,\ldots,S_n}$ indicate the function f but with domain restricted to $S_1 \times \cdots \times S_n$, so it maps $[x_1, \ldots, x_n] \mapsto f(x_1, \ldots, x_n)$ provided $x_i \in S_i$. As abbreviations, write $S_i = T_i'$ for the complement set, meaning $x_i \notin T_i$, and write $S_i = s$ for the singleton set $\{s\}$.

Fix a session $[a, b, c, d, e, f]$ of key agreement scheme k. A **user-oracle watcher** is a function Ω such that:

$$\Omega(d, b, e, k_1|_{a,\{b,e\}'}, k_2|_{\{b,d\}',c}, k_3|_{a,\{b,e\}'}, k_4|_{\{b,d\}',c}) = f.$$

Let $I \subset \{1, 2, 3, 4\}$. An I-**only** user-oracle watcher ignores its oracle inputs for k_j with $j \notin I$. In other words, it only uses oracles for k_i with $i \in I$.

Diversion 3.4.19. Diffie–Hellman is vulnerable to a I-only user-oracle watcher if $I \neq \{\}$. Hashed Diffie–Hellman is vulnerable to a user-oracle watcher if $1 \in I$ or if $2 \in I$.

User-oracle distinguishers and divulgers can be defined similarly.

Diversion 3.4.20. A user-oracle divulger in the case of Diffie–Hellman is known as the q-strong Diffie–Hellman problem.

Arguably the user-oracle watcher is not very realistic, because the oracle access to k_3 and k_4, representing users directly revealing the agreed keys f and g. Most applications of key agreement do not immediately reveal the agreed keys, so it is unrealistic to have an oracle that directly reveals the key.

On the other hand, it is quite realistic to detect whether any particular guess of the agreed key has been used.

To this end, another general notation is introduced. If F is a n-input function, let $F?$ be the Boolean-valued function taking $n+1$ inputs, such that $F?(x_1, \ldots, x_n, x_{n+1})$ is true only if $F(x_1, \ldots, x_n) = x_{n+1}$. Call this function a **test** for F. In the restricted-domain function $F|_{S_1,\ldots,S_n}$, write $S_i = *$ to indicate no restrictions on the input x_i.

A **test-user-oracle watcher** is a function

$$\omega(d, b, e, k_1|_{a,\{b,e\}'}, k_2|_{\{b,d\}',c}, (k_3|_{a,*})?, (k_4|_{*,c})?) = f.$$

3.4.5.2 Irreducibility-oracle attacks

Sometimes, proofs with strong security conclusions arise by conjecturing the lack of a reduction between hard two problems.

For example, there is no known efficient reduction that uses a decision Diffie–Hellman problem solver to solve the Diffie–Hellman problem. In some cases, the best general algorithms to solve the problems are equally efficient, yet there is no proof of equivalence, despite much effort.

Diversion 3.4.21. Sometimes, these problems are called **gap** problems, indicating that there is a gap in the known reductions. But this name is misleading if there no known gap in difficulty of the problems.

The extensive effort spent trying to find a reduction, suggests that no reduction exists, a condition that could be called **irreducibility**. This is evidence is that an oracle to solve one problem does not help solve another.

Some applications of key agreement have been proven to meet realistic security aims if the key agreement resist irreducibility-oracle attackers.

To be completed.

3.4.6 Agreeable and disagreeable attack definitions

To be completed.

Sometimes, definition of attack types can be described using oracle functions. For example, the success condition of a watcher, can be described into terms of the equality of two oracle functions:

$$\begin{aligned}\alpha &: [a, b, c, W, k_1, k_2, k_3, k_4] \mapsto W(k_1(a, b), b, k_2(b, c)),\\ \upsilon &: [a, b, c, W, k_1, k_2, k_3, k_4] \mapsto k_3(a, k_2(b, c)).\end{aligned}$$

Furthermore, the success condition does nothing more than move and copy the non-oracle inputs. This makes the definition of a watcher applicable to any key agreement scheme. Informally, we call such a definition an **agreeable** attack definition, because the definition makes sense for any key agreement, because treats the function k_i as oracles, and any the data values involved as opaque values to copied between various input oracles.

Informally, a **disagreeable** attack definition is one that is not agreeable. The attack might defined for a some k, but it is undefined for other choices of k. Disagreeable attack definitions can be said to be **over-specific** to the key agreement scheme k over which they are defined.

As an example, we describe a modified watcher definition, made disagreeable and over-specific to Diffie–Hellman. The original watcher W computes $W(d, b, e) = W(b^a, b, b^c) = b^{ac}$. The modified watcher W' computes $W'(b^a, b, b^c) = b^{ac+1} + 1$. For Diffie–Hellman, the watcher and modified watcher are essentially equivalent, since it only involves a multiplication by b and an addition of 1. But the original watcher is not a disagreeable definition, because its success condition can be defined using the functions k_i as oracles. The modified watcher involves the extra operations (multiplication by b and addition of 1), which might not be meaningful or applicable to key other key agreement schemes, and in particular to k_i given as oracles.

Diversion 3.4.22. Just to be clear, the modified watcher given above against Diffie–Hellman is actually a watcher against a modified Diffie–Hellman key agreement scheme.

Diversion 3.4.23. Intuitively speaking, suppose that some application of a key agreement scheme k relies on resistance against a disagreeably defined attacker against k.

Then the application is not really using k for key agreement.

3.4.7 Semigroup-defined attacks

To be completed.

Abstractly, one can to formulate nearly arbitrary problems in a semigroup, or a ring.

Under the reductionist viewpoint of the previous chapter, one can ponder whether each such problem corresponds to a security property for key agreement.

This question might illustrate why security definitions can be hard to get right: the open-endedness of the security means the difficult quite arbitrary problems might be relevant to security.

3.4.7.1 Systems of a equations

A watcher W (or a wedge) against a multiplicative key agreement scheme can re-formulated as a function for partially solving the following system the equations in a semigroup:

$$\begin{aligned} d &= ab, \\ e &= bc, \\ f &= abc, \end{aligned}$$

where d, b, e are the known variables, and the a, c, f are the unknowns. A watcher tries to find one of the unknowns, namely f.

Such systems can be generalized. Let $h_1, \ldots, h_m$ be knowns. Let $x_1, \ldots, x_n$ be unknowns. Consider a system of equations:

$$L_i(h_1, \ldots, h_m; x_1, \ldots, x_n) = R_i(h_1, \ldots, h_m; x_1, \ldots, x_n),$$

for $i \in \{1, \ldots, q\}$. Each of L_i and R_i are known functions taking a semigroup product of some sequence of their inputs.

In the case of watchers, let $[h_1, h_2, h_3] = [d, b, e]$ and $[x_1, x_2, x_3] = [f, a, c]$, and let the six functions be:

$$\begin{aligned} L_1(h_1, h_2, h_3; x_1, x_2, x_3) &= h_1 \\ R_1(h_1, h_2, h_3; x_1, x_2, x_3) &= x_2 h_2 \\ L_2(h_1, h_2, h_3; x_1, x_2, x_3) &= h_3 \\ R_2(h_1, h_2, h_3; x_1, x_2, x_3) &= h_2 x_3 \\ L_3(h_1, h_2, h_3; x_1, x_2, x_3) &= x_1 \\ R_3(h_1, h_2, h_3; x_1, x_2, x_3) &= x_2 h_2 x_3 \end{aligned}$$

Given such a system, some natural problems arise.

- Is there a solution $x_1, \ldots, x_n$?
- Find x_1 (for some solution $x_1, \ldots, x_n$)?

Because there are so many possible system of equations, it would seem that some should have very little security impact on an application of key agreement. Perhaps the following type of systems are the ones relevant to key agreement.

Consider t sessions $[a_i, b_i, c_i, d_i, e_i, f_i]$ of k. The set of variables in the system is $\{a_i, b_i, c_i, d_i, e_i, f_i : 1 \leq i \leq t\}$. Some of the variables are known to the attacker. Some of the variables are unknown to the attacker. In multiplicative key agreement, the system of equations include the session equations:

$$\begin{aligned} d_i &= a_i b_i, \\ e_i &= b_i c_i, \\ f_i &= a_i b_i c_i \end{aligned}$$

The system may also include some extra equations. Typically, the extra equations are just equalities between some of the variables, usually between some of the a_i, or between some of the c_i, representing the situation where Alice or Charlie re-usages their secrets.

The attacker then tries to find one or more of the unknown variables. The attack could instead just try to determine whether the system is solvable.

Diversion 3.4.24. The attacker might know partial information about some of the unknowns. For example, a distinguisher (defined previously) knows a set of the two unknowns containing the target unknown, leaving only single of bit information about the target unknown for the attacker to determine.

The attacker might also be able to choose some of the variables. Some of the (partially) known variables might get revealed after some of the choices, allowing for subsequent choices of variables to be adaptive.

Diversion 3.4.25. In the probabilistic setting, success rate generally needs to be calibrated, in a way meaningful to the application, to compare against a baseline attacker that ignores (all) the input, and simply guesses the output.

Chapter 4

Wedges: some generic strategies

This chapter lists some strategies to compute a wedge (Definition B.9.1), which, recall, is a ternary operator with notation $[d, b, e] \mapsto d \wedge_b e$ and the property that $(ab) \wedge_b (bc) = abc$ for all a, b, c.

Seclusiveness of key agreement (its basic security) is resistance to watchers, which amounts to the difficulty of computing a wedge operator (see §3.1.4.3).

4.1 Wedge by division

Dividing (Definition B.8.1) is a way to compute a wedge (Definition B.9.1):

Lemma 4.1.1. *If / is a divider, then*

$$d \wedge_b e = (d/b)e, \tag{4.1.1}$$

defines a wedge operator.

Proof. Let / be a divider (meaning $((ab)/b)b = ab$ for all a, b).

If $[d,b,e] = [ab,b,bc]$ for any a,b,c, and $\wedge$ is defined as in (4.1.1), then

$$\begin{aligned}(ab) \wedge_b (bc) &= d \wedge_b e \\ &= (d/b)e \\ &= ((ab)/b)(bc) \\ &= ((ab/b)b)c \\ &= (ab)c \\ &= abc,\end{aligned} \tag{4.1.2}$$

where the second equality is from (4.1.1), the first and third from the definition of d and e, the fourth and sixth from associativity, and the fifth from $/$ being a divider.

This holds for all $[a,b,c]$, so $\wedge : [ab,b,bc] \mapsto abc$ meets the definition of a wedge operator. □

Diversion 4.1.1. A wedge is at most one multiplication more costly than division.

Division on the left (§B.8.5) can also be used:

Lemma 4.1.2. *If $\backslash$ is a left divider, then*

$$d \wedge_b e = d(b\backslash e), \tag{4.1.3}$$

defines a wedge operator.

Proof. Mirror image of proof of Lemma 4.1.1. □

Diversion 4.1.2. The wedge operator symbol $\wedge$ is motivated by the merger of the formulas

$$(d/b)e = d \wedge_b e = d(b\backslash e). \tag{4.1.4}$$

So, $\wedge_b$ is a concatenation of 3 symbols: $/$ and $\backslash$ and b. A directly concatenated notation $d/b\backslash e$ is too easily misconstrued as two divisions.

Wedge by division essentially works even if the divider $/$ has outputs outside the given semiring S. Suppose that T is an **extension** or **supersemigroup** of S, meaning that $S \leq T$ (S is a subsemigroup of T). Suppose that $/$ is a divider for semiring T, which might have $d/b \notin S$ even if $d, b \in S$.

Fix any function $f : S^3 \to S$ and define an **extended** wedge by division operator:

$$d \wedge_b e = \begin{cases} (d/b)e & \text{if } (d/b)e \in S, \\ f(d,b,e) & \text{if } (d/b)e \notin S. \end{cases} \tag{4.1.5}$$

To see that this defines a wedge in S, consider any $a, b, c \in S$, put $d = ab$ and $e = bc$. Because $/$ is a divider in T, we have $((ab)/b)b = ab$. Multiplying this on the right by c shows that $(d/b)e = ((ab)/b)(bc) = ((ab)/b)b)c = abc \in S$. Therefore, $d \wedge_b e = (d/b)e$ by the definition of $\wedge$ above. Therefore, $(ab) \wedge_b (bc) = abc$ for all $a, b, c \in S$.

For some key agreement schemes, an incomplete wedge §B.9.3 suffices to build a watcher. In this case, a faulty divider (§B.8.6) can be enough to construct an incomplete wedge.

Lemma 4.1.3. *If $/$ is a faulty divider with divider-set T, then*

$$d \wedge_b e = (d/b)e, \tag{4.1.6}$$

defines an incomplete wedge operator correct for seeds $[a, b, c] \in T \times S$.

Proof. Restrict a, b, c in the proof of Lemma 4.1.1 to the case that $[a, b] \in T$. □

Diversion 4.1.3. A semigroup can be said to be **wedge-divisive** if the least costly algorithm for the wedge operation is not significantly more costly than a division and a multiplication. (Wedge-divisive semigroups exist: for example, those in which division can be achieved using a wedge, §5.4.)

A semigroup can be said to be **wedge-severe** if the opposite holds, in other words, if there is an algorithm to solve the wedge problem that is significantly more efficient than computing a division (by b) and a multiplication (the algorithm of Lemma 4.1.1). Wedge-severe semigroups exist, some with wedge algorithms listed in the rest of this chapter.

Diversion 4.1.4. Side-division (§B.8.6.4) also suffices to compute a wedge. Conversely, any wedge operator of the form $d \wedge_b e = f(d, b)e$ defines a side-division algorithm $d/b = f(d, b)$.

Diversion 4.1.5. In non-associative magma, a left wedge is a function with $[ab, b, bc] \mapsto (ab)c$, but the construction in (4.1.1) does not necessarily result in a wedge. In other words, the proof of Lemma 4.1.1 relies essentially on the associativity of the semigroup.

4.2 Wedge by inversion

Fix b in a semigroup S. We say that q is **wedge inverse** of b if there is a wedge operator with

$$d \wedge_b e = dqe, \tag{4.2.1}$$

for all $d, e \in S$. Equivalently, q is a **wedge inverse** of b if

$$abqbc = abc \tag{4.2.2}$$

for all $a, c \in S$. One type of wedge algorithm is to apply a wedge inversion algorithm (an algorithm to find a wedge inverse q of b) and then applying two multiplications.

When used as a watcher, the work in finding q can be re-used across all key agreement sessions that use the same base value b. Some applications of key agreement might re-use the non-secret b many times.

The wedge by inversion algorithm is not entirely distinct from the previous (§4.1) wedge by division algorithm. The two algorithms can overlap when division by inversion (§5.2) is used.

Diversion 4.2.1. In some semigroups, a wedge inverse tends also to be a **divisional inverse** (defined in §5.2), thereby implying the existence an effectively equivalent division algorithm.

Diversion 4.2.2. The algorithms also overlap since, if q is a wedge inverse of b, then $d/b = dq$ defines a side-division algorithm (§B.8.6.4).

The following lemma limits idempotent-free semigroups from having any wedge inverses.

Lemma 4.2.1. *If any element of a semigroup has a wedge inverse, then the semigroup has an idempotent element.*

Proof. Suppose that q is the wedge inverse of b. Let $e = qbqb$, then

$$\begin{aligned} ee &= (qbqb)(qbqb) \\ &= (qbqbq)(bqb) \\ &= (qbq)(bqb) \\ &= (qbqbq)(b) \\ &= (qbq)(b) \\ &= (qb)(qb) \\ &= qbqb \\ &= e, \end{aligned}$$

using associativity and the wedge inverse rule $abqbc = abc$ with $a = c = q$. □

Diversion 4.2.3. Two examples of idempotent-free semigroups include: $\{1, 2, 3, \ldots\}^+$, the positive integers under addition; and $\{2, -2, 4, -4, \ldots\}^\times$, the even nonzero integers under multiplication.

Diversion 4.2.4. Although b might not have a wedge inverse in S, there sometimes exists a semigroup extension T of S in which b does have an inverse. For two examples: S could be the positive integers under addition, and T the set of all integer; S be integers larger than 10 under multiplication, and T be the set of positive rational numbers under multiplication.

But, the mere existence of such T does not imply an effective algorithm for wedge by inversion. For example, the practicality of the multiplication in T might depend on practical division in S.

4.3 Wedge by constant

A **wedge by constant** has $d \wedge_b e = u \wedge_v w$ for all $d, b, e, u, v, w \in S$.

Let $z = u \wedge_v w$ for some $u, v, w \in S$. Then $(ab) \wedge_b (bc) = u \wedge_v w = z$ and $(ab) \wedge_b (bc) = abc$, so $abc = z$ for all $a, b, c \in S$.

Elements z is absorbing. To see this, compute $yz = yabc = (ya)bc = a'bc = z$, writing $a' = ya$. Similarly, $zy = z$. By convention, we can write $z = 0$. So $abc = 0$ for all $a, b, c \in S$, if S has a wedge by constant. In other words, S is 3-nilpotent. Conversely, if S is 3-nilpotent ($abc = 0$ for all $a, b, c \in S$), then S has a wedge by constant: $d \wedge_b e = 0$.

A multiplicative semigroup S is n-nilpotent for positive integer n if the subset $S^n = \{s_1 s_2 \ldots s_n : s_i \in S\}$ has only one element.

Diversion 4.3.1. If S is n-nilpotent, then the sole element of S^n is an absorbing element (§B.6.2), so, under standard conventions, we (re-)write the sole element of S^n as 0.

(To see that $0 \in S^n$ is absorbing, observe that $S^i \subseteq S^{i+1}$ for all i, since $s_1 \ldots s_i s_{i+1} \in S^{i+1}$ is a product of i factor, as in $(s_1) \ldots (s_{i-1})(s_i s_{i+1})$, the last factoring being $s_i s_{i+1}$. Now, for any $s \in S$, we have $0s, s0 \in S^{n+1}$, since $0 \in S^n$. But $S^{n+1} \subseteq S^n = \{0\}$, so $0s, s0 \in \{0\}$ and $0s = s0 = 0$, so 0's absorbing, as required.)

Diversion 4.3.2. A t-nilpotent semigroup is $(t+1)$-nilpotent. For example, a 2-nilpotent semigroup is a 3-nilpotent.

Diversion 4.3.3. Recall that a **nilpotent** element s of a ring is an element such that $s^n = 0$ for some positive integer n. The ring definition and semigroup definition are related. Ring element s is nilpotent if and only if semigroup $\{0, s, s^2, s^3, \ldots\}$ is nilpotent. (This semigroup is a subsemigroup of the ring's multiplicative semigroup.)

To minimize confusion, **nilpotent** applied to an element means ring nilpotent, and applied to a semigroup means semigroup nilpotent.

A potential confusion is that a non-nilpotent ring element s can generate a nilpotent semigroup $\langle s \rangle = \{s, s^2, s^3, \dots\}$. What can happen is that the 0 of the ring does not belong to $\langle s \rangle$, even though $\langle s \rangle$ has its own absorbing element. In semigroup theory, such s generating a nilpotent semigroup are sometimes called **aperiodic**. This report prefers to call such elements **period** 1 elements (§B.4.3).

Alternative names include **idemperiodic**, to emphasize the idempotents have period 1, or **nilperiodic**, to emphasize the nilpotent (ring) elements have period 1. But these two terms are bit too awkward, and are not needed often enough in this report to adopt them.

Diversion 4.3.4. In group theory, the terminology **nilpotent** has a more indirect meaning, and is not closely related to the meaning in semigroup theory. Unfortunately, these two meanings of nilpotent clash. A group is never nilpotent as semigroup, unless it is trivial. A non-trivial nilpotent group is not nilpotent as semigroup (even though it is a semigroup). A non-trivial nilpotent semigroup is not nilpotent as a group (and not even a group at all).

Diversion 4.3.5. If division is difficult in 3-nilpotent semigroup S, then S is a wedge-severe semigroup (Diversion 4.1.3).

Difficult division in 3-nilpotent semigroups is plausible, because a 3-nilpotent semigroup can be formed from any binary operation, including cryptographic function like HMAC, as described in §**??**.

Division in these semigroups seems difficult in general.

Diversion 4.3.6. If S is 3-nilpotent, then the wedge operator $d \wedge_b e = 0$ may be considered to be an instance of the wedge by side-division algorithm (§B.8.6.4), for the side-division algorithm $d/b = 0$.

4.4 Wedge by multiplication

A **wedge by multiplication** has $d \wedge_b e = de$. Some semigroups have wedge by multiplication, which usually means the key agreement scheme is insecure (unless the multiplication of d and e is somehow infeasible).

A semigroup S is **self-distributive** if $abc = abac = acbc$, for all $a, b, c \in S$. In other words, a self-distributive semigroups is a semiring where the addition and multiplication operations are identical, $a + b = ab$ for all a, b, because then $abc = a(b + c) = ab + ac = abac$ and $abc = (a + b)c = ac + bc = acbc$. To make the notation more familiar, when clear from context, write $a + b = ab$ in a self-distributive semigroup.

Self-distributive semigroups have a wedge by multiplication. To see this, compute as follows:

$$\begin{aligned}(ab) \wedge_b (bc) &= (ab)(bc)\\ &= (a(bb))c\\ &= (a(b+b))c\\ &= (ab+ab)c\\ &= ababc\\ &= a(ba+bc)\\ &= ab(a+c)\\ &= abac\\ &= ab+ac\\ &= a(b+c)\\ &= abc,\end{aligned} \tag{4.4.1}$$

for any $a, b, c \in S$, as required for a wedge operator.

Diversion 4.4.1. Wedge by multiplication can sometimes be considered wedge by division (§4.1) combined with division by identity (§5.3). So, for example, wedge by multiplication works in polarized semigroups (in addition to self-distributive semigroups).

4.5 Wedge by deletion and concatenation

Fix a function $h : L \times R \to M$. The h-**free** semigroup S_h is defined below. For one-way h, division has difficult instances, but the wedge is easy.

Diversion 4.5.1. Cryptographers can base h to their favorite hash function, for example, $h(a, b) = \text{SHA-256}(a\|b)$, where $a\|b$ means representing a and b as octet strings and then concatenating them.

The elements of S_h are non-empty sequences with entries in the disjoint union $L \uplus R \uplus M$ (using distinguishable copies of each of L and R and M). All such sequences are allowed except those with an entry in L followed by an entry in L. (In other words, it is $L \times R$-free.)

To multiply sequences $a = [a_1, \ldots, a_i]$ and $b = [b_1, \ldots, b_j]$, compute

$$ab = \begin{cases} [a_1, \ldots, a_i, b_1, \ldots, b_j] & \text{if } [a_i, b_1] \notin L \times R, \\ [a_1, \ldots, h(a_i, b_1), \ldots, b_j] & \text{if } [a_i, b_1] \in L \times R. \end{cases}$$

In other words, multiplication is concatenation of sequences, unless that would create adjacent entries in $L \times R$ (which are not allowed). The adjacent entries in $L \times R$ are replaced by an entry in M, by application of the function h.

Equivalently, S_h is the free semigroup generated by the set $L \uplus M \uplus R$ modulo the congruence generated by the relations $[a][b] = [h(a,b)]$ for $[a,b] \in L \times R$.

Division in S_h has some difficult instances. Consider $a = [a_1]$ with $a_1 \in L$ and $b = [b_1]$ with $b_1 \in R$. Then $d = ab = [h(a_1, b_1)]$. Computing d/b amounts to finding a pre-image of the function $a \mapsto h(a,b)$, which should be infeasible if h is a one-way function.

Computing a wedge in S_h is easy. Compare the final entries the left entries of e to the entries of b. In relevant instances, they will match except possibly for the single rightmost entry of b. Do the same comparison for d, except one compares the right entries of d, and only the leftmost entry of b might fail to match.

Let d' be d with the matched entries deleted. Let e' be e with the matched entries deleted. Let b' be b with unmatched end-entries (first or last) deleted (so, if b_1 does not match the corresponding entry of d, then delete it). Concatenate d' and b' and e' as the value for $d \wedge_b e$.

4.6 Wedge by monomorphism

The wedge problem in one semigroup can be solved by using an injective morphism into another semigroup.

Lemma 4.6.1. *If $f : S \to T$ is a semigroup morphism, and $g : T \to S$ is a function such that $g(f(s)) = s$ for all $s \in S$, then*

$$d \wedge_b e = g\left(f(d) \wedge_{f(b)} f(e)\right) \tag{4.6.1}$$

defines a wedge operator in S, given a wedge operator in T.

Proof. Suppose $d = ab$ and $e = bc$ for $a, b, c \in S$. It suffices to prove that

$d \wedge_b e = abc$ using (4.6.1). Now

$$\begin{aligned} d \wedge_b e &= g\left(f(d) \wedge_{f(b)} f(e)\right) \\ &= g\left(f(ab) \wedge_{f(b)} f(bc)\right) \\ &= g\left((f(a)f(b)) \wedge_{f(b)} (f(b)f(c))\right) \\ &= g\left(f(a)f(b)f(c)\right) \\ &= g\left(f(abc)\right) \\ &= abc \end{aligned}$$

as desired. □

An important special case is wedge in a subsemigroup S of T. In this case, the injective morphism is $f : S \to T : s \mapsto s$. The function $g : T \to s$ can be defined $g(t) = t$ if $t \in S$ and $g(t) = h(t)$ if $t \notin S$, where $t : (T - S) \to S$ is any function from $T - S = \{t \in T : t \notin S\}$ to S. In other words, the wedge $\wedge$ in subsemigroup $S \leq T$ can be taken as the same as that in T, except any outputs in T can be moved arbitrarily into S.

Diversion 4.6.1. Lemma 4.6.1 provides a feasible wedge operator for S if:

- the morphism f is feasible,
- the wedge operator in T is feasible,
- the function g is feasible.

Diversion 4.6.2. The existence of g means that f is an injective function, so that $|T| \geq |S|$. The reason that a finding wedge in T helps to find a wedge in S cannot be because T is smaller in size. Rather, T must have some specialized structure making the wedge operator easier. The morphism f and function g can be viewed as methods to extend this special structure of T into S.

Diversion 4.6.3. Because f is an injective morphism, we may regard S as a subsemigroup of T. More precisely, T has a subsemigroup $f(S)$ that is isomorphic to T.

The function g need not be a morphism, but the restriction of g to domain $f(S)$, as in $g : f(S) \to S$, provides one direction of the isomorphism between S and $f(S)$.

Diversion 4.6.4. We also did a transfer (a reduction) of a wedge from one semigroup to another, in the §4.7, but that transfer did not directly use a injective morphism.

4.6.1 Aside: wedge as two multiplications

Consider the subset bSb of S. This subset is closed under the binary wedge operator $\wedge_b$. Furthermore, $\wedge_b$ is uniquely defined on bSb, and it is associative on bSb. Therefore, so bSb forms a semigroup with operation, $\wedge_b$. Write this as $T = (bSb)^{\wedge_b}$.

Let $U = S^{\vee_b}$ be the following semigroup with operation $\vee_b$ defined

$$a \vee_b c = abc.$$

So, the underlying set of U is the same as the underlying set of S. The binary operation U (its multiplication) costs two multiplications in S.

Diversion 4.6.5. The semigroup U is isomorphic to a Rees matrix semigroup $\text{Ree}_m(S)$ (see §4.7 and §C.8.6). Put $m = [b]$, meaning a 1×1 matrix with sole entry valued b. The isomorphism is $h : a \mapsto [1, a, 1]$, satisfying $h(a \vee_b c) = h(a)h(c)$.

The function $f : S^{\vee_b} \to (bSb)^{\wedge_b} : a \mapsto bab$ is a semigroup morphism because $f(a \vee_b c) = f(abc) = babcb = bab \wedge_b bcb = f(a) \wedge_b f(c)$.

So, the wedge operator $\wedge_b$ on bSb is an image of the binary operator $\vee_b$, which costs two multiplications in S. Using this viewpoint to implement $\wedge_b$ in $bSb^{\wedge_b}$ would require inverting f. Inverting f, seems to require at least two divisions. This would only partially implements $\wedge_b$ in S, because it is limited to arguments in the subset bSb, whereas $\wedge_b$ has relevant inputs outside of bSb.

Diversion 4.6.6. This observation shows that if finding an efficient isomorphism between any two isomorphic semigroups is easy, then at least some of the inputs to a wedge operator would cost only two multiplications.

This partially suggests that the semigroup isomorphism is difficult in general. Otherwise, wedges would seem to be easy in the general.

To be completed.

4.7 Wedge by Rees index deletion

The **Rees matrix semigroup** $\text{Ree}_m(S)$, detailed in §C.8.6, for a base semigroup S and rectangular matrix m with entries $m(i, j) \in S$, has elements and multiplication defined by:

$$[h, a, i][j, b, k] = [h, am(i, j)b, k],$$

where $a, b \in S$ and h, i, j, k are indices to the matrix m (with h, j being left indices, and j, k right indices).

Lemma 4.7.1. *In* $\mathrm{Ree}_m(S)$*, the wedge operator can be computed as*

$$[g, d, h] \wedge_{[i,b,j]} [k, e, l] = [g, d \wedge_b e, l], \tag{4.7.1}$$

where $\wedge$ *on the right is a wedge operator in* S*.*

Proof. Write

$$\begin{aligned} A &= [g, a, n], \\ B &= [i, b, j], \\ C &= [o, c, l], \\ D &= [g, d, h], \\ E &= [k, e, l]. \end{aligned}$$

Assume $D = AB$ and $E = BC$. Expanding both sides of $D = AB$ as triples, gives

$$\begin{aligned} g &= g, \\ d &= am(n, i)b, \\ h &= j. \end{aligned}$$

Similarly, for $E = BC$,

$$\begin{aligned} k &= i, \\ e &= bm(j, o)c, \\ l &= l. \end{aligned}$$

So,

$$\begin{aligned} D \wedge_B E &= ABC \\ &= [g, am(n, i)bm(j, o)c, l] \\ &= [g, (am(n, i)b) \wedge_b (bm(j, o)c), l] \\ &= [g, d \wedge_b e, l] \end{aligned}$$

as desired. □

Diversion 4.7.1. The inputs to the definition (4.7.1) of a wedge operator in $\mathrm{Ree}_m(S)$ that have $(i, j) \neq (k, g)$ are not valid instances of the wedge problem in $\mathrm{Ree}_m(S)$. A wedge operator is permitted to produce arbitrary outputs for such invalid inputs, because they are irrelevant to the wedge operator.

Diversion 4.7.2. In general, the wedge operator (4.7.1) seems faster than using division in $\mathrm{Ree}_m(S)$, because the best division algorithm (that I could find) seems to requires a search for a factor of d and e.

Consequently, it seems that, in general, a Rees matrix semigroup is wedge-severe (Diversion 4.1.3).

Diversion 4.7.3. To be completed.

A theorem of Rees [Wikipedia] says that a **completely simple** semigroup S is isomorphic to a Rees matrix semigroup $\mathrm{Ree}_m(G)$ where G is a group. If both directions of the isomorphism are efficient (are they?), then the wedge problem in S is reducible to the wedge problem in a group G, which should be equivalent to inversion in the group.

4.8 Wedge by coordinate

Suppose that $S = S_1 \times S_2 \times \cdots \times S_n$ is a **Cartesian product** of finitely semigroups $S_1, S_2, \ldots, S_n$, meaning that the elements of S are sequences $[s_1, \ldots, s_n]$ with coordinate $s_i \in S_i$, and multiplication has multiplies the coordinates, so

$$[a_1, \ldots, a_n][b_1, \ldots, b_n] = [a_1 b_1, \ldots, a_n b_n].$$

Then **wedge by coordinate** means applying of n wedges, one wedge in each coordinate:

$$[d_1, \ldots, d_n] \wedge_{[b_1,\ldots,b_n]} [e_1, \ldots, e_n] = [d_1 \wedge_{b_1} e_1, \ldots, d_n \wedge_{b_n} e_n]. \tag{4.8.1}$$

Diversion 4.8.1. Wedge by coordinate can also be defined similarly for infinitary Cartesian products, although in that case, wedge by coordinate should be not considered as a practical strategy, since there will be infinitely many coordinates, and thus infinitely many wedges to evaluate. But in this multiplication by coordinate should not be considered an algorithm.

Diversion 4.8.2. Subsemigroups of Cartesian products sometimes arise. Then a combination of wedge by monomorphism (the subsemigroup embedding) combined with wedge by coordinate might be applicable.

An example of subsemigroup of an (infinitary) Cartesian product, consider the positive integers $\mathbb{P} = \{1, 2, 3, \ldots\}$ under multiplication. For each prime q, let $\mathbb{P}_q = \{1, q, q^2, \ldots\}$ be the subsemigroup of $\mathbb{P}$ consisting of powers of q. Consider $\mathbb{P}$ as a subsemigroup of the infinitary Cartesian product $\mathbb{P}_2 \times \mathbb{P}_3 \times \mathbb{P}_5 \times \ldots$, by way of prime-power factorization. For example, 20 maps to $[4, 1, 5, 1, 1, 1, \ldots]$.

Sometimes such subsemigroup permit efficient multiplication by coordinate or wedge by coordinate, despite having an infeasibly large number of coordinates (infinite). This could be because only finitely many coordinates are non-trivial.

In the example of $\mathbb{P}$ above, multiplication by coordinate and wedge by coordinate are not very practical as algorithms, when $\mathbb{P}$ is represented by the standard decimal representation, because it requires prime-power factorization, which is slow.

4.9 Wedge by trial discrimination

For any semigroup, define a **discrimination** function $\delta : S^4 \to \{0, 1\}$ by the rule:

$$\Delta(w, x, y, z) = \begin{cases} 1 & \text{if } [w, x, y, z] = [ab, b, bc, abc] \text{ for some } a, b, c \\ 0 & \text{if } [w, x, y, z] \neq [ab, b, bc, abc] \text{ for all } a, b, c \end{cases} \tag{4.9.1}$$

A wedge by **trial discrimination** algorithm computes $d \wedge_b e$ is specified by a list $[z_1, z_2, \ldots]$. The algorithm loops over i, starting from $i = 1$ and incrementing i by 1 at each iteration. In each iteration, the loop computes $\Delta(d, b, e, z_i)$. The loop ends as soon as $\Delta(d, b, e, z_i) = 1$. After the loop ends, the algorithm returns z_i as the value of $d \wedge_b e$.

Diversion 4.9.1. Given a wedge operator $\wedge$, let function $\Delta_\wedge$ be defined by $\Delta_\wedge(w, x, y, z) = [(w \wedge_x y) = z]$, taking value 1 only if $w \wedge_x y = z$, and otherwise taking value 0.

The function $\Delta_\wedge$ often fails to be a discrimination function, for example if there does not exist $[a, b]$ such that $[w, x] = [ab, b]$. However, if $\Delta_\wedge$ is multiplied by two boolean divisibility functions (testing if $[w, x] = [ab, b]$ and if $[x, y] = [b, bc]$, then the resulting product function is a discrimination function.

In some cases, the fastest known way to compute the discrimination function requires computing the wedge $w \wedge_x y$ and comparing this to z. In these cases, computing a wedge by trial discrimination does not help. In other cases, the discrimination function can be computed more quickly then a wedge.

A wedge by trial discrimination is an instance of the more general trial search method. Probabilistic forms of trial search are discussed in Appendix D. Probabilistic forms of trial search can be applied to trial discrimination.

Chapter 5

Division: some generic strategies

This chapter lists some generic strategies for division, meaning to compute an operator / such that

$$((ab)/b)b = ab$$

for all a, b (in a multiplicative semigroup, or in a ring, or in two subsets of a semigroup or ring, or for given random variables defined over a semigroup or ring). (See §B.8 for more detailed definitions of division, and some general theory of division.)

Division impacts key agreement because it can be used to compute wedges, as shown in §4.1, which can be used to construct watchers that defeat seclusiveness (the main security aim of key agreement). Conversely, in some cases, division is the fastest known way to compute wedges, and thus is the fastest known way to defeat the main security aims of key agreement. Division also be used to build a divulger against a key agreement scheme, which has slightly more security impact than a watcher, as shown in §3.2.6.

The description of division strategies here deliberately belabors the obvious. When applying division strategies to sophisticated real-world semigroups, the obvious might become obscured. When considering a real-world key agreement scheme, it might help to take a step back, and walk through a checklist of the obvious, since it would be unfortunate to overlook an easy attack.

The division strategies list here are mostly **generic** in the sense that they do not focus on a specific multiplication. Instead, relatively simple algorithms are listed that seems to divide for a variety forms of multiplication. A thorough security analysis of a specific key agreement scheme can consider such generic attacks, but should also include more specific efforts to attack the scheme under consideration.

5.1 Division by trial multiplication

Division by **trial multiplication**, as discussed in this section, is arguably the simplest and most general division strategy. Trial multiplication is an instance of a more general strategy: **trial search**, which find pre-images of an arbitrary function (multiplication by b, in this instance).

This section focuses on deterministic trial multiplication. Probabilistic versions of trial search are discussed in Chapter D.

Diversion 5.1.1. Division by trial multiplication is a minimum baseline by which to measure how difficult division is in a semigroup.

5.1.1 Trial multiplication in finite semigroups

In this section, the semigroup S is assumed to be finite. Division by trial multiplication is parameterized by a trial function.

Definition 5.1.1. *A **trial search function** is a surjective function*

$$t : \{1, \dots, n\} \to S.$$

*The number n is the **length** of trial function.*

Of course, $n \geq |S|$ for any trial function.

Diversion 5.1.2. Whenever clear from context, n denotes the length of a trial function. In the context of a trial function t, when the notation variable n indicates the size of the domain of t,

We next define notational divider / that will later be proven to be an actual divider.

Definition 5.1.2. *The **trial multiplication** divider / derived from trial search function t computes d/b as follows:*

1. *Let $i = 1$.*
2. *Let $e = t(i)$.*

3. *If $i < n$ and $eb \neq d$, then*
   (a) *Increase i by* 1,
   (b) *Go back to Step 2.*
4. *Return e.*

The trial multiplication divider $/$ is not just a notational divider, it is actual divider, as proved next.

Lemma 5.1.1. *If $/$ is a trial multiplication divider, then it is a divider meaning $((ab)/b)b = ab$ for all $a, b \in S$.*

Proof. Consider any $a, b \in S$, and let $d = ab$. Suppose that the procedure to compute d/b ends with e such that $eb = d$. Then

$$((ab)/b)b = (d/b)b = eb = d = ab.$$

Otherwise, the procedure must have ended with $eb \neq d$, which will be shown to be impossible, due to contradiction.

The procedure must have ended with $i = n$, because if $i < n$, condition $eb \neq d$, which have cause another iteration.

Reaching $i = n$ means that $eb \neq d$ for all trials, so that

$$t(i)b \neq d$$

for all $i \in \{1, \ldots, n\}$. Equivalently, $d \notin \{t(i)b : 1 \leq i \leq n\}$.

But t is surjective, so this means that $d \notin \{sb : s \in S\}$. But that is a contradiction, because $d = ab \in \{sb : s \in S\}$, by setting $s = a$. □

Diversion 5.1.3. In practice, a trial function should be implemented by a practical (feasible) algorithm.

The basic cost of implementing trial multiplication divider $/$ is up to n multiplications in S and n applications of the function t. A more detailed cost analysis, using probabilistic trial functions, is given in Chapter D.

Diversion 5.1.4. In some semigroups, multiplication itself has a cost varying with the elements being multiplied. In such semigroups, the cost of trial multiplication may be complicated to assess. In fact, the cost assessment might be meaningless for assessing the practical security of associative key agreement. In such cases, probabilistic trial search might lead to more meaningful assessments.

5.1.2 Eternal trials in countable semigroups

The natural extension of trial multiplication to countably infinite semigroups is defective in two respects.

- It is not generally guaranteed to produce a divider, or even a complete function, because if there is no suitable d/b, infinitely many trials will be tested, and the procedure will not return any value for d/b whatsoever.
- The cost, in the deterministic setting, is infinite, or more precisely, taking an unbounded number of trials and multiplications.

Nonetheless, remedies to the defects, discussed further below, make the natural extension worth defining, as done next.

Definition 5.1.3. *An* ***eternal trial search function*** *is a surjective function*

$$t : \{1, 2, 3, \dots\} \to S.$$

Definition 5.1.4. *The* ***eternal trial multiplication*** *divider / derived from eternal trial search function t computes d/b as follows:*

1. *Let $i = 1$.*
2. *Let $e = t(i)$.*
3. *If $eb \neq d$, then*
   (a) *Increase i by 1,*
   (b) *Go back to Step 2.*
4. *Return e.*

Again, eternal trial multiplication as described above, is not always fully defined, sometimes never returning an answer. This defect has some remedies such as the following:

- In the probabilistic setting, the expected (average) cost can be finite (see Chapter D for details).
- Given a test for divisibility, never-ending searches can be avoided (see §5.1.3), ensuring that the / always terminates with a correct answer, thus providing a divider.

- If a pre-divider exists (see §B.8.9), then the eternal trial multiplication will always terminate, as formally proved below.

Lemma 5.1.2. *If S has a pre-divider (see §B.8.9), then eternal trial multiplication defines a divider.*

Proof. As in the proof for the finite case, if the eternal trial multiplication computation of d/b returns e with $i < \infty$, the $eb = d$, so if $(d/b)b = d$, (and if $d = ab$, then $((ab)/b)b = ab$.

Otherwise, the procedure to compute d/b iterates forever. This means that $t(i)b \neq d$ for all $i \geq 1$. But t is surjective, this means $d \neq eb$ for all $e \in S$.

Since a pre-divider exists, say $\div$, then $(d \div b)b = b$, by definition of pre-divider. Since $d \div b \in S$, we get a contradiction. □

5.1.3 Divisible eternal trial multiplication

To be completed.

Definition 5.1.5. *A* ***divisibility function*** *is a function $| : S^2 \to \{true, false\}$: $[d, b] \mapsto b|d$ such that $b|d$ is true if and only there exists a such that $d = ab$.*

Eternal trial multiplication can be modified to make it into a divider as follows.

Definition 5.1.6. *The* ***divisible eternal trial multiplication*** *divider / derived from eternal trial search function t compute d/b as follows:*

```
1. If $b|d$, then
    (a) Let $i = 1$.
    (b) Let $e = t(i)$.
    (c) If $eb \neq d$, then
        i. Increase $i$ by $1$,
        ii. Go back to Step 1b.
    (d) Return $e$.
2. Return $d$.
```

Divisible eternal trial multiplication always terminates, and always produces an answer d/b. The resulting operator d/b is a divider.

Diversion 5.1.5. To do: find an example of a semigroup with an efficient divisibility function and trial multiplication being the best possible division strategy.

Diversion 5.1.6. To do: sufficiently fast divisibility functions sometimes speed up the average runtime of finite trial multiplication, because it replaces n trial and multiplication when the inputs $[d, b]$ have $b \nmid d$. (Average run-time is properly defined only in the probabilistic setting.)

Diversion 5.1.7. A divisibility function can be defined using a divider, since $b|d$ if and only if $(d/b)b$. But divisibility would not help to compute d/b, so divisibility is only likely to help if it is faster than division.

5.1.4 Aside: transfinite search in uncountable semigroups

More abstractly, trial multiplication may be viewed as the finite (or countable) case of a theoretical transfinite search.

Definition 5.1.7. *A* ***transfinite trial search function*** *is a surjective function*

$$t : W \to S,$$

where the domain W is well-ordered by a relation $\leq$.

Definition 5.1.8. *The* ***transfinite trial multiplication*** *divider / derived from transfinite trial function t compute d/b as follows:*

$$d/b = \begin{cases} t(m) & \text{if } m = \min\{v : t(v)b = d\}, \\ d & \text{if } \{\} = \{v : t(v)b = d\}. \end{cases}$$

Diversion 5.1.8. Well-ordering of W is used in the definition in order to formally generalize the features of trial search. In particular, the order of the trials can affect the outcome of the result.

Diversion 5.1.9. Division by transfinite trial multiplication proves that a divider always exists, in uncountable semigroups.

Diversion 5.1.10. In some cases. there might exist divider / that are not obtained by transfinite trial multiplication, at least as we have described it.

Suppose $a_i b_j = d$ for $i, j \in \{1, 2\}$. Suppose that $d/b_i = a_i$. Then this divider is not obtained by trial multiplication. Suppose that $a_1 < a_2$ in the well-ordering of S. Any trial multiplication would have $d/b_2 \neq a_2$, because $d/b_2 \leq a_1 < a_2$.

One could define a different well-orderings for different inputs to the divider, to get around this. But then it should no longer be considered trial multiplication, because the dependency of the well-ordering on the inputs is something more powerful than trying all multiplications. In other words, such input-dependency would make the division more advanced.

In the extreme case, take any divider at all, /, for each input d, b define a well-ordering that makes d/b minimal. Then only one trial multiplication is used, and actually the trial is unnecessary because the trial is define to be successful.

Diversion 5.1.11. The axiom of choice, instead of well-ordering, can be used, but it less constructive (at least according to some naive set theory).

For each $b, d \in S$, define a set $d//b = \{a : ab = d\}$, and a set $d///b$ such that $d///b = d//b$ if $d//b$ is not empty, and otherwise $d///b = S$. All the sets $d///b$ are not empty. By the axiom of choice, there exists a function / such that $d/b \in d///b$ for all $b, d \in S$.

To see that operator / is a divider, consider $(ab)/b$. By definition, $(ab)/b \in (ab)///b$. But $a \in (ab)//b$, so $(ab)//b$ is not empty, meaning $(ab)///b = (ab)//b$, by definition. Therefore $(ab)/b \in (ab)//b$, which means that $((ab)/b)b = (ab)$.

This has merely shown the existence of a divider /. In some case, there may be many different /, and the axiom of choice does not help which of these choices. By comparison, each well-ordering defines a specific /, and can thus be viewed as more constructive (ignoring the issues of transfinite).

5.1.5 Optimality of trial multiplication

To be revised.

Diversion 5.1.12. For two examples where trial multiplication might be the fastest division algorithm, consider the semigroups $S^{\langle\mathrm{AES}\rangle}$ and $S^{\langle\mathrm{HMAC}\rangle}$ formed as the semigroup closures of AES and HMAC, as described in §**??**.

The semigroup $S^{\langle\mathrm{AES}\rangle}$, right division has input consisting of a message block and a cipher block, with the goal being to compute the key (or an equivalent key). The security of AES therefore relies on difficult right division in $S^{\langle\mathrm{AES}\rangle}$. Indeed, the fastest right division algorithm seems to be trial multiplication. Left division is $S^{\langle\mathrm{AES}\rangle}$ is easy, because it corresponds to recovering the message given the key and cipher block.

The security of HMAC also relies on difficult right division, because it corresponds to the task of finding the HMAC key from the message and its HMAC tag. Again, the best right division algorithm seems to be trial multiplication. Semigroup $S^{\langle\mathrm{HMAC}\rangle}$ seems to have difficult left division, being the problem of recovering the message from the HMAC key and the HMAC tag. Generally, the advertised security HMAC does not rely on difficult left division in $S^{\langle\mathrm{HMAC}\rangle}$, because the standard security definition for a message authentication code (such as HMAC) does not guard the secrecy of the message (that task is deferred to encryption).

Despite the difficult division, both left and right, in $S^{\langle\mathrm{HMAC}\rangle}$, associative key agreement from $S^{\langle\mathrm{HMAC}\rangle}$ is not secure at all, as explained in §**??**.

Diversion 5.1.13. A semigroup in which trial multiplication is the fastest division algorithm (for both left and right division) should perhaps be called **division-triable**.

Diversion 5.1.14. In some semigroups, trial multiplication is not the best division algorithm: significantly faster division algorithms exist.

Indeed, the example $S^{\langle \mathrm{AES} \rangle}$ has right division in which trial multiplication seems to be the best algorithm, but has a left division algorithm (decryption) that has essentially the same cost as multiplication (encryption).

Diversion 5.1.15. Perhaps the most secure possible semigroup would be one where:

- division is the best wedge algorithm (which we called wedge-divisive in the chapter on wedge algorithms),
- trial multiplication is the best division algorithm (which we called division-triable).

Unfortunately, this report finds quite the opposite: all division-triable semigroups contemplated in this report are wedge-severe (the wedge problem is easy).

If this pattern is due to some fundamental reason, then other division and wedge algorithms must be considered.

5.2 Division by inversion

Sometimes b has an inverse q (see §B.7 for various definitions of inverse). In this case, division by b using inversion is possible.

Lemma 5.2.1. *There exists a divider / such that, for each b with a right divisional inverse (Definition B.7.5),*

$$d/b = dq \tag{5.2.1}$$

for a right divisional inverse q of b.

Proof. Define d/b as follows.

- If b has a right divisional inverse, then let $d/b = dq$ for some arbitrary choice q of right divisional inverse of b.
- If b has no right divisional inverse and $\{e : eb = d\}$ is non-empty, then let d/b as any arbitrary choice c of element in $\{e : eb = d\}$.
- Otherwise, let d/b be any arbitrary choice z of element in S.

To see that / is divider, for any $a, b \in S$, compute

$$((ab)/b)b = \begin{cases} abqb & \text{if } b \text{ has a right divisional inverse} \\ cb & \text{if } b \text{ has no right divisional inverse} \end{cases}$$

where: q is the chosen inverse of b, and c is the chosen element of $\{e : eb = ab\}$. The set $\{e : eb = ab\}$ is not empty, because it contains a, so, the third case of $(ab)/b$ being defined as an arbitrary choice z in S does not arise when computing $(ab)/b$.

In the first case above, we get $abqb = ab$ because q is a right divisional inverse of b. In the second case above, we get $cb = ab$ because $c \in \{e : eb = ab\}$.

So $((ab)/b)b = ab$ for all a, b, meaning that / is a divider. □

Diversion 5.2.1. Conversely, q is a right divisional inverse of b if there exists a divider / with $d/b = dq$ for all d.

More generally, if $Q : S \to S$ is a function such that $Q(b)$ is a right divisional inverse of b for all $b \in S$. Then $d/b = dQ(b)$ is a divider.

An algorithm that computes an inverse q of b in order to compute division $d/b = dq$ by a single multiplication can be called a **division by inversion** algorithm.

Strategies to compute inverses vary widely. Some are discussed in Chapter 6.

5.2.1 Non-invertible elements

In many important semigroups, some elements are not invertible. For example, in the semigroup $\mathbb{P}$ of positive integers under multiplication, the only element with any kind of inverse is 1. In other words, division by inversion does not always work.[1]

Diversion 5.2.2. Sometimes, a given semigroup can be extended to allow inverses, for example the the positive integers $\mathbb{P}$ under multiplication can be extended to positive rationals $\mathbb{Q}_{>0}$ under multiplication.

Existence of such extensions does not lead immediately to division algorithms. The extension's multiplication can effectively require division in the the base semigroup.

The example of positive rationals as an extension of the positive integers shows this. To compute $91/13$ in the positive integers, we could compute $91 \times \frac{1}{13}$, where $\frac{1}{13}$ is a positive

[1] Hence the rest of this chapter.

rational and the inverse of 13. Multiplication in positive rationals takes two multiplications of positive integers, multiplying numerators and denominator, and also a reduction step to put the fraction in lowest terms. (A unique form, a monogram, is needed for key agreement.) The reduction step implies an ability to divide positive integers, since $\frac{91}{13}$ reduces to $\frac{7}{1} = 7$.

5.3 Division by identity

Under special conditions, letting $d/b = d$ or $d/b = b$, leads to successful division. Division by an idempotent (an element b with $bb = b$, see §B.6) is an example.

Lemma 5.3.1. *There exists a divider such that*

$$d/b = d \tag{5.3.1}$$

whenever b is idempotent.

Proof. There exists a divider $\div$. Let $d/b = d$ if b is idempotent, and let $d/b = d \div b$ otherwise. We need to prove that $((ab)/b)b = ab$ for all a, b. If b is not idempotent, then $((ab)/b)b = ((ab) \div b)b = ab$, because $\div$ is a divider. If b is idempotent, then $((ab)/b)b = (ab)b = a(bb) = ab$, because $bb = b$. □

Diversion 5.3.1. An idempotent is its own inverse. Division by inversion would compute $d/b = db$.

If the semigroup S is a monoid (it is unital with multiplicative identity 1), then 1 is also a right divisional inverse of any idempotent element. So, in a monoid, division by inversion can take the form $d/b = d1 = d$ from Lemma 5.3.1.

When S is not monoid, it might be case the computing $d/b = d$ is considerably faster than computing $d/b = db$.

Diversion 5.3.2. Idempotent semigroups with costly multiplication may exist.

For an artificial example of an idempotent semigroup, take boolean sets under intersection and apply an arbitrary, but very slow-to-implement bijection to obtain an isomorphic idempotent semigroup whose multiplication is very slow.

To make this less abstract, we can use cryptography. Fix an AES key k and an integer $N \approx 2^{50}$. Let the semigroup $S = \{0, 1\}^{128}$ have underlying set consisting of all bit strings of length 128. Define multiplications as follows: $st = \mathrm{AES}_k^{-N}(\mathrm{AES}_k^N(s) \& \mathrm{AES}_k^N(t))$, where & indicates bit-wise multiplication (AND), and $\mathrm{AES}_k^{\pm N}$ indicates N applications of AES encryption with the fixed key k (and $-N$ applications of decryption if N is negative).

This semigroup seems to have very slow multiplication, because a single multiplication seems to require three applications of very slow bijections. But division is very fast because all its elements are idempotents, so we can compute $d/b = d$, essentially for free.

This example shows that division can be much faster than multiplication. This artificial semigroup is a proof-of-concept for the possibility that a semigroup can have division more efficient than multiplication. In other words, it serves a reminder, or a warning flag, that even if a multiplication is slow, division might be faster.

More specifically, this argument argues strongly against a strategy of devising a semigroup with decelerated (deliberately slow) multiplication. This example shows that decelerating multiplication does not necessarily decelerate division.

In other words, obtaining slow division requires more fundamental than slow multiplication.

A special class of semigroups that has been considered in semigroup theory is the class of polarized semigroups.

Definition 5.3.1. *A semigroup is* ***polarized*** *if $abc = ac$ for all a, b, c.*

Division-by-identity works well in a polarized semigroup.

Lemma 5.3.2. *In a polarized semigroup, the operator / defined by*

$$d/b = d,$$

for all d, b, is a divider.

Proof. Put $d = ab$ and compute $((ab)/b)b = (d/b)b = db$, because $d/b = d$. But $db = abb$, because $d = ab$. Put $c = b$ in the equation $abc = ac$ to get $abb = abc = ab$. Therefore $((ab)/b)b = ab$. □

A divider defined $d/b = b$ for all d, b can also be considered. If $d/b = b$ defines a divider in a semigroup S, then the semigroup has the following limited structure: S can be partitioned into disjoint non-empty subsets $S_1, \dots, S_n$, with elements $e_i \in S_i$ such that $ab = e_i$ whenever $b \in S_i$.

Furthermore, if $d/b = b$ defines a divider /, then $d \div b = d$ also defines a divider $\div$. To see this, compute $((ab) \div b)b = (ab)b = (((ab)b)/b)b = bb = ((ab)/b)b = ab$. In other words, a divider defined $d/b = b$, is redundant, since $d \div b = d$ can be used instead.

Overall, the conditions under which division-by-identity works seem quite easy to detect, and easy to avoid.

5.4 Division by wedge

We saw already that the wedge problem is solvable by division. Sometimes, the converse applies. In these cases, the the division problem has even greater importance, since the main attacks against key agreement all rely on the difficulty of the division problem.

Lemma 5.4.1. *There exists a divider such that*

$$d/b = d \wedge_{bb} b, \tag{5.4.1}$$

whenever b is sidle-invertible (meaning there exists q with $bbq = b = qbb$, and see §B.7.5).

Proof. As done in other proofs of this report, define $/$ by modifying any existing divider $\div$. In this case, change its output to that in (5.4.1) whenever the second input b is sidle-invertible. Otherwise, let $d/b = d \div b$. To show that $/$ is divider, means to show that $((ab)/b)b = ab$. When b is not sidle-invertible, this follows from $\div$ being a divider. All that remains is to prove it when b is sidle-invertible.

Let q be the sidle inverse of b, meaning $bbq = b = qbb$. Then

$$\begin{aligned}
((ab)/b)b &= ((ab) \wedge_{bb} b)b \\
&= ((a(qbb)) \wedge_{bb} (bbq))b \\
&= (((aq)(bb)) \wedge_{bb} ((bb)q)b \\
&= ((aq)(bb)q)b \\
&= aq(bbq)b \\
&= aq(b)b \\
&= a(qbb) \\
&= ab
\end{aligned}$$

as required. □

Diversion 5.4.1. Lemma 6.2.1 shows that if b has a sidle inverse, then it has a middle inverse (which is easily seen to be right divisional inverse). Therefore, division by inversion in applicable in the case.

Nonetheless, division by wedge is relevant for at least two reasons. First, computing a wedge could be easier than computing an inverse. Second, division by wedge can serve as reductionist argument, even if not a practical attack, excluding the possibility that the wedge operation is somehow easier than division.

Combining division-by-wedge algorithm above with the wedge-by-division algorithm results in the following wedge-by-wedge algorithm:

$$d \wedge_b e = (d/b)e = (d \wedge_{bb} b)e. \tag{5.4.2}$$

The middle input to the wedge on the right is bb, not b. This wedge-by-wedge algorithm might be useful if computing a wedge with middle input bb is easier than computing a wedge with middle input b. This wedge-by-wedge algorithm might work even if b is not sidle-invertibility.

Diversion 5.4.2. Perhaps a binary wedge operator $\wedge_{bb}$ is infeasible, even though $\wedge_b$ is feasible. We still might want to divide by b (in attacking key agreement, an attacker who divides might learn Alice's long-term secret a, not just the session key f). An alternative strategy is to divide using the formula

$$d/b = (d \wedge_b \sqrt{b}) \wedge_b \sqrt{b}.$$

This formula requires a method to compute an element $\sqrt{b}$ such that $b = \sqrt{b}\sqrt{b}$. It probably also requires some invertibility condition for $\sqrt{b}$, such as sidle-invertibility.

More obviously, division can sometimes be achieved by $d/b = d \wedge_b 1$, which works if b has a right unital inverse, meaning a q such that $bq = 1$. To see this, compute $((ab)/b) = ((ab) \wedge_b 1)b = ((ab) \wedge_b bq) = abq = a1 = a$, which shows that / is a (cancelling) post-divider, and therefore a divider (multiply both sides by b on the right).

5.4.1 Generalized den Boer reductions

Suppose that S is a commutative semigroup, that $\wedge_b$ is given as an oracle, that we want to divide d/b where $d = ab$ with $a = g^x$ for a known g and unknown positive integer x, and g has the property $g = g^{n+1}$ for a positive integer n. Suppose that n is a product of small number $n = n_1 \dots n_t$, such that we know each n_i.

Here is a method to compute d/b in this situation. The idea generalizes a method due to den Boer for solving discrete logarithms using a Diffie–Hellman oracle.

The plan is to compute $x_i = x \mod n_i$ for each i, then solve for x using the Chinese remainder theorem. Then output $a = g^x$.

Diversion 5.4.3. Finding x via the the x_i is essentially the Pohlig–Hellman algorithm for solving discrete logarithms. The wrinkle in the den Boer algorithm is that we do not deal directly with g^x but rather representations in another semigroup.

To compute x_i, we need to compute $g^{x(n/n_i)}b$, and compare it to $g^{y(n/n_i)}$ for $y \in \{0, \dots, n_i - 1\}$. The matching y is x_i.

The procedure to compute $g^{x(n/n_i)}b$ is most easily explained with a little extra notation. Write $[a]$ for ab. Let $[S]$ be the semigroup $\{[a] : a \in S\}$

with multiplication law $[a][c] = [a] \wedge_b [c]$. (More formally, we should write $[(Sb)^{\wedge_b}]$ to more clearly indicate the inner semigroup has the binary wedge $\wedge_b$ as its output, and the enclosed semigroup translates this to multiplicative notation.)

Expanding the definition, $[a][c] = [a] \wedge_b [c] = ab \wedge_b cb = ab \wedge_b bc = abc = acb = [ac]$, by commutativity of S. Therefore, the map $[\cdot] : S \to [S]$ is a semigroup morphism.

Since $[\cdot]$ is a semigroup morphism, we have $g^{x(n/n_i)}b = [g^{x(n/n_i)}] = [g^x]^{n/n_i}$. But we also know the base $[g^x] = g^x b = ab = d$, and the exponent n/n_i, so we can compute the power $[g^x]^{n/n_i}$ using the square-and-multiply algorithm in $[S]$ (again, where multiplication of S is the wedge operation $\wedge_b$ of S).

Diversion 5.4.4. Actually, this algorithm requires merely that b commute with g. In particular, it actually does not require that the whole semigroup S is commutative. But the subsemigroup generated by b and g needs to be commutative.

Diversion 5.4.5. An improvement on the den Boer algorithm, due to Maurer–Wolf, uses some extra algebraic structure on the semigroup S, such that to the set $[S]$ can be equipped with an extra operation $+$, making it a ring.

With this ring structure on on $[S]$, the Maurer–Wolf idea is to form a random elliptic curve over $[S]$, and then try Pohlig–Hellman.

The den Boer method does not work if any of the n_i are too large. The Maurer–Wolf method might work in these cases, because heuristic conjectures suggest that a random elliptic curve would have an effectively random size near to n, and with an non-negligible probability of the curve size being a product of small numbers, enabling Pohlig–Hellman.

The extra algebraic structure is available in the Diffie–Hellman setting. The operation $+$ on $[S]$ in that case is multiplication mod p in the Diffie–Hellman mod p. More generally, as in in elliptic curve Diffie–Hellman, the operation $+$ on $[S]$ is the multiplication on the group G whose exponentiation operation is used to define multiplication of S.

5.5 Division by logarithm

A **discrete logarithm** (§B.3.6 for details) of d to the base b, is a positive integer such that $d^e = b$. If $e \geq 2$, then

$$d/b = b^{e-1} \tag{5.5.1}$$

provides a division algorithm, which we call **division by logarithm**. More precisely, division by logarithm means computing d/b by first computing a discrete logarithm e of d to the base b, if such a discrete logarithm exists, and then computing b^{e-1}.

Diversion 5.5.1. The exceptional case of $e = 1$, which means computing b/b, can be considered division by logarithm in some cases. If $1 \in S$, then we can set $b/b = 1$, which can be considered division by logarithm by the convention that $d/b = b^{e-1} = b^0 = 1$. (Often 1 is given, and easy to find, but there might exist semigroups in which 1 is not easy to find, in which division by logarithm needs some extra work.)

But a more precise meaning of b^0 is defined in §B.7.4: with $b^0 = bb^{-1}$ for the unique b^{-1} is a co-mutual inverse. In this case, the subsemigroup $\langle b, b^{-1}\rangle$ is a group, and b^0 is its multiplicative identity, even if it is not the multiplicative for the whole of S. Finding b^0, like finding 1, might take extra work, beyond what should be division by logarithm.

The main cost of division by logarithm is usually finding a discrete logarithm e. The powering computation b^{e-1} is usually less costly, being done with square-and-multiply, with at most $2\log(e)$ multiplications in S.

If there is known to be a discrete logarithm $e \leq E$ for some known integer E, then e can be found in a small multiple of $2\sqrt{E}$ multiplications in S. Such algorithm are well-known in cryptography: Shanks' baby-step-giant-step and Pollard ρ.

Diversion 5.5.2. A simple generalization is **division by subtracting logarithms**: if $d = g^e$ and $b = g^f$ for positive integers e and f with $e \geq f$, then let $d/b = g^{e-f}$.

Diversion 5.5.3. Division by subtracting logarithms can sometimes be extended to handle the case that $e < f$. If $g^n = 1$ for some n, then let $d/b = g^{(e-f) \bmod n}$.

5.6 Division by isomorphism

If two semigroups S and T are isomorphic, then **division by isomorphism** in S means to map the inputs in S to their images in T, then divide the images in T and then map the output in T back to S.

The cost of division by isomorphism is one division in T plus three isomorphism evaluations.

If the isomorphisms between S and T are efficient, then division S and T is essentially equivalent.

5.6.1 Division by monomorphism

A slight generalization of division by isomorphism is **division by monomorphism**. Recall that a monomorphism is an injective morphism from a given semigroup S into another T, and that isomorphisms are precisely the surjective monomorphisms.

Lemma 5.6.1. *If*

- *function $f : S \to T$ is a semigroup morphism,*
- *function $g : T \to S$ is a function such that $g(f(s)) = s$ for all $s \in S$,*
- *operator $\div$ is a divider in T, and*
- *image $f(S)$ is closed under the $\div$ operator in T,*

then the operator $/$ in S defined as

$$d/b = g\left(f(d) \div f(b)\right) \tag{5.6.1}$$

is a divider in S.

Proof. For any $a, b \in S$, we have $f(ab), f(b) \in f(S)$, so $f(ab) \div f(b) \in f(S)$, since $f(S)$ is closed under $\div$ by hypothesis. So, there exists $e \in S$ with $f(ab) \div f(b) = f(e)$. Calculate,

$$\begin{aligned}
((ab)/b)b &= g\left(f(ab) \div f(b)\right) b \\
&= g(f(e))b \\
&= eb \\
&= g(f(eb)) \\
&= g\left(f(e)f(b)\right) \\
&= g\left((f(ab) \div f(b))f(b)\right) \\
&= g\left(((f(a)f(b)) \div f(b))f(b)\right) \\
&= g\left(f(a)f(b)\right) \\
&= g\left(f(ab)\right) \\
&= ab,
\end{aligned}$$

so $((ab)/b)b = ab$, for all $a, b \in S$. □

For $/$ in (5.6.1) to be an algorithm for division in S, algorithms for the functions f and g are needed, as well an algorithm for the divider $\div$ in T.

Diversion 5.6.1. A given divider $\div$ in T might not have the property that $f(S)$ is closed in $\div$. In other words, not every division algorithm for T can used to divide by monomorphism.

Interestingly, such a closure property was not required for reduction of wedges via a morphism (in §4.6).

Diversion 5.6.2. Division by monomorphism can be considered as division by isomorphism because semigroup S and $f(S)$ are actually isomorphic. The morphisms f and g above are isomorphisms between them (when the range of f is restricted to $f(S)$ and, more importantly, the domain of g is restricted to $f(S)$). With this view, the main difference is that $f(S)$ lives inside a larger semigroup T, which may naturally occur commonly to spell out the full setting.

Diversion 5.6.3. It is sometimes possible to define **partial division by partial morphism**, by weakening the condition of f so that it is only a partial morphism: multiplicative only for the inputs of interest.

Diversion 5.6.4. Division by subtracting logarithm can be considered as a case of partial division by partial morphism, the partial morphism being the discrete logarithm function.

5.6.2 Difficult isomorphisms

Evaluating a morphism $f : S \to T$ is not always easy, even if f is known to exist and is easy to define (by an implicit equation, for example). In this case f can be said to be **difficult morphism**. Similarly, if f is easy, but its inverse g is difficult, then we call also call f a **difficult isomorphism**. In either case, division by isomorphism is difficult, given only a difficult isomorphism.

A important example is Diffie–Hellman key agreement. Consider the semigroup S from the proof of Lemma 2.18.1, which has an example multiplication in Table 2.2 with $p = 5$. There exists an isomorphism from this semigroup to another semigroup where all division is efficient. The isomorphism requires evaluating a discrete logarithm modulo p. For large p, it is a difficult isomorphism. More detail is given below.

Let g be a primitive element modulo p, which recall, means that for all $y \not\equiv 0 \bmod p$, there exists a unique integer x with $0 \leq x < p - 1$ and $g^x \equiv y \bmod p$. In other words, g generates the multiplicative subsemigroup of nonzero integers modulo p, and x is the discrete logarithm of y. Write $x = \log_g(y)$, when clear the context.

Next, we define a semigroup T with same underlying set $\{[0], \ldots, [p-2], [p-1], \ldots, [2p-3]\}$ and multiplication:

$$[a][b] = \begin{cases} [ab \bmod (p-1)] & \text{if } a < p-1, \quad b < p-1, \\ [(ab \bmod (p-1)) + p - 1] & \text{if } a > p-1, \quad b < p-1, \\ [(ab \bmod (p-1)) + p - 1] & \text{if } a < p-1, \quad b > p-1, \\ [p-1] & \text{if } a > p-1, \quad b > p-1. \end{cases} \tag{5.6.2}$$

In other words, the only non-trivial part of multiplication is multiplication modulo $p-1$. Division is easy in T. For example, use the extended Euclidean algorithm to find q such $bq \equiv 1 \bmod p-1$. Then compute $[d]/[b] = [(dq \bmod (p-1)) + e]$ where $e \in \{0, p-1\}$, choosing whichever value of e such that $([d]/[b])[b] = [d]$, if there is any (and there will be if $[d] = [a][b]$).

We next define an isomorphism $f : S \to T$, as follows:

$$f([y]) = \begin{cases} [y \bmod (p-1)] & \text{if } y < p, \\ [\log_g(y) + p - 1] & \text{if } y > p. \end{cases} \tag{5.6.3}$$

To be verified.

Diversion 5.6.5. If division by morphism minus the cost of evaluating the (partial) morphism f is efficient, then we can say that division is **metamorphically easy**.

Otherwise, we can say that division is **amorphously difficult**. Key agreement using a semigroup with amorphously difficult division could be called **amorphously seclusive**. Or, perhaps more clearly, it could be called **structurally seclusive**, on the grounds that the mathematical term **structure** is often used to mean something preserved across the isomorphism class.

5.6.3 Isomorphism for factorialized addition

This section considers a nonstandard semigroup operation on positive integers, **factorialized addition** (discussed further in §C.19.3.2). Factorialized addition $+_!$ is defined using prime factorizations via

$$\left(\prod_{i\geq 1} p_i^{a_i}\right) +_! \left(\prod_{j\geq 1} p_j^{b_j}\right) = \prod_{i,j\geq 1} p_{i+j}^{a_i b_j}, \tag{5.6.4}$$

where the positive prime integers are $p_1, p_2, \ldots,$. (So, $p_5 = 13$, for example.)

Because multiplication requires factorization of the inputs, multiplication is generally slow for large inputs, unless the inputs are represented in a factorized form. For the rest of this section, we switch to multiplicative notation, by writing write $[a][b] = [a +_! b]$.

We define a morphism $f : S \to T$, where T is multiplicative semigroup of the ring $\mathbb{Z}[x]$ of integer (univariate) polynomials, defined as

$$f : \left[\prod_{i\geq 1} p_i^{a_i}\right] \mapsto \sum_i a_i x^i. \tag{5.6.5}$$

For example, $f([20]) = f([p_1^2 p_3]) = 2x + x^3$. Evaluating f has a cost very similar to a single multiplication in S.

Division in $T = \mathbb{Z}[x]$ is achieved by a standard and efficient algorithm: **polynomial division** (see §**??** for a review).

The image $f(S)$ is the subsemigroup of T consisting of polynomials with with no constant term and all other coefficients non-negative.

The function g needed for division by morphism can be computed by determining primes of indices of the given indices corresponding to the degree of polynomial terms, raising to them powers of the coefficients. Algorithms to find a prime given its index are at least as efficient as factorization algorithms.

Consequently, the cost of division in S is almost the same as multiplication in S, being dominated by the cost of two factorizations of integers.

Diversion 5.6.6. The example semigroup aims to illustrate the arguably obvious fact that obscure complications to semigroup multiplication do not necessarily imply insurmountable obstacles for division.

In other words, it is *not easy to devise difficult division.* Of course, this is just another special instance of the cryptology maxim that *obscurity does not ensure security.*

5.6.4 Numerical isomorphism?

Work in progress, to be verified.

Suppose that S is a multiplicative semigroup. Suppose that S is practical, as described in §B.2. Suppose that $T = \mathbb{P}^+$, the additive semigroup positive integers, under standard addition. Suppose that $f : S \to T$ is an injective morphism, as in Lemma 5.6.1. Suppose that the complement $f(S)$ in T is a finite set.

Diversion 5.6.7. Subsemigroups of the non-negative integers under addition are often called **numerical** semigroups (usually if their complement is finite).

The theory of numerical semigroups is non-trivial (surprisingly), and has applications (due to Kakeya) to the theory of the symmetric functions.

Since $\{0\} \cup f(S)$ is a numerical semigroup (the f and S defined), we call such a morphism f a **numerical** morphism.

Next, we suppose a technical condition related to the usage of the semigroup in key agreement. We suppose that the complement of $f(S)$ is significantly smaller than than the sets of a and c values used by Alice and Charlie.

Finally, suppose that the procedure defining the inputs of the division problem has the following **non-growth** property. The seed values a and b, used to generate the instance $[d, b] = [ab, b]$, each have a probability distribution with the following property. Draw a random sample r drawn from the probability distribution, and let $p(n)$ be the probability that $f(r) = n$. We require that the $p(n)$ is a non-increasing function of n. In other words, larger values of $f(r)$ are rarer than smaller values of $f(r)$.

We now outline an idea for an algorithm to implement a morphism $f : S \to T$ with the property above, and also to compute its inverse function $g : T \to S$, as needed for division by morphism.

First there is a pre-computation phase, that is done before the input to f is given.

Select a sparse matrix m with entries small non-negative integers $m_{i,j}$, mostly filled with zeros, but with no column being all zeros.

Then sample many values $r_1, \ldots, r_n$ from S, using the probability for the value a and b. The value $z_i = f(r_i)$ is now an unknown value, which we will try to determine using linear algebra over the integers.

For matrix column j, compute the value:

$$c_j = r_1^{m_{1,j}} r_2^{m_{2,j}} \ldots r_n^{m_{n,j}}.$$

Each factor of the form r_i^0 is omitted from the product. Since no column is all zeros, each product on the right is non-empty, and therefore defined.

Scan for collisions in these value between columns. Suppose that $c_j = c_k$ is collision. Form, an integer equation for the unknown $f(r_i)$.

$$0 = (m_{1,j} - m_{1,k})f(r_1) + \cdots + (m_{n,j} - m_{n,k})f(r_n).$$

These equations are sparse.

Repeat until enough collisions to give a solvable system of linear equation for the n unknowns $z_1 = f(r_1), \ldots, z_n = f(r_n)$. This equations are homogeneous, so we can only solve up to a scalar factor. Because of the non-growth property of sampling, we take lowest scaling (or a small multiple thereof), as the most probable solution.

We expect that the solved values $f(r_u)$ to be the smaller values, since smaller integers are more likely to satisfy small-coefficient linear equations.

Next, instead of computing $d/b = g(f(d) - f(b))$ using Lemma 5.6.1, we outline a more direct division algorithm, based on Shanks' baby-step giant-step algorithm.

The direct division will be more general than Lemma 5.6.1 in the sense that it does not require computation of $f(b)$, and does not require b to drawn by the sampling procedure as a and the values r_i. Nonetheless, it is essentially the same idea as Lemma 5.6.1.

Each step of the algorithm requires computing a value $g(t)$, for some positive integer t, as described below. First, try to solve

$$t = x_1 f(r_1) + \cdots + x_n f(r_n)$$

with x_i positive integers. This is an equation in integers (and is a variant of the subset sum problem?). Given the x_i, then let

$$g(t) = r_1^{x_1} \dots r_n^{x_n}.$$

In order for this method of computing g to be effective for small values of t, we need the known values of $f(r_i)$ to be sufficiently small.

Next compute two lists:

$$D = \{dg(u)\}, \qquad B = \{bg(t)\}$$

where the integers t are selected randomly, from some appropriate range. Then try to find an intersection of these two lists, so that $dg(u) = bg(t)$.

Then answer $d/b = g(u - t)$.

We would need $g(u - t) = a$, so we should choose u in the range of sums of two random samples, such as $f(r_1) + f(r_2)$.

A critical question is how costly is this type of algorithm? In particular, is it any faster than trial multiplication?

5.7 Division by re-scaling

The section describes various division-by-rescaling methods, which meaning transforming the inputs, then dividing. Either the re-scaled inputs allow for easier division, or else it is form a **self-reduction**, showing that the difficulty of division does not vary (much) over the choice of inputs, at least for the semigroups in which division-by-rescaling works.

Several versions of division by re-scaling share the common features, whose notation will be the following.

The re-scaled divider will be written as $/$.

In most case, the re-scaled divider makes use of an existing divider, which will be written $\div$. The point of $\div$ is that its computation might be easy for specially crafted inputs, or more theoretically, $\div$ is used a hypothetical solution to the show the division problem can be self-reductive, with its difficulty not varying much over the inputs.

Let $r \in S$ be some element, that we call the **re-scaler**. In some case the re-scaler r can be chosen at random. In other cases, a special re-scaler r is selected, and division by re-scaling includes the cost of finding the special re-scaler.

5.7.1 Re-scaling by parallel right multiplication

Lemma 5.7.1. *If $(sr)\div r = s$ for all s (in other words, r is right-cancellative), then operator / defined by*

$$d/b = (dr) \div (br) \tag{5.7.1}$$

is a divider.

Proof. Compute:

$$\begin{aligned}((ab)/b)b &= (((abr) \div (br))b \\ &= (((abr) \div (br))br) \div r \\ &= (((a(br) \div (br))(br)) \div r \\ &= (a(br)) \div r \\ &= ((ab)r) \div r \\ &= ab,\end{aligned}$$

so $((ab)/b)b = ab$ for all a, b. □

Diversion 5.7.1. The divider / from (5.7.1) can sometimes work even if r is not fully right-cancellative.

For example, in the proof, right cancellation by r is used in two steps.

5.7.2 Re-scaling by inside-out multiplication

Lemma 5.7.2. *If $(s \div r)r = s$ for all s, (in other words, r divides s, on the right, written $r|s$, for all $s \in S$), then operator / defined by*

$$d/b = (d \div (rb))r \tag{5.7.2}$$

is a divider.

Proof. Compute:

$$\begin{aligned}((ab)/b)b &= (((ab) \div (rb))r)b \\ &= ((((a \div r)r)b) \div (rb))(rb) \\ &= ((((a \div r)(rb)) \div (rb))(rb) \\ &= (a \div r)(rb) \\ &= ((a \div r)r)b \\ &= ab,\end{aligned}$$

so $((ab)/b)b = ab$ for all a, b. □

Diversion 5.7.2. The condition that $r|s$ for all s is very restrictive on r. For example, it implies that r has a **left inverse** q such that $qrr = r$ (see Table B.2), which is $q = (r \div r) \div r$.

For example, it is true for $r = 1$, but then the lemma is vacuous, since $d/b = (d \div (1b))1 = d \div b$.

Diversion 5.7.3. The divider in (5.7.2) can sometimes even when $r|s$ fails to hold for all $s \in S$.

For example, in the proof, the only step requiring divisibility by r has $s = a$. When trying to compute d/b, we are not given a, so it may be difficult to find a specific r that divides a.

A variation of the divider in (5.7.2) takes input d and somehow tries to find r such that $r|a$ given that $d = ab$.

5.7.3 Re-scaling by multiplying ratios

Lemma 5.7.3. *If $(r \div s)s = r$ and $(t \div r)r = t$ and for all $s, t \in S$ (in other words $s|r|t$ for all s, t), then operator / defined by*

$$d/b = (d \div r)(r \div b) \tag{5.7.3}$$

is a divider.

Proof. Compute:

$$\begin{aligned}((ab)/b)b &= ((ab) \div r)(r \div b)b \\ &= ((ab) \div r)r \\ &= (ab),\end{aligned}$$

so $((ab)/b)b = ab$ for all a, b. □

Diversion 5.7.4. The divider $/$ from (5.7.3) uses the divider $\div$ twice, and is only useful when it can be arranged somehow that on cost of one of these is at most half of the cost that one is aiming for.

Diversion 5.7.5. The condition that $s|r|t$ for all s, t that $s|t$ for all $s, t \in S$.

Diversion 5.7.6. If $1 \in S$, then putting $r = 1$ in the divider of (5.7.3) is essentially division by inversion, it becomes $d/b = (d \div 1)(1 \div b)$.

Diversion 5.7.7. Putting $r = eb$ in (5.7.3) gives $d/b = (d \div (eb)((eb) \div b)$, which is very much like $(d \div (eb))(e)$, which is very much like divider (5.7.2).

Diversion 5.7.8. The proof above also works, for given d and b, if $b|r|d$.

Given b and d, finding such an r with that property would allow this division to work.

Diversion 5.7.9. In the semigroup of §2.18 derived from Diffie–Hellman key agreement, we can choose an r that depends on d, such that computing $d \div r$ in (5.7.3) is easy. Let $1 < h < p$, and let $r = d[h]$. Then, $d \div r = [h^{-1} \bmod (p-1)]$.

Choosing h uniformly at random in the given range, makes r range uniformly over the set of Diffie–Hellman public keys. In other words, computing a discrete logarithm (to fixed a base) is essentially equally difficult over all inputs.

5.8 Division by cross-multiplication

This section describes **division by cross-multiplication**.

Diversion 5.8.1. They can be many kinds of cross-multiplication, and only some will lead to division by cross-multiplication.

In other words, although division by cross-multiplication involves cross-multiplication, something more essential underlies division by cross-multiplication. Nonetheless, the name division by cross-multiplication describes what is happening at least at superficially high-level.

5.8.1 Cross-multipliers

A **cross-multiplier** is a binary operator written $*/$ such that

$$(a */ b)b = (b */ a)a, \tag{5.8.1}$$

for all $a, b \in S$ such that there exists $x, y \in S$ with $xb = ya$. See §B.8.13.3 for more detail.

5.8.2 Division by cross-multiplication

A right divider $/$ is **division by cross-multiplication** if:

$$d/b = (b \ast/\, d)\backslash(d \ast/\, b). \tag{5.8.2}$$

for some left divider $\backslash$ and some cross-multiplier $\ast/$.

To be clear, equation (5.8.2) can fail to define a right divider, for some choices of $\backslash$ and $\ast/$.

Diversion 5.8.2. In some semigroups, some of the fastest dividers use an equation (5.8.2), at least at a high-level. An example is Shanks' Baby-Step-Giant-Step algorithm.

An example of division by cross-multiplication is the following.

Lemma 5.8.1. *If $\ast/$ is a cross-multiplier, and $\backslash$ is a left post-divider and there exists a right post-divider, then the operator $/$ defined by (5.8.2) is is a post-divider.*

Proof. To prove that $/$ is a post-divider means to prove that $(ab)/b = a$ for all a, b, which is to say that $d/b = a$ if $d = ab$. A post-divider exists, so let $\div$ be any post-divider. Calculating,

$$\begin{aligned} d/b &= (b \ast/\, d)\backslash(d \ast/\, b) \\ &= (b \ast/\, d)\backslash(((d \ast/\, b)b) \div b) \\ &= (b \ast/\, d)\backslash(((b \ast/\, d)d) \div b) \\ &= (b \ast/\, d)\backslash(((b \ast/\, d)(ab)) \div b) \\ &= (b \ast/\, d)\backslash((((b \ast/\, d)a)b) \div b) \\ &= (b \ast/\, d)\backslash((b \ast/\, d)a) \\ &= a, \end{aligned}$$

proving that $/$ is a post-divider. □

5.8.3 Cross-multiplication by trivial operations

Some cross-multipliers are not too helpful for division, even though can be described easily.

- In a commutative semigroup, $a \ast/\, b = a$ defines a cross-multiplier.
- In a semigroup with a zero, then $a \ast/\, b = 0$ defines a cross-multiplier.

- In a group, then $a{*/}b = b^{-1}$, defines a cross-multiplier. (More generally, in a semigroup with a left post-divider, then defining $a *\!/\ b = q$ for any middle inverse q of b defines a cross-multiplier).

Cross-multiplication can also be obtained by applying co-multiplication and division, see §B.8.13.3.

A recursive strategy for cross-multiplication is to let:

$$a *\!/\ b = \begin{cases} a & \text{if } ab = ba, \\ ((ba) *\!/\ (ab))a & \text{if } ab \neq ba. \end{cases}$$

This strategy succeeds if it terminates (by reaching the first case after many iterations), but fails if it enters an infinite loop (repeating the second case forever).

5.8.4 Cross-multiplication by collision

A randomized strategy for cross-multipliers is to find matching entries (collisions) in two arrays $[r_1a, r_2a, \ldots, r_ma]$ and $[s_1b, \ldots, s_nb]$, so that $r_ia = s_jb$, and then set $a *\!/\ b = r_i$ and $b *\!/\ a = s_j$.

5.8.5 Arranging easier left division

Division by cross-multiplication uses division itself, specifically it uses a left division to implement a right division. This seems to limit to its general usefulness, but there are two useful case, despite this limitation.

Firstly, in a reductionist argument, division by cross-multiplication can show that right division is not much harder than left division, only needing some cross-multiplications extra. For example, in commutative semigroups, the left-operand function cross-multiplier define $a *\!/\ b = a$ can be used to show that left and right division are equally difficult (as one would expect).

Secondly, with a careful choice of cross-multiplier, it can be arranged that inputs to the left division operation fall into an easy case for evaluation. In other words, division by cross-multiplication is useful for division if the cross-multiplier produces easy-to-divide elements.

For example, in §2.18, the semigroup of associated with Diffie–Hellman key agreement, has elements, $[x]$ with $x < p$, in which division is easy. The cross-multiplication by collision method can be used to find such an easy-to-divide cross-multiplier. This resulting division algorithm is essentially a form of Shanks' Baby-Step-Giant-Step algorithm for computing the discrete logarithm.

5.9 Division in subsemigroups

Recall that a subsemigroup S is a semigroup T is a subset S of T closed under the multiplication of T, which makes S into its own semigroup, with multiplication defined the same as T (except it is less general).

Diversion 5.9.1. Division in subsemigroups can be considered as a special case of division by isomorphism. It seems an important enough case to treat separately.

Subsemigroup S might not be closed under a given divider $/$ in T. For example, even if $a, b \in S$, it might be that $(ab)/b \notin S$. In other words, a divider in T is not sufficient to construct a divider in S.

Nonetheless, the fact that S is not closed under $/$ is not an obstacle for computing a wedge by division in S. See §4.6.

A subsemigroup $S \leq T$ can potentially have much faster division than in T. In other words, even if T is secure for key agreement, a subsemigroup S might not be secure.

Diversion 5.9.2. Even if S is closed under the division operator $/$ in T and the operator $/$ is fast on average, in the sense of probabilistic division, applying $/$ might be slow on average. For example, if S has size (or probability measure) negligible in comparison to T, then the cost $/$ in S might contributed only negligibly to the cost of $/$ in T.

5.10 Division by Rees matrix index search

A strategy for division in the Rees matrix semigroup (see §4.7 and §C.8.6) $\text{Ree}_m(S)$ is to compute, is try computing

$$[i, d, l]/[k, b, l] = [i, (d/(m(j, k)b), j] \tag{5.10.1}$$

for multiple trials of j, until $(d/(m(j, k)b))m(j, k)b = d$. In other words, we seek j such that $(m(j, k)b)|d$. The cost of this division in $\text{Ree}_m(S)$ is at most $|J|$ division plus $2|J|$ multiplications in S (where J is the set of possible values for J, one of the index-sets for the matrix m).

Diversion 5.10.1. For comparison, computing a wedge in $\text{Ree}_m(S)$ by the method §4.7 did not use a search over J. So, computing a wedge is $|J|$ times faster than division.

5.11 Division in product semigroups

To be completed.

There are several ways to combine two (or more) semigroups into a yet another semigroup.

This section looks at how to divide in these various combinations.

Diversion 5.11.1. If one can show that the various product constructions do not increase the difficulty of division (or the wedge problem), then one can confine one's consideration to semigroup not composed by some constructions (indecomposable relative to such constructions).

5.11.1 Division by coordinate division in Cartesian products

The **Cartesian** product of semigroups S and T is the set of pairs $S \times T$, with multiplication defined $[s,t][u,v] = [su, tv]$. In the Cartesian product, **division by coordinate division** means a divider / such that:

$$[d,e]/[b,c] = [d/b, e/c], \tag{5.11.1}$$

where the dividers on the right are in S and T. In other words, divide in each coordinate.

Diversion 5.11.2. More generally, the cartesian product can be defined for any family of semigroups $\{S_i\}_{i \in J}$, as the functions $f : J \to \bigcup_{i \in J} S_i$, with the property that $f(i) \in S_i$.

Coordinate division is defined similarly,

Diversion 5.11.3. The cost of the division by coordinate division in $S \times T$ is typically the sum of the costs of division in S and T.

Diversion 5.11.4. When considering probabilistic division, the success rate of division by coordinate division in $S \times T$ is typically product of the success rates in division in each of S and T.

More precisely, this assumes that the input distribution of the coordinates in $S \times T$ are independent. For correlated inputs, the success rate is not necessarily the product.

Diversion 5.11.5. Sometimes, there can be dividers in $S \times T$ that are not division by coordinate division.

For example, if S has (at least) two distinct dividers, say / and $\div$, and T has more than two elements, then a divider in $S \times T$ can be defined to use / or $\div$ on its first coordinate, depending on the values of the second coordinate.

5.11.2 Division by deletion in free (co)products

The **free product** (or **co-product**) of semigroups S and T is set of non-empty arrays with entries in the disjoint union $S \uplus T$ modulo the congruence generated the condition that a pair adjacent entries in the same component semigroup (S or T) can be replaced by a single entry consisting of the product.

Diversion 5.11.6. In the category of monoids, a different **free product** arises, which is the semigroup free product modulo the further congruence that allows any entry of value of 1, in either S or T to be dropped.

The following division by deletion will not generally work in this free monoid product. (Is the free monoid product related to a fiber product?)

In the free product U of semigroups S and T, each element can be put in a canonical form where the array entries alternate between S and T.

A divider / in the free product is **division by deletion** if d/b is computed as follows. Put d and b standard form, with entries alternating in S and T, and suppose that these alternating arrays are $b = [b_1, \ldots, b_n]$ and $d = [d_1, \ldots, d_{m+n}]$, with $n \geq 1$ and $m \geq 0$.

If $m \geq 1$ and $d_{m+1} = b_1$, then let $d/b = [d_1, \ldots, d_m]$. If $m = 0$ or $d_{m+1} \neq b_1$, then let $d/b = [d_1, \ldots, d_m, d_{m+1}/b_1]$ (with $d_1, \ldots, d_0$ indicating no array entries).

5.11.3 Division by axial inversion in tensor products?

To be completed.

A **tensor** product of semigroups S and T can be defined. (See §C.9.1.5 for more discussion of the tensor product of semigroups.)

5.11.4 Division in semidirect products – ???

To be completed.

5.11.5 Division in fiber products – ???

To be completed.

5.11.6 Division in (co)equalizer semigroups – ???

To be completed.

5.11.7 Wreath product division?

Diversion 5.11.7. Wreath products arise in the Krohn–Rhodes structure theorem of semigroups. Consequently, division in wreath products might be especially important.

To be verified?

Given two multiplicative semigroups S, T, the **left wreath product** is $S \wr T$ is a semigroup with underlying set $T \times S^T$, (where, as usual, S^T is the set of functions $T \to S$), and multiplication rule:

$$[A, \alpha][B, \beta] = [AB, (\alpha \circ \lambda_B)\beta] \tag{5.11.2}$$

where:

- $\lambda_B : T \to T : t \mapsto Bt$ is left multiplication by B;
- $\circ$ means function composition, so that $\alpha \circ \lambda_B \in S^T$, with $\alpha \circ \lambda_B : t \mapsto \alpha(Bt)$;
- S^T is considered a semigroup with multiplication $\phi\theta : t \mapsto \phi(t)\theta(t)$, using multiplication in S.

Diversion 5.11.8. The **right wreath product** is defined similarly, except the underlying set is $S^T \times T$ and right multiplication $\rho_A : t \mapsto tA$ is used instead of left multiplication, so $[\alpha, A][\beta, B] = [\alpha(\beta \circ \rho_A), AB]$.

Diversion 5.11.9. We assume that some practical means is available to represent functions in the set S^T. If T is infinite, then this is probably infeasible, unless we restrict to some subsemigroup of the full wreath product, having only those functions which are practically representable, and so on.

A potential right division algorithm in the left wreath product is defined as:

$$[D, \delta]/[B, \beta] = \left[D/B, (\delta/\beta) \circ \hat{\lambda}_B\right] \tag{5.11.3}$$

where:

- D/B means applying a division algorithm in T;

- δ/β means applying a division in semigroup S^T, by extending to division algorithm S (to each value of the function);

- $\hat{\lambda}_B : T \to T : t \mapsto B\backslash t$ represents left division.

It can be verified that this division algorithm is successful as a mid-divider, provided that the left division by B, the function $\hat{\lambda}_B$, is successful as an pre-divider.

Diversion 5.11.10. A similar right division algorithm in the right wreath product can probably be built using only right division in the base semigroups: as $[\delta, D]/[\beta, B] = [(\delta/\beta) \circ \hat{\rho}_B, D/B]$.

To be completed.

What about the wedge in the wreath?

5.12 Division by binary search

Suppose that S is a countable, totally ordered semigroup. Fix an enumeration $s : \mathbb{P} \to S$ of S. Division by binary search is the following algorithm to compute d/b.

1. Let $n = 1$.

2. Repeat the following:

   (a) Let i_n be the smallest integer such that, for all $m < n$, it is true that $s(i_m)b < s(i_n)b$ if and only if $s(i_m)b < d$.

   - If no such i_n exists, stop and output an arbitrary value (likely incorrect).

   (b) If $s(i_n)b = d$, then output $d/b = s(i_n)$ and stop.

   (c) Else increase n by 1, and continue.

Division by binary search provides division because $d = ab$, then $a = s(j)$ for some j.

To be completed.

Diversion 5.12.1. The enumeration $s : \mathbb{P} \to S$ can be arbitrary, and need not have any connection with semigroup operation or the ordering. However, it should be efficient if the binary search algorithm is to be efficient.

Diversion 5.12.2. Binary search is a well-known general algorithm, with many more applications (not just division).

Diversion 5.12.3. When choosing i_n, one need not keep the history of all previous i_m, only the maximum below d and the minimum above d (whichever of these exist.)

Diversion 5.12.4. Division by binary search is like a type of trial multiplication, except the trial values $a' = s(i_n)$ can depend on inputs to division d and b. (Strictly speaking, trial multiplication as we have defined it, the trial values cannot depend at all on d or b.)

5.13 Division by complement-transpose (of relations)

The semigroup $\mathrm{Rel}(X)$ of relations on a set X, under composition, is well-known. For a review of detail, and some notations, see §C.5.6 and §C.15.2.

Two unary operations on $\mathrm{Rel}(X)$ are complement (written $d \mapsto d^c$) and transpose (written $b \mapsto b^t$).

Definition 5.13.1. *The* ***complement-transpose*** *divider / in* $\mathrm{Rel}(X)$ *is defined as*

$$d/b = (d^c b^t)^c. \tag{5.13.1}$$

Lemma 5.13.1. *The complement-transpose divider in* $\mathrm{Rel}(X)$ *is a (right) divider.*

Proof. The task is to show that $((ab)/b)b = ab$ for all a, b (from Definition B.8.1 of a mid-divider). For the complement-transpose divider, the task is then to show

$$((ab)^c b^t)^c b = ab. \tag{5.13.2}$$

Some notation and terms: for points $x, y \in X$ and relation $a \in \mathrm{Rel}(X)$, write xay to mean that $(x, y) \in a$, and call xay a **relationship**. Also, say xay is false if $(x, y) \notin a$.

We aim to prove (5.13.2) by showing $xaby$ implies $x((ab)^c b^t)^c by$, and conversely. So, the two directions of the implication are described separately, in each of the next two paragraphs.

Suppose $xaby$. By definition of ab in $\mathrm{Rel}(X)$, this means there exists $z \in X$, such that with xaz and zby. Fix this z (for rest of this paragraph). For all w, either zbw or not. If zbw is false, then $wb^t z$ is false (by definition of the transpose relation b^t). If zbw is true, then, since we already have xaz, together with zbw, we get $xabw$, and therefore $x(ab)^cw$ is false (by definition of complement). So, for each w, at least one of $x(ab)^cw$ or wb^tz is false. Equivalently, no w has both $x(ab)^cw$ and wb^tz. Therefore, $x(ab)^cb^tz$ is false. Consequently, $x((ab)^cb^t)^c)z$ is true. Since zby already, we have $x((ab)^cb^t)^c)^cby$, as desired.

Suppose $x((ab)^cb^t)^c)by$. Then there exists z with $x((ab)^cb^t)^cz$ and zby. The relationship $x(ab)^cb^t)^cz$ means that for all w, at least of one $x(ab)^cw$ and wb^tz is false. In particular, this applies if $w = y$: at least one of $x(ab)^cy$ of yb^tz is false. But zby means yb^tz, which is not false, so $x(ab)^cy$ must be false. If $x(ab)^cy$ is false, then $xaby$, as desired. □

Diversion 5.13.1. The proof above resorted to first principles, dealing with elements of X and so on.

A better proof might exist: one that works at the level of operating on relations using some more fundamental axioms between the various operations.

Diversion 5.13.2. In alternative notation for complement and transpose, the complement-transpose divider formula is $d/b = (d^-b')^-$.

Diversion 5.13.3. The cost of the complement-transpose divider is superficially easy to assess in terms of the underlying operations. However, the cost of the three operations in the semigroup $\mathrm{Rel}(X)$ might be difficult to assess.

If X is a large enough set, implementing semigroup multiplication for arbitrary relations might be infeasible. Even so, there may potentially be special subsemigroups of $\mathrm{Rel}(X)$ in which multiplication is, for some reason, efficiently implementable. These subsemigroups might not be closed under the complement or transpose operations. Nevertheless, the formula for the complement-transpose divider is at least worrisome.

To be completed.

Diversion 5.13.4. The complement-transpose divider is neither a post-divider nor a pre-divider.

Diversion 5.13.5. An existing theory of **residuated lattices** might include the complement-transpose divider.

Diversion 5.13.6. Left division in $\mathrm{Rel}(X)$ can be obtained from right division using transposition: $b\backslash d = (d^t/b^t)^t$. Starting with complement-transpose right division, this process obtain complement-transpose left division $b\backslash d = (b^td^c)^c$, because $b\backslash d = (d^t/b^t)^t = (d^{tc}b^{tt})^{ct} = (d^{ct}b)^{tc} = (d^{ctt}b^t)^c = (d^cb^t)^c$.

Diversion 5.13.7. The following terse snippet of J programming language code demonstrates, how associative key agreement using the semigroup of relation, and how complement-transpose division works.

```
mul=:+./ .*.
div=:(-.@mul~-.)~|:
rnd=:0=(?@$~2&#)"0
d=:a mul b[e=:b mul c['a b c'=:4 rnd 3#7
4 2$(f-:g);b;a;c;d;e;(f=:a mul e);(g=:d mul c)
(f-:(d div b) mul e),d-:(d div b) mul b
```

This code represents relations using boolean matrices.

5.14 Division by descent

Algorithms for integer division and polynomial division are well known.

Diversion 5.14.1. Integer division strategies date to ancient times. Polynomial division is also well-known and standard.

For a thorough treatment of classical division polynomial, see Knuth [Knu98, §4.3.1, Algorithm D] for integer division and [Knu98, §4.6.1], for polynomial division.

The **division by descent** strategy is one possible generalization of such classical integer and polynomial division algorithms.

Diversion 5.14.2. Classical division algorithms also allow the computation of a remainder when division is not exact. This means classical division problems actually solve a stricter problem that exact division that is of interest to key agreement.

This also means that classical division algorithms are not always the fastest division algorithms, because they are designed to a little extra work, namely finding a remainder if there is one.

For example, a strategy to compute 424263/141421 is to notice that both inputs have the same number of digits, and then divide the first (two) digits only, as in $4/1 = 4$, or $42/14 = 4$. On the premise that $424263 = a \times 141421$, this is likely to be correct. But the classical algorithms, and division by descent also, could be configure to start in a such manner, but would usually take at least one more step than beyond this of computing 4×141421 comparing to 424263. Ultimately, this because the first (two) digits are not enough to determine the remainder, or even if the result is 3.99 to be rounded down to 3, or 4.01 to be rounded down to 4.

Of course, the strategy in this paragraph might not work for general inputs, but the point is merely to state the division as define in this report can be easier, for some inputs, than the classical quotient-and-remainder problem.

Diversion 5.14.3. In number theory, special significance is attached to Euclidean domains, where there is a well-define notion of quotient and remainder.

In computational algebra, there is also a notion of multinomial division using Gröbner bases which is to used to find well-defined remainders.

5.14.1 Descenders

Division by descent uses addition, subtraction and a **descender**, which is now defined.

First, some background on addition and multiplication. A **realm** R is a set together with two binary operations, addition and multiplication. (See §B.12 for more discussion of realms.) Write $R^\times$ for the multiplicative magma of R, and R^+ for the additive magma. (If multiplication in R is associative, then $R^\times$ is a semigroup.)

Definition 5.14.1. *Binary operator $\vee : R^2 \to R$ on realm R is a* ***descender*** *if there exists a divider $/$ for $R^\times$ and a subtracter $-$ for R^+, such that:*

$$d/b \in \{d \vee b,\ ((d - ((d \vee b)b))/b) + (d \vee b)\}, \tag{5.14.1}$$

for all $d, b \in R$.

As detailed in the following section, the idea for division by descent is recursively apply the formulas above.

Diversion 5.14.4. Every divider $/$ is descender. To see this, put $d \vee b = d/b$, then the $d/b \in \{d \vee b, \dots\} = \{d/b, \dots\}$ holds (vacuously).

Consequently, descenders always exist, since dividers always exist.

Diversion 5.14.5. Suppose that realm R is a field. Every binary operator $\vee$ is a descender. Choose a divider $/$ such that $d/0 = d \vee b$. (If $b \neq 0$, then $d/b = db^{-1}$ is the only possible value for d/b.)

Diversion 5.14.6. If $0 \in R$ is both an additive identity and multiplicatively absorbing, then the constant zero-value operator $d \vee b = 0$ is a descender.

To see this, note that $((d - (d \vee b)b)/b) + (d \vee b) = ((d - 0b)/b) + 0 = (d - 0)/b = d/b$. (The usual rule that $d - 0 = d$ holds for any subtracter $-$ because $d - 0 = (d + 0) - 0 = ((d + 0) - 0) + 0 = d + 0 = d$.)

As will be seen the zero descender will not be usable for division by descent.

Diversion 5.14.7. In the realm of positive integers, the operator $d \vee b = d + b$ is not a descender.

To see this, consider $d = ab$. Then $d/b = (ab)/b = a < a(b+1) = ab + b = (ab) \vee b$, so $d/b \neq d \vee b$. But $d \vee b < ((d - (d \vee b)b)/b) + (d \vee b)$, so $d/b \notin \{d \vee b,\ ((d - ((d \vee b)b))/b) + (d \vee b)\}$.

Diversion 5.14.8. In the realm of positive integers, an operator $\vee$ is a descender if and only if $(ab) \vee b \leq a$ for all a, b.

Suppose that $\vee$ is a descender. Then $(ab)/b \in \{(ab) \vee b, \dots + ((ab) \vee b)\}$, implies each case, $a = (ab)/b \leq (ab) \vee b$, so the inequality holds.

Suppose that $(ab) \vee b \leq a$ for all a, b. Define a divider / be the usual rule $(ab)/b = a$, for all a, b, and otherwise $d/b = d \vee b$ if $d \neq ab$ for all a. If $d \neq ab$ for all a, then $d/b = d \vee b \in \{d \vee b, \dots\}$, as required. If $d = ab$, then $d/b = (ab)/b = a$. If $d \vee b = a$, then $d/b \in \{d \vee b, \dots\}$, as required. Otherwise, the hypothesis implies $(ab) \vee b < a$, which means that $((ab) - ((ab) \vee b)b) = (a - ((ab) \vee b))b$, so that $((a - ((ab) \vee b)b)/b) + ((ab) \vee b) = (a - ((ab) \vee a)) + ((ab) \vee b) = a = d/b$, as required.

Diversion 5.14.9. For concreteness, consider the following descender for the semiring of positive integers under multiplication. To compute $d \vee b$, let $e = 1$. Then loop as follows: while $10eb \leq d$, replace e by $10e$. Then return the last value of e as the value of $d \vee b$.

Diversion 5.14.10. Here is a worked example of how to use the previous descender to divide positive integers. We aim to compute 217/7.

First compute $217 \vee 7$. Put $e = 1$. Test the loop condition, which $10 \times e \times 7 = 70 \leq 217$, so replace e by $10e = 10$. Testing the loop condition, we see that $10 \times e \times 7 = 700 \not\leq 217$, so we stop, and get $217 \vee 7 = 10$.

Clearly, since $(217 \vee 7) \times 7 = 70 \neq 217$, it seems that $217/7 \neq 217 \vee 7$, here assuming that 7 divides 217. That would mean $217/17 = (217 - (217 \vee 7)7)/7 + (217 \vee 7) = (217 - 70)/7 + 10 = 147/7 + 10$.

A similar calculation gives $147 \vee 7 = 10$, and $147/7 = 77/7 + 10$. A third calculation gives $77 \vee 7 = 10$, and $77/7 = 7/7 + 10$.

Now, we get $7 \vee 7 = 1$, since starting with $e = 1$, the loop test condition $10e \leq 7$ fails. But now have $(7 \vee 7)1 = 7$, so $7 \vee 7$ meets the condition needed for 7/7, giving $7/7 = 7 \vee 7 = 1$.

Therefore, $217/7 = 147/7 + 10 = 77/7 + 10 + 10 = 7/7 + 10 + 10 + 10 = 1 + 10 + 10 + 10 = 31$.

5.14.2 Finite descent

Iterated descent means the operators defined:

$$d \vee_{\pm}^{0} b = d \vee b, \tag{5.14.2}$$

$$d \vee_{\pm}^{i+1} b = ((d - (d \vee b)b) \vee_{\pm}^{i} b) + (d \vee b). \tag{5.14.3}$$

Above, i is a non-negative integer, and the operator $\vee_{\pm}^{i}$ depends on the descender $\vee$ and on the addition and subtraction operators $+$ and $-$.

A divider / uses **division by descent** if

$$d/b = d \vee_{\pm}^{i(d,b)} b \tag{5.14.4}$$

for some function $i : R^2 \to \mathbb{N} = \{0, 1, 2, \dots\}$.

A simple method to compute $i(d, b)$ is take the minimum i such that $(d \vee_{\pm}^{i} b)b = d$. In a realm that has division by descent (for the given descender and subtracter), then this simple method could be almost the faster way to compute division by descent. The main difficulty would be for inputs with b not dividing d, such that $d \neq ab$ for all a. In this case, the division condition will never be realized, and some other condition will be needed to terminate the method.

For the purposes of attacking the seclusiveness of key agreement, we can assume that $d = ab$. For subtler attacks, we may instead wish to determine whether b divides d. In this latter, the question becomes whether division by descent can be used effectively to test divisibility.

Generally speaking, the efficiency of division by descent requires some notion of size of elements in the realm. Each division, descent, and subtraction need to make the sizes smaller, until a minimal size is reached.

5.14.3 Subtraction

Division by descent uses subtraction, so some method for subtraction is needed.

Subtraction is the additive version of division, so some of the strategies for division might be applicable. (Even division by descent might be applicable, if one has a suitable second operation to pair with addition).

Subtraction of polynomials, for example, can be done by coordinate subtraction (treating the polynomial coefficients as coordinates).

Subtraction of positive integers can use binary search, but a more practical version uses quotient and remainder over a special value r, which we call a radix (such as $r = 10$ for integers represented in decimal). This means two operators $/$ and $\%$ such that $d = (d/r)r + d\%r$ for all d, such that $(r - (d\%r)) + (d\%r) = r$. The following property:

$$\begin{aligned} d - b \in \{&(d/r - b/r)r + (d\%r - b\%r), \\ &((d-r)/r - b/r)r + (((d\%r) + r) - b\%r)\}, \end{aligned} \tag{5.14.5}$$

holds, and suggests to subtract recursively, by choosing the first element of the set whenever it leads to correct subtraction. The second elements correspond to the need to **carry** in long subtraction.

5.15 Power series division

Formal power series are an extension of polynomials, allowing arbitrary many terms. They form a semigroups under various operations, including addition, multiplication, and function composition (with some restrictions).

A practical implementation of power series cannot store an infinite amount of information of all the coefficients. So, either a finite set of coefficients is stored, an approximation or reduction modulo x^N, or else some kind of more formulaic representation is stored (such as rational functions, composite expression such as $e^{x\cos(x)}/(1+x)$).

5.15.1 Standard division

The semigroup of power series under standard multiplication can be divided by methods similar to polynomial division, except that one usually starts by determining the lower degree coefficients of the quotient first.

Knuth [Knu98, §4.7] describes such an algorithm.

5.15.2 Functional division

Power series, including polynomials, can be sometimes composed as functions.

This produces a semigroup, if we restrict to subsets of power series that are closed under function composition. For two examples: polynomials are closed under function composition; power series with zero constant term are closed under composition.

In this case, the usual algorithm for division d/b in this semigroup, is to invert b, as a power series (even if b was a polynomial). To avoid any confusion with standard inversion, this process is usually called **reversion**.

The basic algorithm for reversion of power series is due to Lagrange. Knuth [Knu98, §4.7, Algorithm L] describes this algorithm thoroughly, as well as two other algorithms for reversion (his Algorithms T and N).

5.16 Matrix division

Square matrices, with entries in a semiring, have associative multiplication (see §C.20.4).

Division of square matrices is part of linear algebra, which is well studied, at least when the underlying semiring is a commutative ring, or when the underlying semiring is a skew field (division ring) [HW15].

5.16.1 Matrix division over fields

For $n \times n$ square matrices with entries in a field, matrix division can be done with at most $O(n^3)$ field operations, using the algorithm usually called **Gaussian elimination**. There are very many matrix division algorithms.

Diversion 5.16.1. Matrix division amounts to solving a system of linear equations, which goes back almost to ancient times, far before Gauss.

Many matrix division algorithms freely use inversion of the (nonzero) matrix entries. (See [HW15], for the skew field case.)

In some semirings, inverses might not exist, or might be infeasible to compute. These matrix division algorithm will therefore be undefined or infeasible, in these cases.

5.16.2 Matrix division over commutative rings?

The **Bareiss algorithm** divides matrices without using inversion in the semiring of matrix entries. Instead, division in the base semiring is used.

The Bareiss algorithm requires that the semiring of matrix entries has commutative multiplication.

Chapter 6

Inversion: some generic strategies

This chapter lists some inversion strategies that try to find an inverse of an invertible element. (See §B.7 for discussion of the various types of inverses.)

The division-by-inversion and wedge-by-inversion strategies depend on an inversion strategy. Feasible inversion strategies can lead to attacks on the associated key agreement scheme (if it uses an invertible base point).

6.1 Power inversion

If b is a torsion element, meaning $b = b^{p+1}$ for some positive integer p (see B.4.3 for more detail), then, then b has a power inverse q (see B.7.6), meaning that $q = b^n$ for some positive integer n, and $q = bqb$. Any positive integer n such that $b = b^{n+2}$ is an **inversion degree**.

Power inversion means an algorithm to compute power inverse using knowledge about an inversion degree n.

If an inversion degree n is known exactly, then b^n can computed using the well-known square-and-multiply exponentiation, which takes at most $2\log_2(n)$ multiplications.

Diversion 6.1.1. The well-known square-and-multiply algorithm iterates the equation

$$b^n = \begin{cases} (b^m)^2 & \text{if } n = 2m, \\ (b^m)^2 b & \text{if } n = 2m+1. \end{cases}$$

The idea is to use this equation recursively, meaning to compute b^m by the same equation, replacing n by m. Each iteration takes at most two multiplications: one squaring of a value b^m, and at most one multiplication by b. Since $m \leq n/2$, it takes at most $\log_2(n)$ iterations to reach that case $m = 1$, where $b^m = b$ is given, and no more iterations are required.

If an inversion degree is known to exist within an interval $[m, M]$ for two exactly known positive integers m, M, then a power inverse can computed in at most $2 + (M - m) + 2\log_2(m)$ multiplications:

- Compute b^m, using the square-and-multiply algorithm, at a cost of at most $2\log_2(m)$ multiplications.
- Compute b^i for $i \in \{m + 1, m + 2, \ldots, M + 2\}$, using $2 + M - m$ multiplications by b.

If $m \leq n \leq M$ is an inversion degree of b, then $b^{n+2} = b$, so $b^i = b$ for $i = n + 2$. Therefore, some i will meet the that condition $b^i = b$, and then $n = i - 2$ will be an inversion degree, and b^{i-2} is an inverse of b.

Diversion 6.1.2. An alternative name for **power inversion** is **inversion-by-exponentiation**. Another alternative name is **Fermat inversion**, when considering multiplication modulo a prime integer, because of Fermat's little theorem.

Diversion 6.1.3. Division-by-power-inversion can be compared to division-by-trial-multiplication. Given an inversion degree, with $n < |S|$, then division-by-power-inversion takes at $2\log_s(|S|)$ multiplication, while division-by-trial-multiplication requires $|S|$ multiplication.

Diversion 6.1.4. Measured cost in multiplications can be misleading.

The cost of a multiplication in S may not be constant. In particular, elements might grow in size as they are multiplied. (This effect is most plausible if S is infinite.)

It is therefore possible that power inversion might not be slower than some forms of trial multiplication, if the powers b^i grow in size such that computing them becomes infeasible.

Power inversion of b requires some knowledge of an inversion degree n of b. A further strategy is needed to find this n.

6.1.1 Shor's period-finding algorithm

Shor's quantum-computer algorithm can sometimes find the order of a group. This might be usable to the period, and thus inversion degree of a torsion element of a semigroup.

Diversion 6.1.5. To do this, it might should help be able to distinguish and generate elements of $\langle b \rangle$ separately from other elements of S. In this case, $\langle b \rangle$ is a group, and is ready for application of Shor's algorithm.

Perhaps the converse is true. Being able to run Shor's algorithm on group $\langle b \rangle$ might imply being able to isolate $\langle b \rangle$ from the rest of S.

6.1.2 Elliptic curve point-counting

As an interesting pre-quantum example of finding the order is elliptic curve groups.

One of the best known algorithms for finding the cardinality of the elliptic curve group S over a finite field is to use finite group is the Schoof–Elkies–Atkin algorithm, which takes something like $\log(|S|)^4$ bit operations.

This example is a rather moot, however, because elliptic curve groups have much more efficient inversion algorithms than power inversions.

6.2 Inversion by division

In some cases, inversion of an invertible b is possible using division (by b or some other element).

Diversion 6.2.1. Inversion is only a means to end in this application of this report. Inversion-by-division will end up as being part of division-by-division or wedge-by-division. The main attack advantage will then be easier cases of division, or fewer applications of a more general division algorithm.

Lemma 6.2.1. *If b has a sidle inverse (meaning q such that $bbq = b = qbb$, see §B.7.5), then $b/(bb)$ is a middle inverse of b.*

Proof. Suppose that p is a sidle inverse of b, meaning $pbb = bbp = b$. We aim to prove that $q = b/(bb)$ is a middle inverse, meaning $bqb = b$. Calculating,

$$\begin{aligned} bqb &= b(b/(bb))b \\ &= b((pbb)/(bb))(bbp) \\ &= b(((pbb)/(bb))(bb))p \\ &= b(pbb)p \\ &= bbp \\ &= b, \end{aligned}$$

where the fourth equation removes / from the expression using the usual rule that $((xy)/y)y = xy$, in this case with $x = p$ and $y = bb$. □

Consequently, if $b(b/(bb))b \neq b$, then b has no sidle inverse, it is not sidle-invertible. Testing sidle-invertibility is possible with any divider.

When the previous inversion-by-division algorithm is in division-by-inversion algorithm, we get the following division-by-division algorithm. Let / be a given divider. A new modified divider ÷ by the rule

$$d \div b = \begin{cases} d(b/(bb)) & \text{if } b(b/(bb))b = b, \\ d/b & \text{if } b(b/(bb))b \neq b. \end{cases} \tag{6.2.1}$$

The new modified divider ÷ uses the old divider / one or two times (depending on the case above), and does two or three multiplications (depending on the case above). The main advantage of ÷ over / is when dividing by the same b many times. If that b is sidle-invertible, then operation / is used only once for all the operation ÷ involving the same b.

If semigroup S has a (cancelling) post-divider /, then another inversion by division is possible.

Lemma 6.2.2. *If / is a post-divider and \ is a left divider and b is middle invertible, then*

$$b\backslash(b/b) \tag{6.2.2}$$

is a middle inverse of b.

Proof. Let p be a middle inverse of b, meaning $bpb = b$. We need to prove that $q = b\backslash(b/b)$ is a middle inverse of b, meaning $bqb = b$. Calculating,

$$\begin{aligned} bqb &= b(b\backslash(b/b))b \\ &= b(b\backslash((bpb)/b))b \\ &= b(b\backslash((bp))b \\ &= (bp)b \\ &= b, \end{aligned}$$

where the step $(bpb)/b = bp$ follows from / being a post-divider. □

Diversion 6.2.2. The previous result could be strengthened by only requiring / to be partial post-divider, since it really only need $/_b$ to be unary post-divider.

If a semigroup has (fractioning) pre-dividers, of both the left and right kind, then yet another inversion-by-division is possible.

The following result goes back to at least 1951, as special case of a result of Green (the special case being when a semigroup has a single H-class).

Lemma 6.2.3 (Green?)**.** *If S is a non-empty (multiplicative) semigroup, and $/$ is an right pre-divider, and $\backslash$ is a left pre-divider, then S is a group.*

The identity is $1 = b/b$, for any $b \in S$. The inverse of b is $b^{-1} = 1/b$.

Proof. Since S is non-empty, there exists some $b \in S$. Let $1 = b/b$. Then

$$1b = (b/b)b = b,$$

with the second equation being due to $/$ being an pre-divider. Let a be any element of S. Then

$$1a = 1(b(b\backslash a)) = (1b)(b\backslash a) = b(b\backslash a) = a,$$

with the first and last equations being due to $\backslash$ being an pre-divider. The second equation is due to associativity, the third due to the previous result that $1b = b$. Moreover,

$$a1 = ((a/1)1)(1) = (a/1)(1(1)) = (a/1)(1) = a,$$

with the first and last equations from $/$ being an pre-divider. The third equation is due to the previous result $1a = 1$, which holds if $a = 1$.

We have established that $1a = a1 = a$ for all $a \in S$, so the S is a unital semigroup, with identity $1 = b/b$.

For each $a \in S$, define $a^{-1} = 1/a$. Then

$$a^{-1}a = (1/a)a = 1,$$

where the second equation follows by $/$ being an pre-divider. For brevity, write $a' = a^{-1} = 1/a$ for any $a \in S$. Then

$$aa^{-1} = (1)(aa') = (a''a')(aa') = a''(a'a)a' = a''(1)a' = a''a' = 1.$$

Therefore, a^{-1} provides the usual two-sided inverse of a group. □

In particular, if a semigroup S is not a group, then it lacks a pre-divider on at least one side.

Diversion 6.2.3. The lemma above does not hold if the condition of pre-dividers is replaced by condition of post-dividers. For example, consider semigroup $\mathbb{P}^{\times}$ of positive integers under multiplication. The standard integer division $/$ is a post-divider, because $ab/b = a$ for all a, b. A left post-divider is provided simply by swapping the operands. The positive integers do not form a group.

Moreover, the subsemigroup $\{2, 3, 4, \dots\}$ of positive integers greater than 1, is non-unital, and still has a right and left post-dividers.

Diversion 6.2.4. If S is a commutative semigroup, then an pre-divider $/$ exists if and only if S is a group.

Diversion 6.2.5. Recall that if S is a finite semigroup, then a divider $/$ is pre-divider if and only if it is a post-divider.

Diversion 6.2.6. Recall that an pre-divider $/$ exists if and only if right multiplication $\times_b$ is surjective for each b. If we define a binary operation to be **surjective** (or **injective**) precisely if the functions obtained by fixing one of the input have the property, then the lemma says that if a semigroup has surjective multiplication, then its multiplication is injective (but not necessarily the converse).

Diversion 6.2.7. Recall that the **left** semigroup $S = L(X)$ on a set X has multiplication rule $ab = a$ for all $a, b \in S$. Then $/$ defined by $d/b = d$ is an (right) pre-divider. Clearly, $L(X)$ is not a group. So, the lemma truly requires both left and right pre-dividers to exist.

Diversion 6.2.8. The lemma computes the identity and the inverse using right division. Therefore efficient inversion is possible if the right pre-divider is efficient. The left pre-divider need not be efficient: it suffices merely to exist.

However, an efficient left divider can be re-constructed by left multiplication by the inverse (assuming multiplication is efficient).

6.3 Inversion by wedges

Lemma 6.3.1. *If b is co-mutually invertible (meaning there exists an element q such that $bqb = b$ and $qbq = q$ and $qb = bq$, see §B.7.4), then the co-mutual inverse of b can be found using wedge operator $\wedge_{bbb}$ via:*

$$q = b \wedge_{bbb} b. \tag{6.3.1}$$

Proof. Let p be the co-mutual inverse of b. We need to show that $q = p$. But $b = bbbpp = ppbbb$, by our previous observations, so

$$q = (b \wedge_{bbb} b) = (ppbbb \wedge_{bbb} bbbpp) = ppbbbpp = p. \tag{6.3.2}$$

□

So, if b is co-mutually invertible (which is true if S is an inverse semigroup S), then we no longer need to worry that the wedge problem might be significantly easier than the division or inversion problem. Instead, we can study the division or inversion problem.

More obviously, suppose that S is a multiplicative monoid (a unital semigroup), and that $b \in S$ has a unital inverse q, meaning that $qb = bq = 1$. Then, $q = qbq = qb \wedge_b bq = 1 \wedge_b 1$.

6.4 Inversion of relations?

To be confirmed.

Let b be a relation on a set X. Let $S = \mathrm{Rel}(X)$ be the semigroup of relations under composition. Then b has a middle inverse in S if and only if

$$b(b^t b^c b^t)^c b = b, \tag{6.4.1}$$

in which case, $q = (b^t b^c b^t)^c$ is a middle inverse of b.

6.5 Inversion of matrices over fields?

To be verified and corrected.

A square matrix b over a field has a middle inverse, even if it is non-singular (determinant zero).

Diversion 6.5.1. A square matrix over the field of real numbers has an even stricter type of inverse: the Moore–Penrose inverse. (The Moore–Penrose inverse is a middle inverse.)

Assuming the existence of a middle inverse q, a simplistic way to find q relies on the fact that $bqb = b$ defines a linear system in the entries of the unknown matrix q. Existence of q, implies the system has at least one solution, so that it is solvable.

Conventional linear algebra algorithms for solving systems can therefore be applied to solve for q. When b is an $n \times n$ square matrix, the the linear system for q amounts to solving a linear system of n^2 equations in n^2 unknowns. Therefore, this approach might not be the most efficient, since it converts an $n \times n$ matrix problem into an $n^2 \times n^2$ matrix problem.

We next describe another algorithm to find middle inverse. It is derived from any algorithm for finding the matrix inverse of non-singular matrices, of size equal to b or smaller. The described algorithm also proves the existence of a middle inverse.

Again, the next algorithm is probably not the most efficient algorithm for computing a middle inverse. The standard matrix inverse of non-singular matrix is a stricter condition than being a middle inverse. Being based on a looser condition, finding a middle inverse should be easier to find than a standard inverse.

If b is non-singular, then just use standard matrix inversion.

If b is singular with rank k, then b can be factored into two rank k rectangular matrices as $b = cr$ where c has k columns and r has k rows. For example, c can be chosen by taking k independent columns of b, with r indicating how other columns of b are linear combinations of the columns in c. Some standard matrix inversion algorithms will produce c and r when applied to singular matrices.

The matrix rc will be a $k \times k$ square matrix.

If rc is a non-singular matrix, then we can compute usual matrix inverse $(rc)^{-1}$ by the given standard algorithm. Then we can put $q = c(rc)^{-2}r$, with the effect that

$$bqb = (cr)(c(rc)^{-2}r)(cr) = c((rc)(rc)^{-1})((rc)^{-1}(rc))r = c(1)(1)r = b.$$

so $bqb = b$, showing that q is a middle inverse. Similar calculations show that $qbq = q$ and $bq = qb$ so that q is a co-mutual inverse of b.

Diversion 6.5.2. It would nice to write $q = b^{-1}$ for a co-mutual inverse q of b, because a co-mutual is unique. But, this could cause confusion here, because the notation b^{-1} is usually only used for non-singular b. Some readers seeing b^{-1} might interpret this to imply that b is non-singular, which is not the intention here.

Diversion 6.5.3. The matrix $b = \left(\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right)$ has no co-mutual inverse q. To see this, suppose q were a co-mutual inverse. Then $b = bqb = bbq = 0$, because $b^2 = 0$. So, the matrix rc fails to be non-singular, for this b.

Otherwise rc is singular, and a more general method is needed.

If $\hat{r}$ is k-row matrix such that $\hat{r}c = I_k$, the $k \times k$ identity matrix, and $\hat{c}$ is k-column matrix such that $r\hat{c} = I_k$, then $q = \hat{c}\hat{r}$ is a middle inverse of b, since $bqb = cr\hat{c}\hat{r}cr = c1_k1_kr = cr = b$. (It is also a mutual inverse, but not necessarily a co-mutual inverse.)

A matrix $\hat{r}$ can also be found using the standard matrix inverse. Extend the k-column matrix c to a non-singular square matrix C, by appending independent columns that form a basis of the vector space. Matrix C has the same shape as C. In fact, if c is formed from the columns of b, then C takes those columns, places them to the left, and fills in the remaining columns, non-singularly, meaning that C can be regarded a non-singularized modification of b.

Next, compute the matrix inverse C^{-1}, by the given matrix inversion algorithm for non-singular matrices. The rows of C^{-1}, as vectors, form a dual basis to the columns of C, as vectors. Let $\hat{r}$ be formed from the first k rows of C^{-1}. The k rows of $\hat{r}$ are dual to the k columns of c, so $\hat{r}c = 1_k$.

Similarly, the matrix $\hat{c}$ can be constructed, transposing rows and columns in the construction above. This constructions always work, since any independent set of vectors can be extended to a basis. Therefore, the middle inverse always exists.

To repeat, there is likely a much more efficient algorithm.

6.6 Custom inversion algorithms

In some semigroups, specific fast inversion algorithms are available. For example,

- In the group $\mathbb{Z}^+$ of integers under addition, inversion is the function $a \mapsto -a$. For typical integer implementations, this merely requires toggling of the sign flag (stored as a bit).

- In $\mathbb{F}_p^+$, with additive notation, inversion of a is obtained by $p - a$ (using standard integer subtraction).

- In $\mathbb{F}_p^\times$, inversion of nonzero a can be obtained through the extended Euclidean algorithm, or one of its variants.

- In elliptic curve groups, inversion is usually more efficient than multiplication. For example, in short Weierstrass coordinates, the inversion of point (x, y) is $(x, -y)$.

Appendix A

Some applications of key agreement

Key agreement is not an end in itself, but a means to a greater security aim. Typically, key agreement must be part of a larger system, such as:

- a subsystem that **authenticates** the key agreement deliveries d and e, because, otherwise key agreement would vulnerable to **man-in-the-middle**, where an adversary replaces a delivery, say d, by his own d',
- a subsystem that **applies** the agreed keys for some purpose, such as keeping application data private between Alice and Charlie, because, otherwise, (if the keys were not used at all), then there would be no need to agree on them,

In other words, the application relies on the key agreement, and the key agreement relies on the authentication. These three subsystems may be thought of a chain, each subsystem being a link. The usual saying applies: a chain is only as strong as its weakest link.

Diversion A.0.1. In some cases, the method used to authenticate deliveries in key agreement can be re-used to authenticate the application data. For example, if deliveries are authenticated using a digital signatures, then application data can be authenticated using digital signatures.

In other cases, the authentication method for deliveries is not re-usable for application data. For example, Alice and Charlie can use **public hand-off** to authenticate their deliveries. Public hand-off means that they meet in public, verbally read out their deliveries to each other (making minimal effort to hide the deliveries), and then type these deliveries into the secure devices. (The point of public hand-off to minimize the reliance on any third parties for authentication, especially any network devices that connects to Alice and Charlie's secure devices.)

If Alice and Charlie use public hand-off to authenticate their deliveries, then they can also exchange digital signature verification keys. Then, the digital signatures can be used from then on for authentication. So, authentication of the application data can be done using digital signatures, without ever relying at all on key agreement.

In practice, Alice and Charlie may purchase their secure devices from a trusted retailer, who has installed within the secure devices some digital signature verification keys. Essentially, the retail purchase is a form a public hand-off. These directly verified keys are known as **trust anchors**. The owner of the trust anchor key as a **root certification authority**. Authentication of messages is then achieved by combining a certification chain with a digital signature of application data. These system, excluding to the part about signing the actual application, are part of a system, usually called a **public key infrastructure** (PKI).

If key agreement is securely achieved, then Alice and Charlie can use also a message authentication code (MAC) instead of digital signature to authenticate application data. Modern approaches to symmetric-key cryptography tend to combine both encryption and authentication. Using a MAC with an agreed key has the advantage of relying less on the security of digital signature scheme. If the deliveries use public hand-off, then trust in the third parties of the PKI can be avoided.

In some situations, the authentication method for deliveries is not desirable for the application data. In particular, digital signatures can be forwarded to third parties for verification. Digital signatures lack deniability. Asynchronous key agreement has a high degree of deniability, because both Alice and Charlie can independently compute the agreed key. In other settings, a digital signature is perhaps too slow. (One of these reasons might explain why the transport layer security (TLS) protocol using digital signature for authentication in the handshake sub-protocol but not in the record sub-protocol.)

A.1 Public-key encryption

In public-key encryption, Alice would publish her delivery d as fixed value. The published delivery is then called a **public key**, and can be written as $A = d$, to better illustrate the relation between Alice's value a and her public key A.

Diversion A.1.1. In many applications of asynchronous key agreement, the deliveries are meant to be used only once. The term **delivery** reflects this fact. It is little odd to re-use a delivery. Thus, re-branding may be appropriate when the deliveries are re-used.

Diversion A.1.2. The term **public key** has a very strong tradition, so we follow it now, despite the many reasons for not following this tradition for the corresponding term **delivery** in general key agreement.

Yet other alternatives for the term **public key** are considered briefly here.

The Bitcoin protocol used the term **address** for values derivable from a public (signing) key. In my opinion, their term **address** would be an improvement over **public key**, even if applied directly to the public key.

The terms **address** is also closely connected in meaning to **delivery**. Usually the recipient's address must be know to the sender, so the term **address** seemed inappropriate for asynchronous key agreement.

The term **address** already has many other meanings, such as email address, IP address, web site address, postal address. So, I would suggest calling **crypto address**. The term **crypto address** is a little like a **postal code**, an important value, but not meant human consumption, but rather for the underlying system.

In yet other expositions, I have used the term **locus**, to represent the similarity of the delivery to a lock, to a location (address), and to the fact that a delivery in elliptic curve DH key agreement is a point in the plane, which may also be described a locus of zeros of a set of equations (an elliptic curve, and one or more lines).

When Charlie wishes to send a message m to Alice, he does the following.

1. He fetches her published delivery $d = A$ (now also called a public key, or even crypto address), from a trusted source.
2. He computes his delivery e, according to the asynchronous key agreement scheme.
3. He computes the agreed key g.
4. He then uses a (probabilistic) function E, usually called **authenticated encryption**, to a create a cipher(text) $C = E_g(m)$.
5. He sends $[e, C]$ to Alice.

When Alice receives $[e, C]$, she computes her agreed key f, which should equal g, unless $[e, C]$ has been modified. Then she uses another function D, called **authenticated decryption**, to recover a possible message $m' = D_f(C)$. If $[e, C]$ was transmitted successfully, we will have $m = m'$, because $f = g$ and $D_f(E_f(m)) = m$ for all f and m.

Diversion A.1.3. The pair consisting of the authenticated encryption and decryption functions are also called **data encapsulation mechanism** (DEM), as explained in further below.

The security of public-key encryption has various definitions, such one abbreviated to IND-CCA2. In some case, a security proof for a public-key encryption is condition on some assumption, including the basic security of underlying key agreement scheme, such as Diffie–Hellman key agreement.

A.2 Key encapsulation

Shoup introduced the notion of **key encapsulation**, as a notion which intermediate between some types of key exchange and public-key encryption.

A **key encapsulation mechanism** (KEM), as this report would call it, is essentially non-asynchronous key agreement scheme, except that two of the four functions are merged into one.

Each asynchronous key agreement scheme can be used to develop a KEM, simply by merging two of the functions on one side into one function.

Again, security proofs for a KEM are often relations to the basic security of the underlying key agreement scheme (such as Diffie–Hellman key agreement).

To be completed.

A.3 Non-interactive key exchange

An asynchronous key agreement scheme can be used as part of a **non-interactive key exchange** (NIKE) scheme, a folklore notion named and formalized by Cash, Kiltz, and Shoup [CKS08], to generalize what is sometimes called **static** Diffie–Hellman key exchange.

In terminology of this report, a NIKE scheme has each user generate a single, static delivery value. The single delivery is registered with an authority, together with an identifier for the user. When two users use NIKE, the authenticity of the static deliveries is provided by the authority. The two users do not need to interact directly with each other in real time, having obtained the static deliveries from the authority during some provisioning phase. Hence the term **non-interactive**.

Conversely, any given NIKE scheme implies an asynchronous key agreement by forgetting such details as the registration authority, the identifiers and so on.

Past definitions of NIKE assume that each user has identical pair of algorithms, **public key generation** and **exchanged key computation**. Consequently, the existing NIKE definitions assume that the underlying asynchronous key agreement scheme is ambidextrous (per earlier notes): there are no left and right users.

However, it may only only require minor modification of the NIKE definitions to accommodate chiral asynchronous key agreement schemes.

A.4 Handshakes of interactive communication protocols

Perhaps the application of key agreement most often used in practice, is an interactive **secure connection** protocol between Alice and Charlie. Examples include the Transport Layer Security (TLS) protocol and the Internet Key Exchange (IKE) protocol, both of which can use Diffie–Hellman key agreement.

In this type of application, (asynchronous) key agreement is used near the start of the secure connection session during a phase called **handshake**. Often, handshake includes key agreement deliveries from both sides, which are only used once, to achieve a property often called **forward secrecy**. A handshake might also include other information beyond the deliveries used in key agreement. It may include introductions, negotiations, and authentication tags, which are outside the scope of key agreement as defined in this in report.

Diversion A.4.1. Forward secrecy, and several other advanced security aims, are best defined at the level of the handshake part of the secure communication protocol. At the lower level of (asynchronous) key agreement, which is merely a component of the higher protocol, more basic security aims can be defined, with the advanced security aims deriving from the basic security security.

For example, forward security can only be defined when it is clear which of Alice and Charlie's secrets a and c are to be re-used, and which are used only once. The handshake protocols within secure communication protocol define such details. The lower level key agreement scheme, as defined in this report, does not specify whether the secrets are re-used or not, so forward secrecy is not define-able.

Cryptographers have implicitly accepted this in the context of Diffie–Hellman, by separating higher security aims from more basic problems such as Diffie–Hellman problem.

This report generalizes Diffie–Hellman to larger class of mathematical operations, under the label of key agreement scheme, but retains the distinction that the generalization is a more basic primitive whose security properties are more basic.

As with both public-key encryption and non-interactive key exchange, these handshake protocols include steps for authenticating the delivery values. Many handshake protocols ensure the authenticity of the deliveries (usually Diffie–Hellman public keys), by applying a digital signature to the fresh deliveries. The signing keys used for digital signature, are authenticated using certificates, as part of a public key infrastructure.

A.5 Trust networks

Typically, Alice and Charlie must use some means to protect the deliveries d and e, for reasons of security, such as to avoid a **man-in-the-middle** attack. Generally, such protections are provided outside the narrow scope of key agreement (as defined in this report).

Diversion A.5.1. The term **out-of-band** is often use to describe communication through an alternative channel, and in cryptography, some security aims may be achieved using a out-of-band channel. This notion applies to key agreement if think of all data beyond the key agreement deliveries as out-of-band: then security aims such as protection against man-in-middle attacks is provided out-of-band.

Of course, in other settings, the data sent to authenticate the deliveries will share the same the communication band as the deliveries, so this usage of out-of-band is not entirely appropriate.

Many of the applications above rely upon some kind of many-party **trust network**, such as a **public key infrastructure** with **certification authorities** and **trust anchors**.

A typical public-key infrastructure involves applying digital signatures to public keys. As a network, it may be distributed, or it may hierarchical.

It may also be possible to build a trust network without digital signatures, instead relying on asynchronous key agreement. For example, Alice could **hand** an initial delivery d to Charlie, and Charlie could **hand** an initial delivery e to Alice. They keep these deliveries safe, but only used the agreed key to authenticated fresh deliveries that they **send** each over unauthenticated channels. The fresh agreed keys are authenticated using the initial authenticated keys.

This would only create one link between Alice and Charlie. But as each user develops more links, an authenticated network would be grow. Then authenticated deliveries could be sent through this network between non-adjacent members of the network. Of course, such a scheme would require careful study to determine its security: some mechanism would be needed to resist corruption at a single node of the network, for example.

Appendix B

Semigroup theory: some basics

This chapter covers a few relevant basic properties of semigroups in general.

- It reviews customary notions and notations (such as associativity, semigroups, multiplication, addition, powers, subsemigroups, generators, zero, one, idempotents, inverses, morphisms, and Green's relations).
- It states some idiosyncratic definitions (such as various definitions of division, wedges operation, and epigram or monogram representations).
- It proves a few elementary facts (such as relations between division types – proofs being special cases of classical algebra like Green's theorem).

Table B.1 summarizes the most essential basics.

Other parts of this report use these basics to describe various strategies for semigroup algorithms.

More thorough treatments of semigroup theory can found outside this report, in the listed references, Wikipedia, or the journal *Semigroup Forum*, and it is by skimming these resources that any non-trivial results appear in this chapter. This chapter adopts narrower treatment a semigroup theory than in other treatment of semigroup theory. It is not ambitious, and does not describe any of the powerful classification theorems for semigroups. Rather, it aims to answer (my own) naive semigroup-beginner questions that lead up to some properties division and wedges that might aid in the understanding for the security analysis of many forms of associative key agreement. Along the way, there are many small diversions, which could easily be irrelevant to cryptography.

Term	Notation	Condition
Magma	M^*	$* : M^2 \to M$
Operation	$a * b$	$[a, b] \mapsto a * b$
Additive magma	M^+	$[a, b] \mapsto a + b$
Multiplicative mg.	M (or S)	$[a, b] \mapsto ab$
Associative (mult.)	ab	$a(bc) = (ab)c$
Commutative (mult.)	ab	$ab = ba$
Medial (mult.)	ab	$(ab)(cd) = (ac)(bd)$
Semigroup (= assoc. mg.)	S^*	$* : S^2 \to S$, $a * (b * c) = (a * b) * c$
Multiplicative sg.	S	$S^2 \to S : [a, b] \mapsto ab$, $a(bc) = (ab)c$
Multiplication (times)	ab	In semigroup S: $a(bc) = (ab)c$
Addition (plus)	$a + b$	In semigroup S^+: $a + (b + c) = (a + b) + c$
Division (over)	a/b	$((ab))/b)b = ab$
Subtraction (minus)	$a - b$	$((a + b) - b) + b = a + b$
Wedge	$d \wedge_b e$	$(ab) \wedge_b (bc) = abc$
Left division	$a \backslash b$	$b(b \backslash ba) = ba$
Post-division, (cancelling)	a/b	$(ab)/b = a$
Pre-division, (fractional)	a/b	$(a/b)b = a$
Identity (one)	1 (or I)	$1a = a1 = a$, $1 \in S$
Identity (zero)	0 (or 0^+)	$0 + a = a + 0 = a$, $0 \in S^+$
Absorbing (zero)	0 (or ∞)	$0a = a0 = 0 \in S$
Absorbing (infinity)	∞	$\infty + a = a + \infty = \infty \in S^+$
Power (exponentiation)	$a^n \in S$	$a^1 = a$, $a^{n+1} = a^n a$, $(n \in \{1, 2, 3, \dots\})$
Idempotent	e	$e^2 = e$
Idempotent-ordering	$e \leq f$	$e = ef = fe$, $e^2 = e$, $f^2 = f$
Morphism	$\mu : S \to S$	$\mu(ab) = \mu(a)\mu(b)$
Subsemigroup	$T \leq S$	$a, b \in T \implies ab \in T$
Generated sbsg.	$\langle B \rangle$	$\bigcap_{T : B \subseteq T \leq S} T = \{b_1 b_2 \dots b_n : b_i \in B\}$
Order	$\mathrm{ord}(b)$	$\mathrm{ord}(b) = \vert\langle\{b\}\rangle\vert = \vert\{b, b^2, b^3, \dots\}$
Period, torsion	p	$b^n = b^{n+p}$, $p \in \{1, 2, 3, \dots\}$
Middle inverse	q	$bqb = b$
Co-mutual inverse	b^{-1}	$bb^{-1}b = b$, $b^{-1}bb^{-1} = b^{-1}$, $bb^{-1} = b^{-1}b$.
Nilpotent sg.		$S^n = \{s_1 s_2 \dots s_n : s_i \in S\} = \{0\}$, $n \in \{1, 2, 3, \dots\}$.

Table B.1: Some semigroup notations, terminology and definitions

Ring theory is mainstream and well-developed topic in modern algebra, and arguably more mature and more richly developed than semigroup theory. Because every semigroup embeds into a ring being isomorphic to a subset of the ring that is closed under multiplication, many of the deep theorems in ring theory might have impact in semigroup theory, and thereby impact on key agreement. But, by comparison, the basics presented chapter are all the more negligible in comparison to deep theory of rings.

In other words, the reader is asked to read this chapter with plenty of proverbial salt.

B.1 Operations

A **semigroup** is a standard definition from algebra:

Definition B.1.1. *A* ***semigroup*** S^* *is a set* S *together with an associative binary operation* $* : S^2 \to S : [a, b] \mapsto a * b$*.*

Diversion B.1.1. Recall again that binary operation $*$ is **associative** if

$$a * (b * c) = (a * b) * c$$

for all $a, b, c \in S$.

Diversion B.1.2. A **magma** is a weaker notion: a magma S^* is a set S together with a binary operation on S. Every semigroup is magma, but a magma is not a semigroup if its binary operation is not associative.

Typical proofs that S^* forms a semigroup first show that S^* is magma, and then show that $*$ is associative.

Some definitions and facts about semigroups generalize to magmas. In particular, division can be defined for magmas.

Diversion B.1.3. Semigroups are ubiquitous in algebra. A few examples of semigroups are sketched in Appendices C.

B.1.1 Multiplicative semigroups

If the binary operation of a semigroup is written ×, or any other notation called **multiplication**, such as $\cdot$ or $*$ or $\otimes$, then the semigroup is said to be **multiplicative**.

Being multiplicative is merely a matter of **notation** and **terminology**. Every semigroup can be made multiplicative by changing the notation and terminology for its binary operation, without affecting its structure, its efficiency or its security.

Diversion B.1.4. Given a non-multiplicative semigroup, such as S°, with a binary operation written $\circ$, we sometimes form a multiplicative semigroup by surround elements of S in square brackets, as $[a]$ for $a \in S$, so that $[a][b] = [a \circ b]$.

By default, assume that a semigroup is multiplicative. For example, when considering hypothetical semigroups, or semigroups in general, work the multiplicative notation and terminology.

Diversion B.1.5. When one set has two associative binary operations, multiplicative notation can only be used clearly for one of the two operations. Common example are the integers, the rationals, the reals, polynomials, and many other rings, all of which have associative binary operations: addition and multiplication.

Diversion B.1.6. Call **familiarization** the practice of re-using familiar notation, such as multiplication, for potentially strange objects, such as arbitrary semigroups.

A benefit is factor out similarities between the familiar from the strange, in order words, to leverage our existing intuition or wisdom.

A cost is to falsely suggest all that is true of the familiar (such as standard rational or arithmetic) is also true for the strange (such as arbitrary semigroups), which may have counterintuitive features.

This report continues the program of familiarization for multiplicative notation to other notations, such as division. Indeed, this report perhaps goes further more so than in standard notation for semigroups (let alone in cryptography). The familiar notations of division and divisibility are carried over to general semigroups, rather than introducing newer notations.

Again, the caveat with re-using standard notations no longer obey all of the usual rules when applied to arbitrary semigroups.

B.1.2 Unwritten multiplication

By default, the multiplication symbol is implicitly omitted in algebraic formulas, a traditional mathematical convention. Put another way, the multiplication symbol is empty, silent, or invisible.

Diversion B.1.7. When reading ab to others, one can say **a b** or **a times b**, depending on the need concision or clarity.

The tradition of an unwritten multiplication symbol is extended the label for the semigroup: instead of writing S^* or $S^\times$, we write S, indicating an unwritten multiplication symbol. In other words, a semigroup is indicated by its underlying set, with the multiplication operation being understood from context.

Diversion B.1.8. Disadvantageously, this unwritten multiplication tradition sometimes precludes elements of S to be written with multiple symbols per element, because these could be confused with multiplication of the symbols, as in the following examples.

- It prevents multi-letter variable names, used so often in computer programming. Most mathematicians can dealt with issue quite well, by the use of subscripted variables, such $x_1, x_2, \ldots$ or even x_{length}, and so on.
- Multi-digit numerical elements, such as real numbers, are formed by concatenation of digits, and possibly a decimal point. Multiplying two numbers, such as 1.1 and 5, can be notated by explicit indicating multiplication operator as $1.1 \times 5 = 5.5$. Or, to zealously avoid writing $\times$, by using parentheses as in $(1.1)(5) = 5.5$. Superfluous superscripts can also save parentheses, as in $90 = 2^1 3^2 5$. Sufficiently abstract expositions can side-step this particular issue by forgoing multi-digit arithmetic examples.
- Some semigroups have elements represented by sequences of elements (words) drawn from some set (an alphabet). Using commas to separate sequence is a heavy notation, so a common convention is to represent the word as the concatenation of the letters. When the words also equal the product their elements, then this convention largely matches the default notation for multiplication, up to a point, perhaps of equivalency of sequences. This also introduces a potentially new confusion between the variables and alphabetic values for sequence elements (word letters, or alphabet).

B.1.3 Additive semigroups

A semigroup such as S^+ with semigroup operation is written as $+$, is called **additive**, and the operation is usually called **addition**. Semigroups with similar notation, such as $\oplus$, are also called additive.

Diversion B.1.9. Usually $a + b$ is read as a **plus** b.

B.1.4 Multiplying sets

Subsets of a semigroup (or magma) are often multiplied.

Definition B.1.2. *If $A, B \subseteq S$, then let*

$$AB = \{ab : a \in A, b \in B\}. \tag{B.1.1}$$

Also, write $aB = \{a\}B$ and $Ab = A\{b\}$.

Diversion B.1.10. In non-multiplicative semigroups, translate as usual, so $A + B$ in additive semigroups.

Diversion B.1.11. Set-multiplication makes 2^S, the set of subsets of a semigroup, into its own semigroup.

B.2 Practicality

Semigroups need to be **practical**, in the sense described in this section, to be useful either for associative key agreement (§2.13) or for probabilistic key agreement (§2.9).

The general idea is that all the user algorithms in associative key agreement are practical, meaning users can run them in a short time.

Diversion B.2.1. The reason to distinguish practical semigroups is the abstract semigroup theory includes many impractical semigroups.

B.2.1 Practical multiplication

Users of an associative multiplication will need to multiply semigroup elements.

Definition B.2.1. *A multiplicative semigroup S has $[A, B, C]$-**practical multiplication** if the the natural associative key agreement scheme $k^{[A,B,C,S]} = [k_1, k_2, k_3, k_4]$ of §2.13 has all k_i implemented by known practical algorithms.*

Diversion B.2.2. Essentially, a semigroup has practical multiplication if users have practical algorithm to multiply for the inputs appearing in associative associative key agreement.

So, the users have practical algorithms to compute four functions $[a, b] \to ab$, and $[b, c] \to bc$ and $[d, c] \mapsto d$ and $[a, e] \to ae$, where $[a, b, c, d, e] = [a, b, c, ae, bc]$ for $[a, b, c] \in A \times B \times C$.

Diversion B.2.3. For non-multiplicative semigroups, practicality of the operation applies similarly.

A semigroup S has **fully practical multiplication** if it has $[S, S, S]$-practical multiplication. In many cases of interest, S has fully practical multiplication, even though it is not always necessary.

B.2.2 Practical selection

Users of associative key agreement must select semigroup elements.

Definition B.2.2. *A semigroup S has $[A, B]$-**practical selection** for subsets $A, B \subseteq S$ if there exists practical algorithms for injective function $a : |A| \to A$ and $b : |B| \to B$.*

Diversion B.2.4. For any set A, let $|A|$ denotes both its cardinality, and the smallest ordinal number with cardinality $|A|$. When A is finite, then $|A| = \{0, 1, \dots, |A| - 1\}$, and when A is countable, then $|A| = \mathbb{N} = \{0, 1, 2, 3, \dots\}$.

Diversion B.2.5. There might be semigroups with fully practical multiplication, yet even finding elements is quite impractical. For example, let $n = pq$ be a product of primes, and define a multiplicative semigroup S with set $\{p, 2p, \dots, n - p\}$ with multiplication modulo n. Knowing n gives a practical multiplication algorithm. Being able to find an element of S leads to a factorization of n, which can be difficult in general.

Diversion B.2.6. In probabilistic form of associative key agreement, practicality can be defined relative to random variables a, b, c.

Instead of selection, the users must sample the variables.

For multiplication of random variables, the practicality can be defined using some notion of practicality for probabilistic algorithms such as average run-time. To be completed.

B.2.3 Practical representation

Many semigroups are formally defined such that the elements of the semigroup are equivalence classes. These equivalence classes can be very large set, often infinite sets.

B.2.3.1 Impractical representations

If each semigroup element representation uses extremely large amounts of information, for example 2^{60} bits, then multiplication cannot be efficient, because generating such long outputs takes too long.

Moderately large representations might admit sufficiently efficient multiplication, yet still be impractical for other reasons. For example, Alice and Charlie will need communicate semigroup element representations to each other. Communication is not free: part of the cost is proportional to the data size. A communication of 2^{40} bits may be consider impractical. So, we can regard semigroup element representations of 2^{40} bits or larger as impractical.

B.2.3.2 Epigrams

An **epigram** is an efficient, compact representation of a semigroup element.

This means the usual. Each semigroup element has at least one epigram. Distinct elements have distinct epigrams: if a' is an epigram of element a, and b' is an epigram of element b with and $a \neq b$, then $a' \neq b'$. An efficient algorithm takes inputs a' and b' and computes as output an epigram c' of the element ab, where a is the element represented by epigram a', and b is the element represented by epigram b'. The set of epigrams is usually a set of finite-length bit strings.

Actually. efficiency and compactness of the representations forming epigrams are only required for the semigroups elements that Alice and Charlie will use in associative key agreement. For example, it is possible for an infinite semigroup to have an epigrams, because Alice and Charlie will only ever use finite subsets of the infinite semigroup.

Diversion B.2.7. In associative key agreement, epigrams will be used for deliveries. The systems of epigrams used by Alice and Charlie must be mutually intelligible.

Generally, an implementer of semigroup can use any desired internal representation of semigroup elements that need to be multiplied. Generally, these representations will also be compact, and could be used for epigrams.

However, for the purpose intercommunication, a single mutually intelligible system of epigrams (an **epigraphy**) must be in place.

One of the most common examples of epigram is a member of an equivalence class. In more detail, suppose that the semigroup elements represented by equivalence classes (sets, possibly infinite, or impractically large). An epigram for this semigroup element is any member of the equivalence class. For this to be considered an epigram, the equivalence relation defining the equivalence class must be efficiently implementable.

Diversion B.2.8. For an example, consider the integers modulo 7 under multiplication. Mathematically, each semigroup element is an equivalence class, which is an infinite set of integers. For example, one semigroup element is the equivalence class

$$\{\ldots, -10, -3, 4, 11, 18, 25, \ldots\}$$

, consisting of integers congruent to 4 modulo 7.

As suggested, any member of this set could be used as an epigram for the class, for example, -3 or 4 or 11.

Diversion B.2.9. For another example, consider an elliptic curve group defined by a projective cubic curve. Each semigroup element is a point in the projective plane. Mathematically, each point in the projective plane is a line through the origin in an affine 3-space.

A common epigram for a projective point, is a nonzero point on the affine line (through zero). These affine points are represented by their coordinates, but in writing them down, one often separates the coordinates by colons instead of commas, as a helpful mnemonic that the ratio between the coordinates is what defines the projective point. So, for example, $[1 : 2 : 3]$ and $[-5 : -10 : -15]$ are two epigrams for the same projective point.

B.2.3.3 Monograms

Some semigroup epigram systems have the defect that each semigroup element has multiple different epigrams. We call this situation **epigram ambiguity**.

Epigram ambiguity is problematic for keys. It prevents Alice and Charlie form agreeing on the same key.

Diversion B.2.10. Other problems arising from epigram ambiguity arise from the extra information necessary to represent to accommodate the ambiguity in a epigram, such as Alice's delivery, include the following:

- Ambiguous epigrams are not optimally compressed representations of semigroup elements.
- The choice of epigram might become an information side-channel: revealing information about Alice's secret a.

Any process that resolves epigram ambiguity is called a **monography**. To each semigroup element, a monography assigns one epigram, called the **monogram** of the element. If a semigroup has a practical monography, then we say that the semigroup is **monographic**.

A practical semigroup must be monographic.

The monography process for finding a monogram might be more costly than merely just finding any epigram. So, Alice and Charlie might opt to use monography only where necessary, such as for keys.

Diversion B.2.11. Consider again the integers modulo 7 under multiplication, and the semigroup element which is formally the equivalence class. $\{\ldots, -10, -3, 4, 11, 18, \ldots\}$.

The most common monography for this (and similar) is to select the monogram is as the least non-negative integer epigram, which in this case is 4. A less common monography is to use the integer closest to zero, which in this case is -3.

Diversion B.2.12. Monograms are also called **canonical representations** or **normal forms**, and probably many other names.

This report use the term **monogram** to emphasize non-preference, whereas previous terms sometimes imply specific monographies for certain semigroups.

Diversion B.2.13. An epigram system may have an efficient algorithm to test whether two epigrams represent the same element. In this case, we say that the epigrams are **equalizable**.

In a semigroup with monography, the epigrams are equalizable by computing the monogram for each epigram and the testing equality of the two resulting monograms.

However, in many cases, there might exist an algorithm for testing equality of epigrams that is faster than computing monograms.

Diversion B.2.14. Mathematically, a monography on a semigroup S can be considered as an isomorphism from the original semigroup S to the set of monograms. A reason to make the distinction between these two isomorphic semigroups is that algorithms in the original semigroup, such as those using epigrams, could potentially be much faster than those in the monographic image of the semigroup.

Diversion B.2.15. Returning to the elliptic curve group defined over a cubic projective curve, a common monogram is to map the projective point to an affine point, often by setting one of the coordinates to value 1. The cost of the monography is then generally a field inversion, which is often considerably more costly than a semigroup operation.

In the case of elliptic curve cryptography, it is common to use monograms for communication deliveries, because the ambiguity epigrams is sometimes a source of side-channel leakage.

Diversion B.2.16. More generally, cryptography has rather low tolerance for ambiguity in data representation.

For example, consider hashing text (a password or a message to be signed). Unfortunately, text, and even integers, is notorious for having multiple representations as bit strings. Even simple roman-alphabet text can be represented ASCII or EBCDIC. Lines of text can be separated in different ways, via LF, CR, or CRLF. International text has many encodings, but even Unicode has multiple representations, such as UTF-8 or UTF-16. Integers can be represented in big-endian or little-endian form. Even ASN.1, permits multiple encodings, such BER and PER. (Though ASN.1 has two encodings, DER and CER intended to provide monograms.)

Diversion B.2.17. Semigroups are sometimes described using presentations, given by generators and relators. The Knuth–Bendix algorithm seems to be applicable to finding a monography.

B.3 Powers and logarithms

Powering, or exponentiation, generalizes to any multiplicative semigroup.

Definition B.3.1. *A **power** a^n of an element a in a multiplicative semigroup*

S to an positive integer **exponent** *is n is the product:*

$$a^n = \underbrace{aa\dots a}_{n \text{ copies of } a} \tag{B.3.1}$$

Diversion B.3.1. Inductively, $a^1 = a$ and $a^{n+1} = a^n a$.

Diversion B.3.2. In general semigroups, the power a^n is only defined for positive integers n. So, in general, familiar expressions of standard (real number) arithmetic, such as a^0, a^{-1}, $a^{1/2}$ are undefined.

In multiplicative semigroups, some elements a might have a special kind of inverse called co-mutual inverse. In this case, we write a^{-1} for co-mutual inverse, and $a^{-n} = (a^{-1})^n$ for the powers of a^{-1}, and $a^0 = aa^{-1}$. In this report, the notation a^n with n a non-positive integer is only to be used in this situation, if a has a co-mutual inverse.

Diversion B.3.3. The function $n \mapsto a^n$ could be called **exponentiation**, while the function $a \mapsto a^n$ could be called n^{th}-degree **powering**. The binary operation $[a, n] \mapsto a^n$ could be called either **exponentiation** or **powering**, without much ambiguity.

Powering in semigroups shares some of the familiar rules of standard arithmetic, such as

$$a^m a^n = a^{m+n},$$
$$(a^m)^n = a^{mn},$$

where the addition and multiplication in the exponents on the right side of the equations are the standard operations for positive integers.

Diversion B.3.4. In some semigroups, powering or exponentiation is defined for the underlying set has

B.3.1 Scalar multiplication

In an additive semigroup, the operation analogous to powering is sometimes called **scalar multiplication** and is written:

$$na = \underbrace{a + a + \cdots + a}_{n \text{ copies of } a}, \tag{B.3.2}$$

where clear from context.

In some cases, one would write an instead of na. In some cases, the notations na and an are unsuitable, and a notation such as $[n]a$ or $\{n\}a$ can be used for extra clarity.

See later sections.

B.3.2 Aside: exportability

Powering allows us to define below **exportability**, which generalizes field characteristic, in the sense that the exportability of the additive semigroup of field equals the field's characteristic.

A few preliminary definitions are presented first.

Definition B.3.2. *An* ***exportable*** *subset B of a multiplicative semigroup is a set that does not contain all powers of all its elements.*

Diversion B.3.5. Exportable subsets of additive semigroup are defined using scalar multiples instead of powers.

For each exportable set, we define the following number.

Definition B.3.3. *The* ***exporter*** *e of an exportable subset B is the smallest positive such that*

$$a^e \notin B \tag{B.3.3}$$

for some $a \in B$.

Diversion B.3.6. Exporters in additive semigroups are defined used scalar multiplication instead of powering.

Lemma B.3.1. *Exporters are prime.*

Proof. Let e be the exporter of exportable set B. Let $a \in B$ with $a^e \notin B$. (Such a exists by definition of exporter.)

Suppose that e is not a prime.

First, $e \neq 1$, because $a^1 \in B$ for all $a \in B$. So, $e > 1$, which means that there exist positive integer $f, g < e$ with $e = fg$.

Let $u = a^f$. Since $f < e$, we must have $a^f \in B$, and thus $u \in B$. Since $g < e$, we must $u^g \in B$, so $a^e \in B$, since $a^e = a^{fg} = (a^f)^g = u^g$. This is a contradiction, the condition $a^e \notin B$.

Therefore e must be prime. □

Definition B.3.4. *The* ***exportability*** *of a semigroup S is an ordinal number that is the least upper bound of all exporters of exportable subsets of S.*

Diversion B.3.7. The least upper bound definition implies the following special cases.

- If there are no exporters, then exportability is 0.
- If exporters can be arbitrarily large, then exportability is ω (the smallest infinite ordinal).

- Otherwise, the exportability is a prime number.

Diversion B.3.8. If F_p is a finite field of size p, then its additive semigroup F_p^+ has exportability p.

To see this, first consider $B = \{1, \dots, p-1\}$. The set B is exportable because the scalar multiple $pa = 0 \notin B$, for any $a \in B$. The exporter of B is p, because if positive integer $e < p$, then $ea \in B$ for all $a \in B$. Therefore, the exportability of F^+ is at least p.

Consider any exportable subset B of F^+. Let e be the exporter of B, and suppose that $a \in B$ but $ea \notin B$. Suppose that $e > p$. Then $ea = (e-p)a$, since addition is modulo p. But $e - p$ is a positive integer and less than e, contradicting the assumption that e is the exporter of B. So $e \leq p$.

Therefore, p is the exportability of F_p^+.

Diversion B.3.9. The exportability of a semigroup is zero if only and only if all elements are idempotents, meaning $a^2 = a$, as show below.

If the exportability is 0, then there are no exporters and no exportable sets. In particular, set $B = \{a\}$ is not exportable for any a, so $a^2 \in B$ meaning $a^2 = a$.

Conversely, if all a are idempotent, then $a = a^2 = a^3 = \dots$, so $a^e = a$ for all a. If is any set B, then $a^e \in B$, for all $a \in B$, so B is not exportable. There exists no supporters, so 0 is an upper bound on the set of exporters, and exportability is zero.

Diversion B.3.10. To be revised.

Exportability seems to share some properties with cardinality. If S is a finite semigroup of size s, then the exportability of S is at most s. If T is a subsemigroup of S with exportability t, then S has exportability at least t. If S has an element of infinite order, then S has exportability of ω.

Maybe also: if T is an image of S (under a surjective semigroup morphism), the S has exportability at least that of T?

Diversion B.3.11. Consider the additive cyclic group C_n^+ of size n, which is integers under addition modulo n. Also allow for $n = 0$ for the empty semigroup. The exportability e_n of C_n^+ for small n seems to be as indicated below:

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
e_n	0	0	2	3	3	5	5	7	5	3	5	11	7	13	7	7	5	17

If these numbers are correct, the fact that the sequence above has not yet appeared in the Online Encyclopedia of Integers Sequences (OEIS), suggests that the notion of exportability is not yet widely known. (If the notion is known, then it likely has a completely different name.)

For all n, we have $e_n \leq n$. For prime n, we have $e_n = n$, while it seems that $e_{3^m} = 3$ for all m, so the values of e_n seem to fluctuate between 3 and n, hitting both top and bottom for arbitrarily large n.

Diversion B.3.12. Exportability may be most interesting when applied to medial magmas, where it has some similar properties.

Diversion B.3.13. A set B is non-exportable if and only if it is power-closed, meaning closed under the powering operation.

Power-closed sets define the **power topology** on S by taking the topologically closed sets to be the power-closed sets. This defines a topology because power-closed sets meet the topological axioms for closed sets: $\{\}$ and S are power-closed, the finite union of power-closed sets is close, and the arbitrary intersection of power-closed sets is power-closed.

Typical topological questions can then be asked. For example, in the power topology, the additive semigroup $\mathbb{Q}^+_>$ is connected, but $\mathbb{Q}^+_{\geq 0}$ is disconnected.

Moreover, an arbitrary union of power-closed sets is also power-closed is also power-closed. Therefore, the power-closed sets can form the open sets of another topology, the **open-power topology**.

A set B is power-closed if and only if it is the union of subsemigroups. Therefore, in the open-power topology, subsemigroups form a basis for the open sets.

B.3.3 Powering sets

Powering generalizes to powers of sets. Probably the most important generalization is the following.

Definition B.3.5. *A* ***power*** B^n *of a subset* B *of a multiplicative semigroup to positive integer* n *is the set:*

$$B^n = \{b_1 b_2 \dots b_n : b_i \in B\} \tag{B.3.4}$$

Diversion B.3.14. The set 2^S of subsets of S forms a semigroup under multiplication of sets, and B^n is the power of B in semigroup 2^S.

Diversion B.3.15. In additive semigroups, we can write nB instead of B^n, if clear from context.

Diversion B.3.16. We generally do not attempt to define B^n for value of n that are not positive integers.

A narrower way to define a power of a set is sometimes useful to consider.

Definition B.3.6. *An* ***elemental power*** $B^{[n]}$ *of a subset* B *of a semigroup to positive integer* n *is the set:*

$$B^{[n]} = \{b^n : b \in B\}. \tag{B.3.5}$$

Diversion B.3.17. In an additive semigroup, write $[n]B$ instead of $B^{[n]}$.

Diversion B.3.18. If necessary to avoid confusion with elemental powers, call B^n a **generative** or **factorable** power.

The two types of powers of B are always related by

$$B^{[n]} \subseteq B^n. \tag{B.3.6}$$

If $|B| = 1$, then $B^{[n]} = B^n$.

Diversion B.3.19. Some of the usual powering rules from standard arithmetic also apply to set-powers, such as:

$$\begin{aligned}(B^m)^n &= B^{mn},\\ B^m B^n &= B^{m+n},\\ (B^{[m]})^{[n]} &= B^{[mn]}.\end{aligned}$$

In commutative (or medial) semigroups, the two operations also commute as in

$$(B^m)^{[n]} = (B^{[n]})^m.$$

Diversion B.3.20. A set B is exportable if $B^{[e]} \nsubseteq B$ for some positive integer e.

Diversion B.3.21. A set B has $B^e \nsubseteq B$ if and only if B is not a subsemigroup. If B is not subsemigroup, then the least e such that $B^e \nsubseteq B$ is always 2. In other words, the idea of exportability based on generative powering, has a more trivial result

It is also possible to defines powers in which the exponents or degrees are sets of positive integers, as in:

$$B^N = \{b_1 \dots b_n : b_i \in B,\ n \in N\}, \tag{B.3.7}$$

$$B^{[N]} = \{b^n : b \in B,\ n \in N\}. \tag{B.3.8}$$

B.3.4 Powering arrays (cyclically)

The following notation, generalizing powering, might sometimes be convenient:

$$[a_0, \dots, a_{m-1}]^{[e_0,\dots,e_{t-1}]} = \prod_{i=0}^{t-1} a_{i \bmod m}^{e_i}. \tag{B.3.9}$$

This is always defined in a semigroup if $t, m > 0$ and all $e_i > 0$. Letting $a = [a_0, \dots, a_{m-1}]$ and $e = [e_0, \dots, e_{t-1}]$, we can write this as a^e, and call it the **cyclic power** of a to the e, because the indices of the a_i are treated cyclically modulo m.

Diversion B.3.22. As an example of cyclic powering,

$$[a, b]^{[1,2,1,1,2,1]} = abbabaab.$$

Diversion B.3.23. Cyclic powering is most useful when $m = 2$, for several reasons.

For example, $\langle a, b\rangle = \{[a, b]^e, [b, a]^e : e \in \mathbb{P}^*\}$ (where e ranges over all arrays of positive integers).

Diversion B.3.24. Cyclic powering obeys a calculus similar to the calculus for the usual powering.

For example, $[a, b]^e[b, a]^f = [a, b]^g$, where g is easily determine by concatenating the arrays e and f except for possibly adding the last entry to e to the first of f if e has an even length, as in

$$\begin{aligned} [a,b]^{[1,1]}[b,a]^{[1,1]} &= [a,b]^{[1,1+1,1]} = [a,b]^{[1,2,1]} \\ [a,b]^{[1,2,1]}[b,a]^{[1,2,1]} &= [a,b]^{[1,2,1,1,2,1]} \\ [a,b]^{[1,2,1,1,2,1]}[b,a]^{[1,2,1,1,2,1]} &= [a,b]^{[1,2,1,1,2,1+1,2,1,1,2,1]} = [a,b]^{[1,2,1,1,2,2,2,1,1,2,1]}. \end{aligned}$$

Recursive strategies for computing co-multiples of a and b, or for cross-multiplying a and b, can be described using cyclic powers similar to those above.

Diversion B.3.25. We can extend the definition of cyclic powering to allow some zero (or negative) entries in e in two different ways.

- If all the factors $a_i^{e_i}$ are well-defined then the definition can be used as is. We usually only define b^0 as $b^0 = bb^{-1}$ and define b^{-1} if b^{-1} is a co-mutual inverse of b (see §B.7.4).
- If all $e_i > 0$, then we can treat any $e_i = 0$ as being absent from the product, effectively being equivalent to 1 (even if $1 \notin S$). As long as one entry is nonzero, then this is defined.

These two extensions of cyclic powering might not agree, in some cases. Consider $[a, b]^{[1,0]}$. The first extension gives ab^0 and the second gives a, which might not be equal.

Diversion B.3.26. A more general notation is:

$$[a_1, \ldots, a_m]^{[e_1,\ldots,e_t]}_{[j_1,\ldots,j_t]} = \prod_{i=1}^{t} a_{j_i}^{e_i}.$$

Cyclic powering is a special case of this notation, with

$$[a_1, \ldots, a_m]^{[e_1,\ldots,e_t]} = [a_1, \ldots, a_m]^{[e_1,\ldots,e_t]}_{[1,\ldots,m,1,\ldots,m,\ldots]}$$

where the subscripted array on the right cycles through $[1, \ldots, m]$, repeating as necessary.

B.3.5 Magma powering and products

To be revised.

Occasionally, a generalization powering in a non-associative magma is needed. In repeated multiplication is ambiguous, one must specify the order of multiplications: in **left powering** multiply on the left first, which is defined inductively by $a^1 = a$ and $a^{n+1} = (a^n)a$; **right powering** is defined by $a^1 = a$ and $a^{n+1} = a(a^n)$. Default to left powering, unless otherwise noted.

In general, powering in a magma does not obey the familiar rules. However, in a **medial** magma, meaning a magma obeying the axiom:

$$(ab)(cd) = (ac)(bd), \tag{B.3.10}$$

powering obeys the rule $(a^m)^n = a^{mn}$, which is a special case of a more general result about medial magmas: changing the order of products, defined below.

Definition B.3.7. *In multiplicative magma M, define the left product:*

$$\prod : M^n \to M : [a_1, \dots, a_n] \mapsto \prod_{i=1}^{n} a_i \tag{B.3.11}$$

inductively on n by the rule:

$$\prod_{i=1}^{1} a_i = a_1 \tag{B.3.12}$$

$$\prod_{i=1}^{n+1} a_i = \left(\prod_{i=1}^{n} a_i\right) a_n. \tag{B.3.13}$$

Diversion B.3.27. The right product can be defined similarly, but then some notation would be needed to distinguish the two products.

The left power, for example, is related to the left product by the rule

$$a^n = \prod_{i=1}^{n} a = \prod[\underbrace{a, a, \dots, a}_{n \text{ copies of } a}]$$

It is well-known (and widely known?) that a medial magma, the order of products can be swapped:

Lemma B.3.2. *For any matrix of value $a_{i,j}$ in a medial magma,*

$$\prod_{i=1}^{m} \prod_{j=1}^{n} a_{i,j} = \prod_{j=1}^{n} \prod_{i=1}^{m} a_{i,j}. \tag{B.3.14}$$

Proof. Induction on pairs $[m, n]$.

If $1 \in \{m, n\}$ and the inner product on one side becomes an identity function, while the outer product on the other side vanishes. So both sides become a single product, the same single product.

Otherwise, $m - 1, n - 1 \geq 1$, and we assume by induction that the result holds for pairs $[m - 1, n]$, $[m, n - 1]$ and $[m - 1, n - 1]$. Calculating,

$$\begin{aligned}
\prod_{i=1}^{m}\prod_{j=1}^{n} a_{i,j} &= \left(\prod_{i=1}^{m-1}\prod_{j=1}^{n} a_{i,j}\right)\prod_{j=1}^{n} a_{m,j} \\
&= \left(\prod_{j=1}^{n}\prod_{i=1}^{m-1} a_{i,j}\right)\prod_{j=1}^{n} a_{m,j} \\
&= \left(\left(\prod_{j=1}^{n-1}\prod_{i=1}^{m-1} a_{i,j}\right)\prod_{i=1}^{m-1} a_{i,n}\right)\prod_{j=1}^{n} a_{m,j} \\
&= \left(\left(\prod_{j=1}^{n-1}\prod_{i=1}^{m-1} a_{i,j}\right)\left(\prod_{i=1}^{m-1} a_{i,n}\right)\right)\left(\left(\prod_{j=1}^{n-1} a_{m,j}\right)a_{m,n}\right) \\
&= \left(\left(\prod_{j=1}^{n-1}\prod_{i=1}^{m-1} a_{i,j}\right)\left(\prod_{j=1}^{n-1} a_{m,j}\right)\right)\left(\left(\prod_{i=1}^{m-1} a_{i,n}\right)a_{m,n}\right) \\
&= \left(\left(\prod_{j=1}^{n-1}\prod_{i=1}^{m-1} a_{i,j}\right)\prod_{j=1}^{n-1} a_{m,j}\right)\prod_{i=1}^{m} a_{i,n} \\
&= \left(\left(\prod_{i=1}^{m-1}\prod_{j=1}^{n-1} a_{i,j}\right)\prod_{j=1}^{n-1} a_{m,j}\right)\prod_{i=1}^{m} a_{i,n} \\
&= \left(\prod_{i=1}^{m}\prod_{j=1}^{n-1} a_{i,j}\right)\prod_{i=1}^{m} a_{i,n} \\
&= \left(\prod_{j=1}^{n-1}\prod_{i=1}^{m} a_{i,j}\right)\prod_{i=1}^{m} a_{i,n} \\
&= \prod_{j=1}^{n}\prod_{i=1}^{m} a_{i,j},
\end{aligned}$$

where the equation between the two longest expressions uses the medial property of the magma, the equations between expressions of different lengths uses the definition of left product, and the remaining equations use inductive swapping of order of products. □

Diversion B.3.28. Exportable subsets, exporters and exportability can be defined for magmas, by using left powers. In medial magmas, exporters are prime, because the proof really only involves the power-composition law.

B.3.6 Discrete logarithms

Logarithms, a kind of opposite of exponentiation, are sometimes useful in semigroup algorithms.

Definition B.3.8. *A* ***discrete logarithm*** *of d to base b (in a multiplicative semigroup) is a positive integer n such that:*

$$d^n = b. \tag{B.3.15}$$

The (smallest) discrete logarithm of d to b is defined as:

$$\log_b(d) = \min\{n : d^n = b, n \in \{1, 2, 3, \dots\}\}. \tag{B.3.16}$$

Diversion B.3.29. When clear from context, we call $\log_d(b)$ the discrete logarithm (dropping the term smallest). If there are no discrete logarithms, then the definitions would suggest $\log_d(b) = \infty$, but generally, we instead say that the discrete logarithm is undefined.

Diversion B.3.30. The discrete logarithm can be considered as partial function $\log : S^2 \to \mathbb{P}^+ : [b, d] \mapsto \log_b(d)$, which is only defined for the domain of pairs $[b, d]$ with $d \in \langle b \rangle$, which is only a subset of S^2. The discrete logarithm maps these pairs into an additive semigroup $\mathbb{P}^+$ of positive integers.

For each fixed b: the discrete logarithm defines a function $\log_b : \langle b \rangle \to \mathbb{P}^+ : b^n \mapsto n$. If $\langle b \rangle$ is infinite, then this function is an isomorphism. If $\langle b \rangle$ is finite, the discrete logarithm can be used to induce a semigroup congruence on $\mathbb{P}^+$.

Diversion B.3.31. In additive semigroups, and more generally any non-multiplicative semigroups, define discrete logarithms in the same way.

The discrete logarithm terminology and notation potentially clashes with additive notation and terminology, although in the case of elliptic curve cryptography, this clashes has seemed quite manageable.

If the clash somehow becomes too problematic, then, as a last resort, alternative terminology could be used. In the additive setting, one could say (discrete) scalar division instead of discrete logarithm, and perhaps write $d/_+b$, where the subscript $+$ serves to distinguish proper division if there is a multiplication operation acting on the same set of S^+.

Diversion B.3.32. In a magma, discrete logarithms can be defined, or qualified, by selecting the type of powering used, such as left powering.

Diversion B.3.33. Some semigroups S, have a type of logarithm that is more general than the discrete logarithm. For example, the semigroup $\mathbb{R}_{>0}$ positive real numbers under multiplication, have the **continuous logarithm**, a function $\log : \mathbb{R}^2_> \to \mathbb{R}^+$. obeying various rules, some shared with the discrete logarithm, such as $\log_b(de) = \log_b(d) + \log_b(e)$.

The discrete logarithm, where defined, equals the the general logarithms. Conversely, wherever the more general logarithm takes positive integer value, they the more general logarithm equals the discrete logarithm.

B.3.6.1 Aside: logarithms of sets

To be revised.

This subsection digresses from division, by considering the generalization discrete logarithms to subsets of semigroups.

Discrete logarithms of subsets may reveal something about the general structure of a semigroup, even if they do not directly lead to division algorithms.

First this section addresses various directions along which one can generalize discrete logarithms to sets.

Matching sets This section generalizes the equation to $b^e = d$ from an equality to various possible relations between sets.

Let C and D be sets (where $C = B^e$ or $C = B^{[e]}$, in the intended applications). We consider four ways in which C and D be considered a match:

$$C = D, \tag{B.3.17}$$

$$C \subseteq D, \tag{B.3.18}$$

$$C \supseteq D, \tag{B.3.19}$$

$$C \cap D \neq \varnothing. \tag{B.3.20}$$

We could potentially name these matching schemes, **exacting**, **embedding**, **covering**, and **meeting**. When $C = \{c\}$ and $D = \{d\}$, then these four matching schemes are identical.

We could then name set-logarithm according to the type of matching scheme.

As a convenience to discuss all four possibilities in parallel, such as when defining the set-logarithm, we write $C \sim D$, to mean that $\sim$ is to be replaced later by one of these matching schemes.

For the most part, we default to the meeting scheme, so the set-logarithms are determine by $B^e \cap D \neq \varnothing$.

Logarithm sets and discrete logarithms of sets The **logarithm-set** of D to the base B is written and defined as

$$\log_B(D) = \{e : B^e \sim D\}.$$

The logarithm-set is elemental or factorable depend on whether the power means $B^e = B^{[e]}$ or $B^e = B^{\langle e\rangle}$. The logarithm-set is exacting, embedding, covering or meeting depending on the meaning of $\sim$. We default to meeting.

Each $e \in \log_B(D)$ can be called **discrete logarithm** of D to base B. A positive integer e is a discrete logarithm of $\{d\}$ to base $\{b\}$ if and only if e is a discrete logarithm of d to base b. Therefore, a generalization has been achieved.

For convenience, write $\log_B(d) = \log_B(\{d\})$ and $\log_b(D) = \log_{\{b\}}(D)$.

Recall that $\langle B\rangle$ denotes the smallest subsemigroup of S containing B. Then $d \in \langle B\rangle$ if and only $\log^\circ_B(d) \neq \varnothing$.

The logarithm-set has the following super-additive property:

$$\log_B(dg) \supseteq \log^\circ_B(d) + \log^\circ_B(g),$$

since if $e \in \log_B(d)$ and $f \in \log_B(g)$, then $d = b_1 \dots b_e$ and $g = b_{e+1} \dots b_{e+f}$ for some $b_i \in B$, so $dg = b_1 b_2 \dots b_{e+f}$. We also have bound

$$B \subseteq C \implies \log_B(d) \subseteq \log_C(d).$$

Canonical discrete logarithms A logarithm-set can contain many discrete logarithms, but often one particular discrete logarithm is most interesting and canonical, and perhaps deserves to be the called **the** discrete logarithm.

Often, the most important discrete logarithm is the **minimum-logarithm** written and defined:

$$\log^{\min}_B(D) = \min(\log_B(D)), \tag{B.3.21}$$

which is only defined if $\log_B(D) \neq \varnothing$. If we refer to **the** discrete logarithm, we by default refer to the minimum-logarithm. Similarly, if context, such as formula, implies that $\log_B(D)$ is a positive integer rather than a set, then, by default $\log_B(D)$ is just an abbreviation for $\log^{\min}_B(D)$, with the superscript min being omitted and implicitly meant.

The **infimum-logarithm** $\log^{\min}_B(D)$ is defined just like minimum-logarithm, except that if the logarithm-set is empty ($\log_B(D) = \varnothing$), then we define

$\log_B^{\inf}(D) = +\infty$. In this latter case, the infimum-logarithm is not truly a discrete logarithm.

The **supremum-logarithm** is defined as $\log_B^{\sup}(D) = \sup(\log_B(d))$, with the convention that $\sup(\varnothing) = -\infty$ and $\sup(E) = +\infty$ if E is not bounded above.

The super-additivity of the logarithm-set results in numerical bounds for these operations:

$$\log_B^{\inf}(dg) \leq \log_B^{\inf}(d) + \log_B^{\inf}(g),$$
$$\log_B^{\sup}(dg) \geq \log_B^{\sup}(d) + \log_B^{\sup}(g).$$

Case study: the sup-log The supremum-logarithm is perhaps most interesting when $B = S$ and S has no identity element (so S is non-unital). For brevity, we call supremum-logarithm to the base S the **sup-log**, or **generation**, of $d \in S$. The sup-log is never $-\infty$.

Diversion B.3.34. If S is unital, then the supremum-logarithm to base S is always $+\infty$, because $d = 1^e d$ for all positive integers e.

If $d = d^{p+1}$, for some positive integer p (which is to say that d is periodic or has torsion, per Def. B.4.5), then the sup-log of d is necessarily ∞.

In associative key agreement, the key has sup-log at least 3, the deliveries at least 2.

Diversion B.3.35. The notion of sup-log was first learned by the report author in reading Distler's work [Dis10] on nilpotent semigroups. This report uses different terminology, but the same notion. The following interesting observations are probably also from Distler's work:

- An element d with sup-log $e < \infty$ can factored as elements $d = d_1 \dots d_e$ where each d_i has sup-log 1.
- The elements with sup-log $\geq e$ form a subsemigroup, namely S^e if $e < \infty$. Let S^∞ indicates the subsemigroup of all elements with sup-log equal to ∞.
- If d has finite sup-log e, then $\log_{S^e}^{\sup}(d) = 1$. If d has sup-log ∞ and $\log_{S^\infty}^{\sup}(d)$ is finite, then it seems that such recursive sup-logs might be useful to understanding the place of d in S.

Case study: elemental logarithms and exportability To be revised. Some of material originally was moved earlier.

The elemental minimum-logarithm $\log_B^{\min}(D)$ is also the minimum of the discrete logarithms $\log_b(d)$ for $b \in B$ and $d \in D$.

The special case where D is the complement $S - B$ of B says something interesting about B.

The (elemental) exporter is the elemental, meeting, minimum-logarithm of the complement $S - B$ of B, to the base B.

To be completed.

B.4 Subsemigroups

Definition B.4.1. *A* ***subsemigroup*** *of semigroup S (or a* ***submagma*** *of a magma S) is any subset $T \subseteq S$ closed under the binary operation of the semigroup (or magma). When clear from context, write*

$$T \leq S$$

to mean that T is a subsemigroup of S.

Diversion B.4.1. Another way to characterize subsemigroup of S is a subset T that is also a semigroup (where the binary operation on elements of T is the binary operation of S).

Diversion B.4.2. Another way to characterize subsemigroups (in a multiplicative semigroup) is with set-multiplication (§B.1.4). We have $T \leq S$ if and only if $\{ab : a, b \in T\} = T^2 \subseteq T \subseteq S$ (provided that the multiplication used for elements of T is the same as would used in S).

Similarly, in an additive semigroups S^+, we have $T^+ \leq S^+$ if and only if $2T^+ \subseteq T^+ \subseteq S^+$.

Diversion B.4.3. Every non-empty semigroup S (or magma) has at least two distinct subsemigroups (or submagams): $\{\}$ and S.

(In an empty semigroup, these two subsemigroups are the same.)

Diversion B.4.4. Forming subsemigroups is an important method to construct new semigroups form old semigroups. Several examples of such constructions are given in the appendices.

Diversion B.4.5. Subsemigroups of groups are not necessarily subgroups. For example, positive integers forms a subsemigroup of the group of integers, but positive integers do not from a subgroup.

B.4.1 Generated sets

A special case of subsemigroups is common, general and basic enough to mention here.

Definition B.4.2. *The set* ***generated*** *by a subset $B \subset S$ of a multiplicative semigroup is:*

$$\langle B \rangle = \{ b_1 b_2 \dots b_n : b_i \in B, n \in \{1, 2, 3, \dots\}\}. \tag{B.4.1}$$

For a finite set $B = \{b_1, \dots, b_m\}$, write $\langle B \rangle = \langle b_1, \dots, b_m \rangle$.

Diversion B.4.6. In group theory, a different meaning of generated set is used, by allowing inverses, and this could lead to confusing clashes of notation. This report uses the notation above, but outside the report, the group theory notation may have precedence. So, outside this report, an explanation of the exception distinction for semigroup-generated and perhaps a notation such as $\langle B \rangle^\times$ may be needed.

Diversion B.4.7. Set-powering (§B.3.3) can also be used to describe generated sets:

$$\langle B \rangle = B \cup B^2 \cup B^3 \cup \dots = \bigcup_{n \geq 1} B^n \tag{B.4.2}$$

Lemma B.4.1. *A generated set is subsemigroup (or submagma):*

$$\langle B \rangle \leq S.$$

Proof. Suppose that $a, c \in \langle B \rangle$. Then $a = b_1 b_2 \dots b_m$ and $c = b_{m+1} \dots b_{m+n}$ for some positive integers m, n, and some elements $b_i \in B$. Then $ac = b_1 b_2 \dots b_{m+n} \in \langle B \rangle$. So, $\langle B \rangle$ is closed under multiplication. □

Diversion B.4.8. The intersection of an arbitrary family of subsemigroups (or submagma) of S is a subsemigroup (submagma) of S. So,

$$\langle B \rangle = \bigcap_{T : B \subseteq T \leq S} T. \tag{B.4.3}$$

This formula is more theoretical than (B.4.1) or (B.4.2), because it runs over an index T of subsemigroups which is potentially quite large. In some cases, it is an intersection over uncountably many different$T$.

By comparison, in the definition of $\langle B \rangle$, the index n runs over the countable set of positive integers, and depends in no way on the elements of S that are outside of B.

For these reasons, we have taken the generated set as the definition of $\langle B \rangle$ rather than the intersection of containing subsemigroups.

Diversion B.4.9. In practice, a subsemigroup T is most easily described $T = \langle B \rangle$. In this case, B is called a set of **generators** for T, or a **generator set** for T.

Every subsemigroup has itself for a set of generators: $T = \langle T \rangle$, but such a description is not useful.

Using generator sets to describe subsemigroups is most useful when the generator sets are small, such a finite, or at least much smaller than the subsemigroup itself.

Given a description of a (sub)semigroup, it may be difficult to find small generator sets.

Diversion B.4.10. A given subsemigroup potentially has many different generator sets, so the generator set does not provide a unique method to describe to a subsemigroup.

Given two subsets $B, C \subseteq S$ of a semigroup, it may be difficult to tell if $\langle B \rangle = \langle C \rangle$.

Diversion B.4.11. Some semigroups S have minimal set G of generators for S. A minimal set G of generators is such that $\langle G \rangle = S$, but $\langle H \rangle \neq S$ for all $H \subset G$.

Diversion B.4.12. For a simple example of a semigroup without a finite generator set, let $S^+ = \{[x,y] | x, y \in \mathbb{Z}; 0 \leq y \leq x^2; 0 \leq x \leq y^2\}$, with vector addition, defined as usual: $[x,y] + [x',y'] = [x+x', y+y']$.

In fact, this S^+ seems to have a unique minimal generating set with elements $[0,0]$, $[1,1]$, and $[n,m]$ and $[m,n]$ with $n \geq 1$ and $(n-1)^2 + 2 \leq m \leq n^2$.

B.4.2 The complete lattice of subsemigroups

The subsemigroups of S form a **complete lattice**.

Recall that a complete lattice is a partial order in which every subset has a unique infimum (meet) and a unique supremum (join).

The unique infimum of a set of subsemigroups is their intersection. The unique supremum of a set of subsemigroups is the set generated by the union of the subsemigroups (so all products with factors in the union of the set of subsemigroups).

The **join-irreducible** subsemigroups are those generated by a single element, as in $\langle b \rangle$ for $b \in S$, because these are not the join of different subsemigroups. The minimal non-empty semigroups of S are the subsemigroups generated by idempotent elements.

Dually, a question of potential interest are the **maximal** proper subsemigroup of S, and the meet-irreducible subsemigroups. Intuition from geometry suggests that some meet-irreducible subsemigroups are defined by a single equation.

B.4.3 Order, period, and torsion

Definition B.4.3. *The **order** of an element a of a semigroup S is*

$$\operatorname{ord}(a) = |\langle a \rangle| = \#\{a, a^2, a^3, \dots\}, \qquad \text{(B.4.4)}$$

in other words, the cardinality of the smallest subsemigroup containing a.

Strictly speaking, if a has infinite order, then its order is $\aleph_0$, the smallest countable cardinal number. But the order can never be a larger infinite cardinal, so writing ∞ for the order is usually clear enough.

Diversion B.4.13. In some contexts, it is convenient or clearer to use other notations for order, such as $|a|$, or $\|a\|$ or $\rho(a)$.

Diversion B.4.14. In non-multiplicative semigroups, order is defined by suitable translation of notations.

If an additive and multiplicative semigroup have overlapping sets, the we can refer to the **additive order** and **multiplicative order** of elements. The term **order** alone defaults to multiplicative order.

Notationally, write $\mathrm{ord}^+(a)$ for the additive order, if needed to avoid confusion with multiplicative order $\mathrm{ord}(a)$.

Diversion B.4.15. Order of element a can also be defined in a magma as $\mathrm{ord}(a) = |\langle a\rangle|$, the cardinality of the submagma generated by a.

The order in a magma can be larger than the number of distinct (left) powers of a, because magma multiplication is non-associative, meaning that iterated powers of a are not the only elements generated by a.

Diversion B.4.16. Outside this report, the word **order** sometimes has other meanings. If necessary, then this report's meaning can be called **generated order** or **generated size**, to avoid a clash with other meanings.

The following obvious result is useful (though the proof may be sub-optimal).

Lemma B.4.2. *If a in a multiplicative semigroup has finite order, then*

$$\langle a\rangle = \{a, a^2, \ldots, a^{\mathrm{ord}(a)}\},$$

and all entries of the array $[a, a^2, \ldots, a^{\mathrm{ord}(a)}]$ are distinct.

Proof. Let $n = \mathrm{ord}(a)$, and let $A = \{a, a^2, \ldots, a^n\}$. By definition, $A \subseteq \langle a\rangle$, so A has at most n elements. If $|A| = n$, then A is as large as $\langle a\rangle$, so $A = \langle a\rangle$, and we are done.

Otherwise, A has fewer than n elements, which we aim to show leads to a contradiction. In this case, some two elements of the array $[a^1, a^2, \ldots, a^n]$ are equal, say $a^m = a^{m+p}$ for positive integers m, p with $m + p \leq n$. Then, for any non-negative integer q,

$$a^{m+p+q} = a^{m+(q \bmod p)},$$

by induction. (In more detail, let $r = q \bmod p$ and let $q = up + r$ for some non-negative integer u. Then $a^{m+p+q} = a^{m+p+up+r} = a^{m+p}a^{up+r} = a^m a^{up+r} = a^{m+up+r}$. If $u = 0$, then this is a^{m+r} as desired. Otherwise, $u \geq 1$, and by induction, this is $a^{m+p+(u-1)p+r} = a^{m+r}$ by induction on u.)

Therefore,

$$\langle a \rangle \subseteq \{a^1, \ldots, a^{m+p-1}\}.$$

Hence $\operatorname{ord}(a) \leq m + p - 1$. But $m + p - 1 < m + p \leq n = \operatorname{ord}(a)$, so we get a contraction $\operatorname{ord}(a) < \operatorname{ord}(a)$. □

Diversion B.4.17. Clearly, all elements in a finite semigroup have finite order. The converse can fail, but if all elements of S have finite order, then we could say that S is **element-locally finite**.

(Another, stronger, meaning of locally finite is that every finite set generates a finite subsemigroup.)

Finite order elements have powers that eventually repeat, meaning that the sequence of powers eventually periodic, hence the following definition.

Definition B.4.4. *A **periodicity** of a finite-order element a of a multiplicative semigroup is a positive integer p such that*

$$a^m = a^{m+p} \tag{B.4.5}$$

*for some positive integer m. The **period** $\operatorname{per}(a)$ of a it its smallest periodicity.*

Diversion B.4.18. An integer is a periodicity if only and only if it is a positive multiple of the period. The period is the greatest common divisor of all the periodicities.

Diversion B.4.19. Outside this reports, semigroup elements with period 1 are sometimes[1] called **aperiodic**.

A very special case of finite order elements are those which appear in the repeating sequence of their powers, as defined below.

Definition B.4.5. *A **torsion** element is an element a such that $a = a^{p+1}$ for some positive integer p.*

Diversion B.4.20. Although the report's term **torsion** aims to generalize the meaning of torsion in the setting of groups and rings, it very likely conflicts with precedents in semigroups theory (either different names for the same notion, or different meanings of **torsion**, or both).

This report prefers the term **torsion** is over **periodic**, partly because periodic has many other useful meanings, as periodic functions, but also because **torsion** is used more often in group and ring theory.

The term **cyclic** would also be very logical, but it is occasionally used in group theory to describe groups generated by a single element, including the infinite order group $\mathbb{Z}^+$, which clashes with our intended notion of finite order.

[1] Somewhat confusingly.

If $a \in S$ belongs to a subsemigroup T such that T is also a finite group, then a is a torsion element. The converse holds (which will be shown in the section on inverses).

An alternative characterization of torsion elements is the following.

Lemma B.4.3. *An element a is a torsion element if and only if* $\operatorname{ord}(a) = \operatorname{per}(a) < \infty$.

Proof. To be completed.

□

Diversion B.4.21. If a has finite-order, then we define the **coma** of a as its order minus its period: $\operatorname{ord}(b) - \operatorname{per}(b)$.

The coma c of a is the least non-negative integer such that $a^{c+1} = a^{c+1+p}$ for some positive integer p.

An element is a torsion element if and only if its coma is zero.

Mnemonics for terms **period** and **coma** are the allusions to comets, and to punctuation. Other names for **coma** include **index**, **height**, **depth**, **tail-length**, and so on.

Diversion B.4.22. The **idempotency degree** of a is least positive integer ω such that a^ω is idempotent, if such an integer exists. If a has period p and coma c, then

$$\omega = p\left\lceil\frac{c+1}{p}\right\rceil = (c+p) - (c \bmod p).$$

For a given idempotency degree ω, it will be true that $p|\omega$ and $c = \omega - \gamma$ for a positive integer $\gamma \leq p$. All such p and γ and c can occur.

B.4.4 Aside: local and global properties

WRONG. NEEDS MAJOR RE-WORKING!

The next abstraction tries to formalize special types of properties, either of semigroups, or of their elements, according to how the properties behave under subsemigroups.

Definition B.4.6. *A set C set of subsemigroups of a semigroup S is*

- ***local** in S if $T \leq U$ and $T \in C$ implies $U \in C$ for all $U \leq S$, and*
- ***global** in S if $T \geq U$ and $T \in C$ implies $U \in C$ for all $U \leq S$.*

For example, the set of commutative subsemigroups is global in S, and the set of subsemigroups containing specific element is local in S.

Diversion B.4.23. The notions of local and global apply equally well to sets of submagmas of a magma, and to similar properties.

If set C is both local and global in S, then C is either empty or is the set of all subsemigroups, because S and $\{\}$ are subsemigroups.

Diversion B.4.24. Outside this report, the term **local** has many other meanings, depending on the context.

The closest context may be in commutative algebra, where it refers to various related properties, including having at most one maximal ideal, and to the formation rings of fractions with denominators from a prime ideal.

B.4.4.1 Relatively local and global

A set C is **local relative** to a set D if:

$$T \leq U,\ T \in C,\ U \in D \implies U \in C. \tag{B.4.6}$$

So, in particular, a set C is local in S if and only if it is local relative to all the subsemigroups of S.

Similarly, a set C is **global relative** to a set D if:

$$T \geq U,\ T \in C,\ U \in D \implies U \in C. \tag{B.4.7}$$

B.4.4.2 Element properties

An **element property** P is a class of pairs $[S, a]$ where S is a semigroup, and a is an element. For each $[P, S, a]$, define a set $C = C_{P,S,a} = \{T : T \leq S,\ [T, a] \in P\}$, which is a set of subsemigroups of S.

An element property is **local** if C is local in S for all $[S, a]$. An element property is **global** if C is global in S for all $[S, a]$. An element property is **intrinsic** if it is both local and global.

Membership is an element property consisting of all pairs $[S, a]$ such that $a \in S$. Membership is local.

An element property P is **local relative** to element property Q if $C_{P,S,a}$ is local relative to $C_{Q,S,a}$ for all $[S, a]$. An element property P is **global relative** to element property Q if $C_{P,S,a}$ is global relative to $C_{Q,S,a}$ for all $[S, a]$. An element property P is **intrinsic relative** to element property Q if $C_{P,S,a}$ is both local relative to $C_{Q,S,a}$ and global relative to $C_{Q,S,a}$.

An element property is **relatively local** if it is local relative to membership. An element property is **relatively global** if it is global relative to membership. An element property is **relatively intrinsic** if it is intrinsic relative to membership.

Middle-invertibility, also known as regularity, is an element property consisting of all pairs $[S, b]$ such that $b \in S$ and there exists $q \in S$ such that $bqb = b$. Middle-invertibility is relatively local, but is not relatively global. For example, 2 is invertible in the rationals, but not in the integer subsemigroup.

Identity is an element property consisting of pairs $[S, e]$ such that $e \in S$ and $ea = ae = e$ for all $a \in S$. Identity is relatively global, but is not relatively local. For example, 0 is not a multiplicative identity in the integers, even though it is an identity in the subsemigroup $\{0\}$.

Idempotency is an element property consisting of pairs $[S, e]$ such that $e \in S$ and $e^2 = e$. Idempotency is relatively intrinsic. For example, both 0 and 1 are idempotent in the rationals (under multiplication), and also in every subsemigroup of the rationals (such as non-negative integers), and in every semigroup containing the rationals (such as complex analytic functions, which contains the rationals as constant functions).

Diversion B.4.25. A heuristic for a relatively local element property is that its logical formula has no universal quantifiers (over a semigroup).

A heuristic for a relatively global property is that its logical formula has no existential quantifiers (over a semigroup).

B.4.4.3 Semigroup properties

To be completed.

A **semigroup property** P is a class of semigroups S. For each $[P, S]$, define a set $C = C_{P,S} = \{T : T \leq S,\ T \in P\}$. A semigroup property is **local** is C is local in S for all S. A semigroup property is **global** is C is global in S for all S.

Commutativity is a semigroup property consisting of all semigroups S such that $ab = ba$ for all $a, b \in S$. Commutativity is global, but not local. For example, real numbers under multiplication are commutative, but their extension to quaternions is not.

For a local semigroup property example, let P consist of all semigroups S in which there exists elements $b, q \in S$ such that $bqb = b$. This property is local but not global. For example, P includes semigroup Q of rationals under multiplication, by setting $[b, q] = [2, \frac{1}{2}]$, but P does not includes a subsemigroup S of Q formed of the integers larger than one, since $bqb > b$

for all integers $b, q > 1$.

Finiteness is global semigroup property, and infiniteness is a local semigroup property. More generally, upper bounds on cardinality are global, lower bounds local.

Being a finite group is a global semigroup property, but being a group is not. For example, the semigroup of positive rationals under multiplication is a group, but its subsemigroup of rationals larger than one is not a group.

B.4.4.4 Operator properties

To be completed.

An **operator property** P is a class of pairs $[S, f]$ where S is a semigroup, and f is any function of the form $f : S^m \to S^n$. If $T \leq S$, let $f_T : T^m \to T^n : [t_1, \ldots, t_m] \mapsto f(t_1, \ldots, t_m)$ if $f(T^m)\langle T^n$, otherwise $f|_T$ denotes an error not equal to any function. For each, $[P, S, f]$, define a set $C = \{T : T \leq S,\ [T, f_T] \in P\}$. An operator property is **local** is C is local for all $[S, f]$, and **global** if C is global for all $[S, f]$.

B.4.4.5 Localization and globalization

To be completed.

The **localization** in S of a set of C of subsemigroups of S is the set of $C^- = \{T \leq S : \exists U \in C,\ U \leq T\}$. The localization is local.

The **globalization** in S of a set of C of subsemigroups of S is the set of $C^+ = \{T \leq S : \exists U \in C,\ U \geq T\}$. The globalization is global.

To do: define localization and globalization of properties. The possible aim are things like this: idempotency is the localization of the identity element property.

B.5 Morphisms

Morphisms, functions preserving structure in a category of mathematical object, are now quite central to the modern algebra.

Definition B.5.1. *A **morphism** between semigroups (or magmas) S and T is a function $\mu : S \to T$ such that*

$$\mu(ab) = \mu(a)\mu(b) \tag{B.5.1}$$

for all $a, b \in S$.

Each morphism has an image:

Definition B.5.2. *The* ***image*** *of a morphism μ is the set $\mu(S) = \{\mu(a) : a \in A\}$.*

Lemma B.5.1. *The image of morphism is a subsemigroup (submagma).*

Proof. Observe $\mu(S) = \mu(S^2) = \mu(S)\mu(S)$, so $\mu(S)$ is closed under multiplication, so $\mu(S) \leq T$. □

B.5.1 Congruences

Morphisms will be shown (in the next section) to be closely related to the notion of congruences and quotients (defined in this section).

Definition B.5.3. *A* ***congruence*** *on a semigroup (or magma) S is an equivalence relation $\sim \subseteq S^2$, such that*

$$(a \sim b) \implies ((ac \sim bc)\&(ca \sim cb)) \tag{B.5.2}$$

for all $a, b, c \in S$.

Write $[a]$ for the equivalence class of a under congruence $\sim$.

Lemma B.5.2. *The rule*

$$[a][b] = [ab]$$

is well-defined, and associative if S is a semigroup.

Proof. The question of being well-defined means that $[a][b]$ defines the same equivalence class, no matter which representatives a and b of the equivalence classes are chosen.

To answer this question of being well-defined, suppose that $[a'] = [a]$ and $[b'] = [b]$, meaning $a' \sim a$ and $b' \sim b$. Then

$$a'b' \sim a'b \sim ab,$$

so $[a'b'] = [ab]$. If S is a semigroup, then

$$[a]([b][c]) = [a][bc] = [a(bc)] = [(ab)c] = [ab][c] = ([a][b])[c]$$

so associativity inherited directly. □

This leads to the definition:

Definition B.5.4. *The* ***quotient*** *semigroup (or magma) $S/\sim$ of semigroup (or magma) S by congruence $\sim$ is the set of equivalence classes of $\sim$, with multiplication of classes by multiplication of representatives.*

The quotient semigroup is also called the **congruence semigroup**.

B.5.2 Natural morphisms and congruences

Each congruence defines a morphism:

Definition B.5.5. *The* ***natural morphism*** *for a congruence $\sim$ on semigroup (or magma) is the morphism to the quotient semigroup (or magma)*

$$\mu_\sim : S \to S/\sim : a \mapsto [a] \tag{B.5.3}$$

where $[a]$ is the equivalence class of a.

The natural morphism is surjective, and the image of the natural morphism is the quotient semigroup.

Each morphism defines a congruence:

Definition B.5.6. *The* ***natural congruence*** *for a morphism $\mu : S \to T$ is the relation defined*

$$\sim_\mu = \{[a, b] : \mu(a) = \mu(b)\}, \tag{B.5.4}$$

so $a \sim_\mu b$ if and only if $\mu(a) = \mu(b)$.

The quotient semigroup for the natural congruence is isomorphic to the image μ, under the isomorphism $[a] \mapsto \mu(a)$.

Diversion B.5.1. The natural congruence of the natural morphism of a congruence is the congruence itself, which could be written $\sim_{\mu_\sim} = \sim$ in slightly overloaded notation. The natural morphism of the natural congruence of a morphism is the morphism is restricted to the image, which could be written $\mu_{\sim_\mu} = \mu|_{\mu(S)}$ in slightly overloaded notation.

Diversion B.5.2. Unlike group theory or ring theory, there is no clear kernel of a morphism. The natural congruence plays most of the role of a kernel.

B.5.3 Lattice of congruences

To be completed.

If $\sim$ and $\equiv$ are congruence, then $\sim \cap \equiv$ is a congruence.

The extreme congruences the identity relation ($a \sim b$ if and only $a = b$) and the universal relation ($a \sim b$ for all $a, b \in S$).

In the identity relation, each class is a singleton set, so $[a] = \{a\}$. The natural morphism $a \mapsto [a]$, for identity relation is an isomorphism, with inverse $[a] \mapsto a$.

In the universal relation, there is only one class, so $[a] = S$ for all $a \in S$, and the quotient group has only one element.

Diversion B.5.3. Even if S is empty, the quotient group under the universal relation has one element.

B.5.4 Induced congruences

Definition B.5.7. *The **induced congruence** of relation ρ on semigroup (or magma) S is the smallest congruence $\sim_\rho$ such that $\rho \subseteq \sim_\rho$.*

The induced congruence always exists: just take the intersection of all the congruences containing ρ. The universal congruence always contains ρ.

If clear from context, write S/ρ for $S/\sim_\rho$.

B.5.5 Mergers

Definition B.5.8. *The **merger** semigroup of subset $T \subseteq S$ is the quotient semigroup of the congruence induced by the relation T^2.*

In other words, the merger semigroup is the a semigroup derived from S making all elements of subset T equivalent.

When clear from context, write the merger semigroup as $S/T = S/_{T^2}$.

B.5.6 Cosets

To be completed.

Definition B.5.9. *A **coset** C of semigroup (or magma) S is a set C such that there exist $a \in S$ and a congruence $\sim$ for S such that*

$$C = [a]. \tag{B.5.5}$$

meaning that is an equivalence class of a congruence on S.

B.5.7 Ideals

To be completed.

Semigroup theory often refers to a special type of subset of a semigroup.

Definition B.5.10. *An* ***ideal*** *of semigroup (or magma) is a subset I such that*

$$SI, IS \subseteq I. \tag{B.5.6}$$

In merger semigroup S/I, the set I is an equivalence class, so it is a coset, and all other equivalence classes are singleton sets.

Diversion B.5.4. In S/I, the class I is an absorbing element (see later in the report). In other words, I is the zero element of S/I.

Conversely, any set I that is a pre-image of $0 \in T$ in a morphism $S \to T$ is an ideal.

Diversion B.5.5. The term **ideal** is borrowed from ring theory, but not entirely consistently.

For example, suppose that R is a ring, and I is a ring-theoretic ideal of the ring. Then the quotient ring R/I can be define.

The inconsistency is that one generally has $R^\times/I \neq (R/I)^\times$. But $(R/I)^\times$ is an image of $R^\times/I$.

B.5.8 Normal subsets

To be revised.

Definition B.5.11. *A* ***normal set*** *N of semigroup (or magma) S is a set N such that there exist a morphism $\mu : StoT$ such that $N = \mu^{-1}(\{1\})$.*

Diversion B.5.6. The term normal subset aims to generalize the notion of a normal subgroup from group theory.

If G is a group, and N is a normal subgroup of N. Then N is normal subset of G.

Diversion B.5.7. The whole set of S is the only subset that is normal and an ideal.

B.5.9 Miscellaneous features

Diversion B.5.8. The **center** Z of a semigroup S is the set of z such that $zs = sz$ for all $s \in S$. A semigroup is **normally centered** if the center is normal.

Diversion B.5.9. In the intersection of computability theory and algebra, there is a following notion about monoids, which easily generalizes to semigroups.

Let S and T be semigroups. Let R be a subset of S. We say that T can **recognize** R if there is a semigroup morphism $f : S \to T$ such that $R = f^{-1}(f(R))$. Every morphism-pre-image in S is thus recognized.

A set recognized R is necessarily a union of congruent sets, since a congruent set is simply a morphism-pre-image of singleton set.

A subset R of S is said to be **recognizable** if is recognized by a finite semigroup T.

Recognizable subsets of a finitely-generated free monoid are known as **regular languages** (this may be the Myhill–Nerode theorem).[2]

B.6 Idempotents

Some semigroups have special elements that act like 0 or 1 in standard arithmetic.

Both 0 and 1 are examples of idempotents, which equal their squares. The idempotents of a semigroup lead to a better understanding the semigroup's structure.

B.6.1 Identity elements

Some multiplicative semigroups have an element that generalizes the number 1 in standard arithmetic.

Definition B.6.1. *An **identity element** of a semigroup (or magma) S is an element e, such that $ea = ae = e$ for all $a \in S$.*

Diversion B.6.1. Identity elements are defined similarly for non-multiplicative semigroups, see §B.6.1.1.

Diversion B.6.2. When clear from context, we borrow the integer notation and write 1 for an identity element of a multiplicative semigroups S, if S has an identity element. If we write $1 \in S$, then we mean 1 is an identity element of S.

Lemma B.6.1. *A semigroup has at most one identity element.*

Proof. If 1 and e are identity elements of S, then $1 = 1e = e$, so $1 = e$. □

Definition B.6.2. *A **unital** semigroup, or **monoid**, is a semigroup with an identity element.*

[2] I have not personally verified this result.

Diversion B.6.3. Each element $t \in S$, induces two functions $s \mapsto ts$ and $s \mapsto st$ in the set S^S of functions from S to itself. If both these are functions are the identity function, then t is the identity element.

Diversion B.6.4. The identity element can also be called **neutral element**, perhaps because multiplication by the neutral element seems to have no effect. In some settings, this may avoid confusion with other meanings of identity (such as an equation).

In multiplicative semigroups, the identity element can also be called **unity** or **one**. (The term **unit** should perhaps be avoided, since it is often used to designate invertible elements.)

Diversion B.6.5. Multiplicative semigroups formed of square matrices often use I for the identity element, instead of 1. This may be mainly to avoid confusion with another meaning of 1: the usual identity matrix I has all diagonal entries equal 1 (and all off-diagonal entries 0), where 1 is an element of another algebraic set, a semiring, and 1 is its multiplicative identity.

B.6.1.1 Additive identity

In additive semigroups, the identity element may be written as 0, and called **zero**. So, $0 + a = a + 0 = a$ for all $a \in S$. If $0 \in S^+$, then S^+ is an **additive monoid**.

Diversion B.6.6. If considering multiple semigroups with overlapping sets, then we can write 0^+ for additive identity element of S^+, if necessary to distinguish from other meanings of 0, such as those in §B.6.2. Alternatively, one can write o, or $\oplus$.

B.6.1.2 Sidentity elements

To be revised.

An element r of a semigroup can be called **right sidentity** if $sr = s$ for all $s \in S$. A **left sidentity** l satisfies $ls = s$ for all $s \in S$.

Lemma B.6.2. *If a semigroup S has a right sidentity r and a left sidentity l, then it has an identity element $1 = l = r$.*

Proof. As sidentities, $l = lr = r$, so $r = l$. Let $1 = l = r$, and 1 is both a left sidentity and a right sidentity, so $1a = a = a1$. □

Diversion B.6.7. If S has no identity ($1 \notin S$), then S can only have one sidentity on at most one side (either left or right).

Diversion B.6.8. The set R of right sidentities of a semigroup S is closed under multiplication, and thus forms a subsemigroup. Multiplication in R is **left multiplication**: $ab = a$ for all $a, b \in R$.

Conversely, in any semigroup whose multiplication is left multiplication, all elements are right sidentities.

B.6.2 Absorbing elements

Some multiplicative semigroups have an element that generalizes the number 0 in standard arithmetic.

Definition B.6.3. *An* ***absorbing*** *element of semigroup S, is an element o such that $oa = ao = o$ for all $a \in S$.*

Diversion B.6.9. Absorbing elements can be defined similarly in non-multiplicative semigroups.

Diversion B.6.10. In a multiplicative semigroup S, an absorbing element is usually written as 0, when clear from context. It is also called **zero**, when clear from context.

Lemma B.6.3. *A semigroup can have at most one absorbing element.*

Proof. If 0 and o are absorbing elements, then $o = 0o = 0$. □

Lemma B.6.4. *If element $\emptyset \in S$ is both an identity element and an absorbing element, then $|S| = 1$.*

Proof. Let $a \in S$. Then $a = \emptyset a = \emptyset$. So, $S = \{\emptyset\}$. □

Diversion B.6.11. Additive and multiplicative semigroups often have overlapping sets, as in the case of integers, for example. In some of these cases, the additive identity is not multiplicative absorbing.

The notation 0 fails in this case, because one gets $0 \neq 0$, where the left 0 means additive identity and the right 0 means multiplicative absorbing element. If this clash happens, then alternative notation is needed.

Perhaps most logical would be to write 0^+ for the additive identity, in line with using a superscript to indicate that binary operation notation in effect.

Perhaps more intuitive would be to write ∞ for the multiplicative absorbing element, but this would mean writing $0\infty = \infty$ must be tolerated.

Diversion B.6.12. For a somewhat natural example where $0 \neq 0^+$, consider the set

$$S = \{[n] : n \in \{1, 2, 3, \ldots, \infty\}\}$$

and define both addition and multiplication on S through the rules

$$[a] + [b] = [\max(a, b)]$$
$$[a][b] = [a + b]$$

with the convention that $\max(a, \infty) = \max(\infty, b) = \infty$ and $a+\infty = \infty+b = \infty$. This may be an instance of a **tropical** algebra. Both operations are associative and commutative, and multiplication is also distributive over addition.

Then, $[\infty]$ is an absorbing element, multiplicatively (and additively), while $[1]$ is an identity element additively.

Clearer notation might be to write $0 = [1]$, and $[n] = 0^n$, and $0^\infty = \infty$.

Diversion B.6.13. A fairly mild condition that implies $0 = 0^+$, is that multiplication is right distributive over addition, and $-$ is a post-subtracter. To see this:

$$\begin{aligned}0 &= (0+0) - 0\\ &= (00+00) - 0\\ &= ((0+0)0) - 0\\ &= 0 - 0\\ &= (0^+ + 0) - 0\\ &= 0^+.\end{aligned}$$

Diversion B.6.14. If the set S from the note B.6.12 is extended to include $[0]$ but keep the same formulaic definition for addition and multiplication, then $[0]$ becomes both the additive identity and the multiplicative identity.

In this case, a second notational clash arises $1 = 0^+$.

B.6.2.1 Additive absorbing element

In an additive semigroup, an absorbing element can be written with the symbol ∞, and be called **infinity**. So, generally, $\infty + a = a + \infty = \infty$ holds for all $s \in S$.

Diversion B.6.15. It can sometimes help to think of ∞ in an additive semigroup as negative infinity, instead of positive infinity. For example, $\exp(0) = 1$ and $\exp(-\infty) = 0$, making the exponential function a semigroup isomorphism $\exp : (\mathbb{R} \cup \{-\infty\})^+ \to \mathbb{R}^\times_{\geq 0}$.

B.6.2.2 Side-absorbing elements

An element $p \in S$ is **left side-absorbing** if $pa = p$ for all $a \in S$. An element q is **right side-absorbing** if $aq = q$ for all $a \in S$. If p is left-side absorbing and q is right side-absorbing, then $p = pq = q$, so $0 = p = q$ and 0 is an absorbing element.

Diversion B.6.16. If S has no absorbing element ($0 \notin S$), then S can have side-absorbing elements on at most one side (either left or right).

If S has left multiplication $ab = a$ for all $a, b \in S$, then all elements of S are both left-side-absorbing and right-sidentities.

The left-side-absorbing elements are closed under multiplication, forming a subsemigroup, whose multiplication is left multiplication.

B.6.3 Comparison

If S is a multiplicative semigroup and $o, i \in S$, write $o \subset i$ if

$$oi = o = io. \tag{B.6.1}$$

If 0 is an absorbing element, and 1 is an identity element, then

$$0 \subset 1. \tag{B.6.2}$$

Diversion B.6.17. A mnemonic for $\subset$ is (B.6.2), with the roundness of the 0 on the left of the symbol $\subset$ and the straightness of 1 on the right of $\subset$.

Diversion B.6.18. Another mnemonic is $0 < 1$. We do not use $<$ for $\subset$, because it $<$ has too many other meanings.

When clear from context, then we may say that o is **less** than i.

Diversion B.6.19. We could also say that o **absorbs** i, or that o **nullifies** i, or that i **identifies** o.

Diversion B.6.20. In a semigroup whose elements are sets and whose multiplication is intersection of set, the notation $\subset$ matches with one meaning of $\subset$ as containment of sets.

Sometimes, $\subset$ is used for strict containment, the same way that $<$ is used for strict comparison. Strict containment insists the two compared sets are not equal. This clashes with the semigroup meaning of $\subset$, which allows equal semigroup elements.

Diversion B.6.21. In additive semigroups, the notation $\subset$ makes less sense, so we may wish to avoid it.

The relation $\subset$ is transitive and anti-symmetric:

- If $o \subset e$ and $e \subset i$, then $o \subset i$. To see this, compute oi, as $oi = (oe)i = o(ei) = oe = o$. A mirrored calculation shows $io = o$ too.
- If $o \subset i$ and $i \subset o$, then $i = o$. To see this, compute $o = oi = i$.

The relation $\subset$ is multiplicative in the following sense:

- If $o \subset i$ and $e \subset i$, then $oe \subset i$. To see this, compute $(oe)i = o(ei) = oe$. The mirrored calculation shows $ioe = oe$.

- If $o \subset e$ and $o \subset i$, then $o \subset ei$. To see this, compute $o(ei) = (oe)i = oi = o$. The mirrored calculation shows $eio = ei$.

The multiplicative property leads to the following two subsemigroups of S:

$$S_{\subset e} = \{o \in S : o \subset e\}, \tag{B.6.3}$$

$$S_{e\subset} = \{i \in S : e \subset i\}. \tag{B.6.4}$$

These subsemigroups can be defined for each e. The subsemigroups may be empty semigroups, or they may be the whole semigroup.

Diversion B.6.22. Yet other semigroups can defined from this. For any $o, i \in S$, let $S_{o\subset i} = S_{o\subset} \cap S_{\subset i} = \{e \in S : o \subset e \subset i\}$. If $o \not\subset i$, the $S_{o\subset i} = \{\}$ is empty (because $\subset$ is transitive).

Diversion B.6.23. A subset F of S can be called full if $S_{o\subset i} \subseteq F$ for all $o, i \in F$.

B.6.4 Idempotents

Definition B.6.4. *An* ***idempotent*** *in a multiplicative semigroup S, is an element e such that*

$$e^2 = e. \tag{B.6.5}$$

An identity element 1 is idempotent, and so is an absorbing element 0.

An element is idempotent if and only if it has order 1 (meaning that the subsemigroup it generates has size 1).

Diversion B.6.24. If a semigroup S is a group, then there is exactly one idempotent element: the identity element 1.

To see this, calculate $e = e1 = e(ee^{-1}) = (ee)e^{-1} = ee^{-1} = 1$.

Diversion B.6.25. If a semigroup S has an identity element 1, and a (cancelling) post-divider $/$, (meaning $(ab)/b = a$ for all a, b, see later for more discussion), then there is exactly one idempotent: the identity element 1.

To see this, calculate, $e = (ee)/e = (1e)/e = 1$.

Diversion B.6.26. In additive semigroups, an element e is idempotent if $e + e = e$.

In case of any confusion, we may call e **additively idempotent**. A logical alternative **idemscalant** sounds too awkward.

Diversion B.6.27. In a ring R, the semigroup R^+ is a group, so the only additively idempotent element is the additive identity 0. The property of being additive idempotent is not too interesting.

Moreover, the element 0 is also multiplicatively idempotent. So, all additively idempotent elements are also multiplicatively idempotent.

In a ring, idempotent is used to mean multiplicatively idempotent (idempotent in $R^\times$).

Diversion B.6.28. If R has two operations, $+$ and $\times$, but is not quite a ring in that semigroup R^+ is not a group, then the two meanings of idempotent may yet cause confusion, so the qualifiers additive or multiplicative may be necessary clarify which type of idempotent is being considered.

If R is sufficiently ring-like, then multiplicative idempotents may more interesting and important than additive idempotents, just like in a ring. In this case, the term idempotent could be used to refer to multiplicative idempotent, unless qualified by the term additive.

Diversion B.6.29. Element e is idempotent if and only if $e \subset e$. In other words, the relation $\subset$ is reflective on idempotents.

The set E of idempotents in a semigroup S is partially ordered by the relation $\subset$.

Diversion B.6.30. The alternative notations $\leq$ and $\subseteq$ for $\subset$ make more sense when applied to idempotents, because of all the axioms for partial ordering are met.

B.7 Inverses

In semigroups, there are many types of inverse. Table B.2 summarizes some of these types of inverse. An element is invertible if it has an inverse, but there are as many types of invertibility as types of inverses.

Semigroups sometimes have non-invertible elements. Idempotents are invertible under most definitions, and finite semigroups have at least idempotent, so finite semigroups have at least invertible element (under most definitions of an inverse).

The amount of invertible elements, and the strength of their invertibility is one how to measure close a semigroup is to group, and is a basic part of the understanding of the structure of a semigroup.

In additive semigroups, especially those which overlap with a multiplicative semigroup, the term inverse should be translated to a negation. An invertible element element should be translated to a negate-able element, but there is often little need for such a term.

For consistency, we will symbol q for an inverse of b.

Type	Condition
Middle (regular, default)	$bqb = b$
Mutual	$bqb = b$, $qbq = q$
Right	$bbq = b$
Left	$qbb = b$
Sidle	$bbq = qbb = b$
Co-mutual	$bqb = b$, $qbq = q$, $bq = qb$
Divisional (right)	$abqb = ab$
Divisional (left)	$bqbc = bc$
Wedge	$abqbc = abc$
Power	$bqb = b$, $q = b^n$, $n \in \{1, 2, 3, \dots\}$
Post-divisional	$abq = a$
Pre-divisional	$aqb = a$
Idle (right)	$qbqb = qb$
Idle (left)	$bqbq = bq$
Inflatable	$b = pbqbr$
Semi-unital (right)	$qb = 1$
Semi-unital (left)	$bq = 1$
Unital	$bq = qb = 1$

Table B.2: Various types of inverses q of multiplicative semigroup element b

B.7.1 Middle inverses

In semigroup theory, the nearly standard meaning of inverse and invertibility is the following.

Definition B.7.1. *A **middle inverse** of b of a multiplicative semigroup is an element q such that:*

$$bqb = b. \tag{B.7.1}$$

*If b has a semigroup inverse, then we say that b is **middle invertible**.*

Unless otherwise noted, the terms **inverse** and **invertible**, using without modifiers, will default to middle inverse and middle invertible.

Diversion B.7.1. A **middle invertible** element is often called a **regular element**, after terminology of von Neumann introduced in non-commutative algebra.

Outside this report, a **middle inverse** is sometimes called a **pseudo-inverse**, to distinguish from stronger types of inverses.

Diversion B.7.2. Although the usage of the term **regular** is rather standard terminology in parts of semigroup theory, this report declines to used the term **regular**.

The term **regular** is very overloaded in mathematics, with many unrelated meanings; the general term **regular** does not generally imply invertibility; the everyday meanings of **regular** as ordinary or typical seems inappropriate.

Diversion B.7.3. A **regular semigroup** is then a semigroup in which all elements are middle invertible (also known as regular).

Diversion B.7.4. A $b \in S$ with no semigroup inverse (an irregular element) can sometimes have a divisional inverse. For example, consider the null semigroup S, such that $ab = 0$ for all $a, b \in S$. Then any element $q \in S$ is a divisional inverse of b. If $b \neq 0$, then no element q is a semigroup inverse of b.

Lemma B.7.1. *If / is a mid-divider, then b is middle-invertible if and only if there exists an idempotent e satisfying equations*

$$(b/e)e = b,$$
$$(e/b)b = e.$$

In this case, $q = e/b$ is a middle inverse of b.

Proof. Suppose that q is a middle inverse. Let $e = qb$. Then

$$\begin{aligned} ee &= (qb)(qb) = q(bqb) = qb = e, \\ (b/e)e &= ((bqb)/e)e = ((be)/e)e = be = bqb = b, \\ (e/b)b &= ((qb)/b)b = qb = e, \end{aligned}$$

so e is idempotent, and satisfies the two hypothesized equations.

Conversely, suppose that e is the idempotent satisfying the two hypothesized equations. Let $q = e/b$. Then

$$bqb = b((e/b)b) = be = ((b/e)e)e = (b/e)(ee) = (b/e)e = b,$$

so q is a middle inverse of b. □

Diversion B.7.5. The lemma above would be stronger if it did not depend on the knowing e, but found e via divider.

Elsewhere in this report, we show how to construct such an e using division, but we use more than just a mid-divider. (We resort to both left and right division, and to a post-divider.)

Diversion B.7.6. The note to be moved to section on Green's relations.

The statement of the lemma above can be translated into the terminology of Green's relations, commonly used in semigroup theory, as follows.

An semigroup element is regular (middle-invertible) if and only if its L-class contains an idempotent.

Diversion B.7.7. This note to be moved section on post-dividers.

Middle inversion can be used for partial post-division, relative to a restricted set of inputs:

Lemma B.7.2. *If b is an inverse of p, then $/$ defined by*

$$d/b = dp. \tag{B.7.2}$$

is a post-divider relative to $T = \{(up, b) : u \in S\}$.

Proof. Put $d = ab$ for $(a, b) = (up, b) \in T$. Then $d/b = dp = (ab)p = ((up)b)p = u(pbp) = up = a$. □

In arbitrary semigroups, the product of middle invertible elements might not be invertible, but in some almost commutative semigroups it is, as shown below.

Lemma B.7.3. *If S is a medial semigroup (where $(ab)(cd) = (ac)(bd)$ for all a, b, c, d, see (B.3.10)), and q and p are middle inverses of b and d, then pq is a middle inverse of bd.*

Proof. Calculate:

$$\begin{aligned}(bd)(pq)(bd) &= b(dp)(qb)d\\ &= b(dq)(pb)d\\ &= (bd)(qp)(bd)\\ &= (bq)(dp)(bd)\\ &= (bq)(db)(pd)\\ &= b(qd)(bp)d\\ &= b(qb)(dp)d\\ &= (bqb)(dpd)\\ &= bd,\end{aligned}$$

so pq is a middle inverse of bd. □

Similarly, the qp is also a middle inverse of bd.

B.7.2 Right inverses

In finite semigroups, right invertible elements are co-mutually invertible.

Lemma B.7.4. *If $\langle b\rangle$ is finite, and b is right invertible, then b has a co-mutual inverse.*

Proof. Let q be a right inverse of b, so that $bbq = b$. Let $b^e = b^{e+p}$ for positive integers e and p, which must exist since b has finite order.

If $e = 1$, then b is a periodic element, so it has a co-mutual inverse b^{p-1}.

Otherwise $e = c+1$ for a positive integer c. Then $b^{c+1}q^c = b$ by repeated application of the rule $bbq = b$. Similarly $b^{c+1+p}q^c = b^{1+p}$. But then $b = b^{c+1}q^c = b^{c+1+p}q^c = b^{1+p}$, so b is also a periodic element, and has co-mutual inverse b^{p-1}. □

B.7.3 Mutual inverses

Definition B.7.2. *A **mutual inverse** of b in a multiplicative semigroup is an element q such that*

$$bqb = b, \qquad qbq = q. \tag{B.7.3}$$

Diversion B.7.8. In other words, q is mutual inverse of b if and only if q and b are middle inverses of each other. (Hence this report's term **mutual**.)

Every middle invertible element has a mutual inverse:

Lemma B.7.5 (Well-known!)**.** *If q is a middle inverse of b, then $p = qbq$ is a mutual inverse of b.*

Proof. To see this, just compute bpb and pbp:

$$bpb = b(qbq)b = (bqb)qb = bqb = b, \tag{B.7.4}$$

$$pbp = (qbq)b(qbq) = q(bqb)qbq = q(b)qbq = q(bqb)q = qbq = p. \tag{B.7.5}$$

□

Diversion B.7.9. In some branches of semigroup theory, the mutual inverse serves as the default definition of the **inverse**.

From this report's perspective, a middle (or weaker) inverse implies division, and thereby deserves the name inverse.

Diversion B.7.10. A semigroup in which every element has a unique mutual inverse is known as a **inverse semigroup**; see Petrich [Pet84].

Outside this report, in an inverse semigroup, the notation b^{-1} is sometimes used for the unique mutual inverse of b. This report reserves this notation for co-mutual inverse.

B.7.4 Co-mutual inverses

Definition B.7.3. *A* ***co-mutual inverse*** *of b of a multiplicative semigroup is an element q such that*

$$bqb = b \qquad qbq = q \qquad bq = qb. \tag{B.7.6}$$

If b has a co-mutual inverse, then b is ***co-mutually invertible****.*

Diversion B.7.11. In other words, a co-mutual inverse q of b is a mutual inverse of b that commutes with b.

Lemma B.7.6 (Well-known, Wagner, Preston, earlier?)**.** *Every b has at most one co-mutual inverse.*

Proof. Suppose p and q are both co-mutual inverses of b. Then

$$\begin{aligned}
p &= pbp & &= p(b)p = p(bqb)p \\
&= pbqbp & &= (pb)(qb)p = (bp)(bq)p \\
&= bpbqp & &= (bpb)qp = (b)qb \\
&= bqp & &= (bq)p = (qb)p \\
&= qbp & &= q(bp) = q(pb) \\
&= qpb & &= qp(b) = qp(bqb) \\
&= qpbqb & &= q(pb)(qb) = q(bp)(bq) \\
&= qbpbq & &= q(bpb)q = q(b)q \\
&= qbq = q,
\end{aligned}$$

where equations in the right column mark with parentheses the substitutions based on three equations ($bqb = b$, $qbq = q$ and $bq = qb$) ensured by b and q being co-mutual inverses, and the corresponding 3 equations for b and p. □

Diversion B.7.12. This result can be surprising to a semigroup novice (such as me).

Diversion B.7.13. The proof above might not be the best possible proof; perhaps the Knuth–Bendix algorithm can be applied to obtain different proofs. The given proof takes ten steps, but two steps can be considered double steps, so that 12 rewriting steps (via co-mutual inversion) are taken (ignoring steps for associativity). The step counts can be written:

$$\begin{array}{cccc}
b \mapsto^2 bqb & qbq \mapsto^1 q & qb \mapsto^2 bq & bq \mapsto^1 qb, \\
bpb \mapsto^2 p & p \mapsto^1 pbp & pb \mapsto^2 bp & bp \mapsto^1 pb.
\end{array}$$

The proof has a symmetry in p and q: swapping p and q them reverse the direction of the proof, but also reversing the order of the terms in the products in the first and second half of the proof.

Because of the uniqueness of a co-mutual inverse, write b^{-1} for the co-mutual inverse. Write b^{-n} for $(b^{-1})^n$ for each positive integer. Write b^0 for $bb^{-1} = b^{-1}b$ (though, $b^0 \neq 1$ is possible). The properties of the co-mutual inverse imply that $b^x b^y = b^{x+y}$ for any integers x and z (including zero and negative integers).

Lemma B.7.7. *Element $b \in S$ is co-mutually invertible if and only if $b \in G$ where*

- *G is a subsemigroup of S and*

- *G is a group.*

Proof. Suppose that b is co-mutually invertible, with co-mutual inverse b^{-1}. Consider the subsemigroup $\langle b, b^{-1}\rangle$ generated by b and its co-mutual inverse b^{-1}. Then $\langle b, b^{-1}\rangle = \{b^n : n \in \mathbb{Z}\}$ in the notation above. The element b^0 is the identity of the subsemigroup, and each element b^n has an inverse b^{-n}, thus meeting the axioms of a group.

Suppose b that belongs to subsemigroup G and that G is a group. Being a group, G has an identity element e such that $eg = ge = g$ for all $g \in G$, and an inverse q of b such that $bq = qb = e$. Then q is a co-mutual inverse of b, since $bqb = be = b$ and $qbq = qe = q$ and $bq = e = qb$. □

Diversion B.7.14. A semigroup in which every element has a co-mutual inverse is known as an **completely regular semigroup**; see Petrich [Pet84].

Diversion B.7.15. To see that $b^0 \neq 1$ is possible, consider a semigroup $S = \langle b, b^{-1}, d, d^{-1}\rangle$, where b and b^{-1} are co-mutual inverse, as are d and d^{-1}, with no other relations. So, S is the free semigroup product of $\mathbb{Z}$ with itself (which is larger than the free group product!). Each element of S has a factorization $g_1^{z_1} \dots g_n^{z_n}$ for some positive integer n, for some integers $z_1, \dots, z_n$, and $g_i \in \{b, d\}$ with $g_i \neq g_{i+1}$. The positive integer n can be called the length of the element. Then d^0b^0 has length two, while b^0 has length one, so $d^0b^0 \neq d^0$, meaning that $b^0 \neq 1$. Indeed this semigroup is non-unital.

Diversion B.7.16. Let $B = \{a : aHb = a\}$ be Green's H-class of b.

Then, b is co-mutually invertible if and only if $B^2 = B$. Also, b is co-mutually invertible if and only if $B^2 \cap B \neq \varnothing$. Also, b is co-mutually invertible if and only if B is a group.

If b is co-mutually invertible, and $b \in G$ for a group $G \leq S$, then then $G \leq B$. In other words, the H-class B of b is the largest group containing b.

In other words, the set of the co-mutually invertible elements of a semigroup can be partition into a set of disjoint groups.

B.7.5 Sidle inverses

Definition B.7.4. *A* ***sidle inverse*** *q of element b is an element that is both a left and a right inverse, meaning that $bbq = b = qbb$.*

Lemma B.7.8. *A sidle inverse q of b is a middle inverse of b, and q commutes with b.*

Proof. Calculate:

$$bqb = bq(bbq) = b(qbb)q = bbq = b,$$

so q is a middle inverse. Calculate

$$qb = q(bbq) = (qbb)q = bq,$$

so q and b commute. □

A sidle inverse is not necessarily a co-mutual inverse. For example, if $b = 0$, then all q are sidle inverses of b. But if b has a sidle inverse, then it has a co-mutual inverse as shown below.

Lemma B.7.9. *If q is a sidle inverse of b, then $p = qbq$ is a co-mutual inverse of b.*

Proof. Lemma B.7.8 shows that q is a middle inverse. Lemma B.7.5 shows that $p = qbq$ is a mutual inverse.

To see that p commutes with b, use the fact from Lemma B.7.8 that q commutes with b: so $pb = (qbq)b = (qb)(qb) = (bq)(bq) = b(qbq) = bp$.

Therefore, p is a co-mutual inverse of b. □

B.7.6 Power inverses

A **power inverse** q of b is a middle inverse of b with $q \in \langle b \rangle$, meaning $q = b^n$ for a positive integer n. Any such positive integer n is called an **inversion degree**.

A power inverse q of b is actually co-mutual inverse of b. An element b has a power inverse if and only if b is a torsion element.

Lemma B.7.10. *If b is a torsion element with period p, then b is invertible with a middle inverse:*

$$q = b^{2p-1}. \tag{B.7.7}$$

If $p > 1$, then $b^{2p-1} = b^{p-1}$.

Proof. The definition of semigroup inverse says that q is a semigroup inverse of b provided that $bqb = b$. But

$$\begin{aligned} bqb &= bb^{2p-1}b \\ &= b^{2p+1} \\ &= b^p b^{p+1} \\ &= b^p b \\ &= b^{p+1} \\ &= b, \end{aligned}$$

as required.

If $p > 1$, then $p - 1 \geq 1$, so $b^{p-1} \in \langle b \rangle$, and $b^{2p-1} = b^{p-2}b^{p+1} = b^{p-2}b = b^{p-1}$. □

Diversion B.7.17. In the final sentence of the proof above, in the case of $p = 2$, the factor b^{p-2} is not meant literally, as an element 1 (which may not exist in the semigroup), but can simply be omitted from the expression.

Diversion B.7.18. Consequently the smallest inversion degree n of b is $p-1$ if the period is $p > 1$.

If the period is $p = 1$, then the smallest inversion degree is $n = 1$. strictly speaking.

For the purpose of division-by-inversion, the effective inversion degree can be considered to be 0 in this case.

B.7.7 Divisional inverses

Definition B.7.5. *A* ***(right) divisional inverse*** *of b in a multiplicative semigroup S is an element q such that*

$$abqb = ab \tag{B.7.8}$$

for all $a \in S$.

Diversion B.7.19. A left division inverse q of b satisfies $bqbc = bc$ for c.

Diversion B.7.20. In an additive semigroup, we can speak of subtractive negative q of b if $a + b + q + b = a + b$ for all a.

Lemma B.7.11. *If q is a middle inverse of b, then q is a right divisional inverse of b.*

Proof. Computing, $abqb = a(bqb) = a(b)$, so q is divisional right inverse. □

Diversion B.7.21. A right divisional inverse q of b is also wedge inverse of b, because $abqbc = abc$ for all a, c.

But some wedge inverses are not divisional inverses. For example, in a 3-nilpotent semigroup (where the product of three or more element is zero), every q is a wedge inverse of b, but b has a right divisional inverse if and only if b is right side-absorbing (meaning $ab = 0$ for all a).

Definition B.7.6. *A* ***post-divisional inverse*** *q of b is an element such that any of one of the following equivalent conditions holds:*

- *$d/b = dq$ defines a post-divider (relative to $T = S \times \{b\}$),*

- *$abq = a$ for all $a \in S$,*
- *bq is a **right identity element** of S.*

Diversion B.7.22. In associative key agreement, (recall that) post-division enables recovery of Alice's exact secret a, not just an equivalent secret, which is more damaging if Alice used a for some other purpose. Consequently, a post-divisional inverse has more impact than a (weak) inverse.

Diversion B.7.23. An **pre-divisional inverse** of b is an element q such that $d/b = dq$ defines an pre-divider; equivalently, $dqb = d$ for all $d \in S$, so that qb is right identity in S.

Diversion B.7.24. An post-divisional or pre-divisional inverse is also a divisional inverse.

Diversion B.7.25. If b has a post-divisional inverse q, then b is **right cancellative**: if $ab = cb$, then $a = abq = cbq = c$. So, if b is not cancellative, b cannot have a post-divisional inverse.

Diversion B.7.26. If q is a post-divisional inverse of b, then b is a (middle) inverse of q, because $qbq = q$ (by putting $a = q$).

Diversion B.7.27. If $b \in S$ has both a post-divisional right inverse and S has a left identity l, then S is unital (since $l = lbq = bq$ is both a left and right identity).

B.7.8 Wedge inverses

To be completed. Copy from later sections.

B.7.9 Strident inverses?

Definition B.7.7. *A **strident semigroup** is a pair (S, i), where S is a semigroup, $i \in S$, and i is a right identity of S (meaning $si = s$ for all $i \in S$). We call i the **stridentity**.*

Diversion B.7.28. The term **strident** is new, but the underlying notion is not entirely, as discussed below: some axiomatic definitions of a group do take a two-sided identity as axiomatic, but prove its the existence of a two-sided identity from other axioms, including the existence of a one-sided inverse.

Diversion B.7.29. If the semigroup S in a strident semigroup (S, i) does not have a left identity, then S may have many right identities. In a strident semigroup, one of the right identities is especially selected as the stridentity, and is distinguished from the other right identities.

If the semigroup S in a strident semigroup has a left identity element, then S is unital (a monoid) and $i = 1$.

Definition B.7.8. *If (S, i) is a strident semigroup and $b \in S$, then $q \in S$ is* ***strident inverse*** *of b if $bq = i$.*

Diversion B.7.30. A strident inverse of b is a post-divisional right inverse of b.

Diversion B.7.31. A post-divisional right inverse q is not necessarily a strident inverse. If q is a post-divisional inverse of b, then bq is a right identity, but it might not be the stridentity i.

Diversion B.7.32. A well-known (and somewhat surprising) theorem says that if (S, i) is strident semigroup, and every $b \in S$ has a strident inverse, then S is a group.

The following little lemma can help to establish this:

Lemma B.7.12. *If (S, i) is a strident semigroup, b has strident inverse q and q has a strident inverse, then q is a co-mutual inverse of b.*

Proof. We need to prove the three co-mutual inverse equations $bqb = b$, $qbq = q$ and $bq = qb$.

From the strident semigroup definition, the proof uses the four equations

$$bi = b, \qquad qi = q, \qquad bq = i, \qquad qu = i,$$

where u is a strident inverse of q. (Aside: this proof is a little more general the lemma's claim, because it only requires these four equations, and does not require i to be a right identity for the whole of S.)

To show that b is a middle inverse of q, compute $qbq = qi = q$.

To show that q and b commute, compute $qb = qbi = qbqu = qiu = qu = i = bq$.

To show that q is a middle inverse of b, compute $bqb = bi = b$, using the previous fact that $qb = i$. □

This also shows $ib = b = bi$. If every b is strident invertible, we then can see that i is a full identity (both left and right), and the co-mutual inverse b^{-1} is the required group inverse.

B.7.10 Inflatable elements, inflators, and volume?

An element b of a semigroup S is **inflatable** if there exists $p, q, r \in S$ such that:

$$b = pbqbr. \tag{B.7.9}$$

The triple $[p, q, r]$ is called an **inflator** of b. An inflator has a role similar to an inverse, except that it is not a single element.

If b has a middle inverse q (with $b = bqb$), then b is inflatable, by putting $p = bq$ and $r = qb$, because then $pbqbr = (bq)b(q)b(qb) = (bqb)q(bqb) = bqb = b$. There might exist inflatable b with no middle inverse.

The terminology is motivated by the fact that if an inflatable b appears in the factorization of another semigroup element a, then there also exists factorizations of a with arbitrarily many factors b.

This suggests considering factorization of $a = b_1 b_2 \dots b_v$ into as many non-inflatable b_i as possible, if there are any such factorizations. If there is a finite maximum v, then perhaps it should be called the **volume** of a, if clear from context (such as if no other meanings of volume are applicable in S).

B.8 Division

This section discusses what it means to divide in a semigroup.

Diversion B.8.1. Strategies on how to divide (in various semigroups) are discussed in Chapter 5.

B.8.1 Dividers

Division is defined by means of a **divider**:

Definition B.8.1. *In a multiplicative semigroup (or magma) S, a* ***(right) divider*** *is binary operator*

$$/ : S^2 \to S : [d, b] \mapsto d/b$$

such that

$$((ab)/b)b = ab, \tag{B.8.1}$$

for all $a, b \in S$.

Diversion B.8.2. Definition B.8.1 is not entirely standard, but is an abstraction of the one of the properties of division in standard rational arithmetic.

Diversion B.8.3. Stricter, and laxer, definitions of division are discussed later in this section, §B.8. While Definition B.8.1 is most relevant to this report, other definitions may be more relevant outside this report.

Diversion B.8.4. Some terminology may help discussion.

The term **right** divider is mainly necessary to distinguish from **left dividers**, defined in §B.8.5. Whenever clear, call / a **divider**, and presume that **divider** means /.

The **input** to the divider is the pair $[d, b]$. If input $[d, b] = [ab, b]$ for some a, b, then it is **relevant** or **divisible** input, and $[a, b]$ is the **seed**, otherwise $[d, b]$ is an **irrelevant** or **indivisible** input. The output d/b does not matter for an irrelevant input $[d, b]$, and can modified to an arbitrary value, while still being a divider.

Diversion B.8.5. Many readers may be best off devising their own division algorithms to implement a divider $/$, perhaps perusing the strategies listed in Chapter 5.

The remainder of this section, §B.8, handles general properties of division, including more stringent definitions for division, and obvious minor generalizations.

B.8.2 Existence of dividers

Lemma B.8.1. *In each semigroup (or magma), a divider exists.*

Proof. If $S_{d,b} = \{a \in S : d = ab\}$ is non-empty, then let d/b be any element of $S_{d,b}$. Otherwise, let d/b be any value in S.

For any a, b, we have $a \in S_{ab,b}$, so $S_{ab,b}$ is non-empty, and therefore $(ab)/b \in S_{ab,b}$. Say $(ab)/b = a'$. Then $a' \in S_{ab,b}$, so $a'b = ab$. □

Diversion B.8.6. A contradiction to Lemma B.8.1 would seem to be division by zero, which is traditionally characterized as impossible in standard arithmetic.

A resolution to the seeming contradiction is that tradition (sometimes tacitly) uses a definition of division that is stricter than Definition B.8.1. For more on this, see §B.8.14.

Diversion B.8.7. The proof above is not constructive: it is merely existential. It can lead to a constructive algorithm: trial multiplication discussed in §5.1.

Diversion B.8.8. Logicians will recognize that the proof resorts to the Axiom of Choice, and may be able to confirm whether the Axiom of Choice is required, in the sense that the conclusion of the lemma implies the Axiom of Choice.

B.8.3 Alternative notations

Occasionally, we may need to consider more than one divider in a single multiplicative semigroup. In this case, we may use alternative symbols, such as one of

$$\div \quad \oslash \quad \varnothing \quad \emptyset \quad \% \quad \$ \quad \int \quad /'$$

with some appropriate explanation. So, for example, $\div$ is a divider if and only if $((ab) \div b)b = ab$, for all $a, b \in S$.

In other cases, we may consider modifying a divider $/$, starting with a given binary operation $/$, modifying it, but then using the same symbol $/$, with a convention similar to common computer programming practice where a variable is re-assigned a related value, as in $x = x + 1$.

Diversion B.8.9. Another notation for division is $:$, but this report uses $:$ so often formulas for sets and functions that this would be too confusing to use for division.

B.8.4 Subtraction

In additive semigroups, §B.1.3, notation and terminology is different, so the terminology and notation for division is adjusted accordingly as

- division becomes **subtraction**,
- a divider becomes a **subtracter**, and
- a/b becomes $a - b$ (or $a \ominus b$ in $S^{\oplus}$).

The main condition for a subtracter is therefore:

$$((a+b)-b)+b = a+b \tag{B.8.2}$$

The abstract notions of subtraction and division are identical: the only difference is notation. The remaining abstract considerations about division of this section (§B.8) apply equally well to subtraction, with only a suitable notational adjustment.

Diversion B.8.10. In specific semigroups, such as standard arithmetic, division and subtraction refer to distinct, specific operations, so distinct considerations apply.

B.8.5 Left dividers

Semigroups can be non-commutative, so swapping left and right leads to a definition of a left divider.

Definition B.8.2. *A **left divider** in a semigroup S (or magma), is a binary operator $\backslash : S^2 \to S : [a, b] \mapsto a\backslash b$ such that*

$$ba = b(b\backslash(ba)). \tag{B.8.3}$$

for all $a, b \in S$.

Diversion B.8.11. The notation for $\backslash$ for a left divider is rarely used in standard arithmetic. It may be used in the quasi-group theory.

The notation / for right dividers, by comparison, is more commonly used, often in standard integer arithmetic, and in some programming languages.

Diversion B.8.12. This report focuses right dividers when possible.

For general semigroups, when considering only one divider at time, the symmetry obtained by reversing the order of multiplication operands, converts left dividers to right dividers, so that restricting attention to right dividers is mostly without loss of generality.

For specific semigroups, the order of multiplication is fixed, so left and right dividers must be considered separately.

Diversion B.8.13. In a commutative semigroup (meaning $ab = ba$ for all $a, b \in S$), then we can define left dividers via right dividers simply by swapping the order of inputs: $b\backslash d = d/b$.

Diversion B.8.14. In general, despite constructions for left divider like the one noted above, it does not seem possible to construct left dividers from right dividers (or conversely). For example, semigroups can be constructed (see §**??**) from block ciphers, with one direction of division corresponding to decryption, which is easy, and the other direction of division corresponding to cryptanalysis, which is hard. (See §5.1.)

B.8.6 Notational dividers

Sometimes, a binary operator / fails to meet Definition B.8.1, yet is still relevant.

Definition B.8.3. *A **notational divider** in a multiplicative semigroup S (or magma) is binary operator with the same notation as used for a divider, typically,*

$$/ : S^2 \to S : [a, b] \mapsto a/b. \tag{B.8.4}$$

Formally, a notational divider is just a binary operation. Other divider symbols similar, such as $\div$ or $\oslash$, can be used for notational dividers. Notational subtracters and left dividers can also be considered.

Diversion B.8.15. The adjective **notational** here has expanded the previous meaning of **divider**, which is an exception to the best practice of adjective narrowing meanings.

In cases of confusion, a notational divider / meeting Definition B.8.1 can be called a **true** divider, or a **full** divider, if there is doubt whether it is just a notational divider.

B.8.6.1 Candidate dividers

Perhaps the most common case of a notational divider is a candidate divider: an operator which we will prove to be divider. Until the proof is complete, we only know that / is a notational divider. This report uses the convention often.

B.8.6.2 Faulty dividers

In some cases, a notational divider is not a really divider, which we formalize in this section.

Definition B.8.4. *A **faulty divider** in semigroup S (or magma) is a notational divider / that is not a (true) divider.*

However, in some cases, a faulty divider has as much impact as a divider, and there is little value in focusing on its faultiness.

B.8.6.3 Divider-set and divisible-set

Measuring the successfulness of a notational divider can be done through the following definition.

Definition B.8.5. *The **divider-set** of a notational divider / is the set*

$$\{[a,b] : ((ab)/b)b = ab\}. \tag{B.8.5}$$

A notational divider / is a (true) divider, per Definition B.8.1 precisely if its divider-set is S^2. Otherwise (if the divider-set is not S^2), the notational divider / is a faulty divider.

Diversion B.8.16. In some cases, we may focus on proving that the divider-set of / has certain properties, such as containing a particular subset, and ignore the rest of the divider-set. This means ignoring the question of whether / is faulty of not.

Diversion B.8.17. Elements of the divider-set do not represent inputs to /, but rather seeds to generate the relevant inputs to /, the pairs $[a,b]$ that generate input pairs $[ab,b]$ to / over which successful division is defined.

Diversion B.8.18. The **divisible-set** is

$$\{[d,b] : (d/b)b = b\}, \tag{B.8.6}$$

which is a subset of the inputs to /. If R is the divider-set of / and L is the divisible-set of /, then

$$L = \{[ab,b] : [a,b] \in R\} = \lambda(R),$$
$$R = \{[a,b] : [ab,b] \in L\} = \rho(L)$$

so the divider-set and divisible-sets are closed related.

(Despite the similar notations above, the sets above are related via different constructions: L is related from R by applying transformation $[a,b] \mapsto [ab,b]$ to each element of the set R, while R is defined as a subset of S^2 by restriction under the predicate $[ab,b] \in L$.)

If / is not faulty, then its divisible-set is $\{[ab,b] : a,b \in S\} = \lambda(S^2)$. A divider / can have a divisible-set that is not S^2, even if it is not faulty. If the divisible-set of / is S^2, then / is a divider because $\lambda(S^2) \subset S^2$ (and furthermore / is a pre-divider as in §B.8.10.)

B.8.6.4 Side-dividers

In some cases, a special class of potentially faulty, notational dividers has an impact near to a full divider.

Definition B.8.6. *A **side-divider** on semigroup S (or magma) is a notational divider / such that*

$$(((ab)/b)b)c = (ab)c \tag{B.8.7}$$

for all $a, b, c \in S$.

Every divider is a side-divider, but some side-dividers are not dividers (they are faulty).

Diversion B.8.19. Write $s \models t$ to mean that $su = tu$ for all $u \in S$. A side-divider is characterized by $((ab)/b)b \models ab$.

B.8.7 Unary dividers

To be revised.

Definition B.8.7. *A **unary multiplier** and **unary divider** for an element b in semigroup S (or magma) are the functions*

$$\times_b : S \to S : a \mapsto ab, \tag{B.8.8}$$

$$/_b : S \to S : a \mapsto a/b, \tag{B.8.9}$$

where / is a divider in S.

Diversion B.8.20. In applications to key agreement, the value of b is sometimes fixed, so considering $/_b$ – instead of the more general binary operator / – is sufficiently important.

The set S^S of functions on a set S is a semigroup under function composition. The unary divider is a middle inverse of the unary multiplier:

$$\times_b /_b \times_b = \times_b. \tag{B.8.10}$$

Similarly, if / is a post-divider, then $/_b \times_b = 1_S$, where $1_S : S \to S : s \mapsto s$. If / is a pre-divider, then $\times_b /_b = 1_S$.

Conversely, / is divider if and only if $/_b$ is a middle inverse of $\times_b$ for all b.

Dividing by b is difficult if and only if $\times_b$ is a **pre-image-resistant** function, also known as **one-way function**.

B.8.8 Feasible dividers

This report focuses on semigroups with efficient, or at least practical, multiplication algorithms.

Algorithm for division are important, but they need only be feasible, not necessarily efficient. In other words, even plausible division has an impact on security.

To be revised.

Security of associative key agreement depends on the feasibility, or efficiency, implementing of a (relative) divider: a **division algorithm**. This section formalizes these notions of efficiency, somewhat following the usual formalisms in cryptology.

Division algorithms in Chapter 5 are generally not at a level of detail discussed here, because they are described in terms large abstract classes of semigroups. The kind of details discussed in the section are left to the reader to the determine.

Definition B.8.8. *The* ***deterministic division problem*** *in S (relative to a set $T \subseteq S^2$) is the task of efficiently implementing a mid-divider (relative to T).*

In other words, the division problem is to provide a ***deterministic division algorithm****. The relevant inputs are (d, b) for $(d, b) = (ab, b)$ with $(a, b) \in T$. The output is d/b, which satisfies $(d/b)b = d$ for all relevant inputs.*

A deterministic division algorithm / is rated by two things:

- the extent of T,
- the computational cost of /.

The difficulty of the deterministic division problem is rated according to rating of (known) deterministic division problems.

In practice, we need to consider security against probabilistic algorithms, which do not necessarily provide the same outputs for given inputs. Because probabilities are involved, we can also consider the inputs to have a probability distribution.

Definition B.8.9. *A* ***division algorithm*** *in S is a probabilistic algorithm, written $/ : S^2 \to S : (d, b) \to d/b$.*

The success rate of S relative to a probability distribution[3] *(a, b) over S^2 is the probability:*

$$\Pr_{(a,b),\$} [(d/b)b = b \quad | \quad d = ab] \tag{B.8.11}$$

[3] Random variable, in other words.

where the probability is defined over the random choices of / and the random variable (a, b).

The ***division problem*** *in S is the task of implementing a a division algorithm /.*

The ***division difficulty*** *of S against a probability distribution (a, b) is rated by the computational cost and the success of (known) division algorithms.*

Diversion B.8.21. In theory, a probabilistic algorithm can be de-randomized by preselecting a fixed random set of choices (e.g. coin flips) that it uses. Finding an upper bound on the success rate of all such possible de-randomizations, would provide an upper bound on the success rate of probabilistic algorithms. In practice, probabilistic algorithms are sometimes easier to analyze.

Diversion B.8.22. Regarding the representation of semigroup inputs and outputs of the divider, we consider the epigram representation sufficient.

B.8.9 Post-dividers

Some dividers / obey a condition much stricter than Definition B.8.1.

Definition B.8.10. *A* ***post-divider*** *on a semigroup S (or magma) is a notational divider /*

$$(ab)/b = a, \tag{B.8.12}$$

for all $(a, b) \in S$.

Diversion B.8.23. The term **post** in post-divider refers to the fact that division is applied after multiplication in (B.8.12).

Alternative names for **post-divider** include **cancelling divider** and **retracting divider**.

Diversion B.8.24. Consider two post-dividers / and $\div$.

On relevant inputs of the form $[d, b] = [ab, b]$, the post-divider outputs are equal, because $d/b = a = d \div b$.

On irrelevant inputs of the form $[d, b] \neq [ab, b$ for all a, then it is sometimes possible for $d/b \neq d \div b$.

B.8.9.1 One post-divider means dividers are post-dividers

Every post-divider is a mid-divider:

Lemma B.8.2. *If / is a post-divider (relative to set $T \subseteq S^2$, then / is a mid-divider (relative to the same set T).*

Proof. Multiply both sides of the post-division equation $(ab)/b = a$ by b, to get the division equation $((ab)/b)b = ab$. □

In semigroups where cancellation fails, a post-divider fails to exist: if $d = a'b = ab$ with $a' \neq a$, then d/b can only take one of the values a or a'. If $d/b = a$, then $(a'b)/b = d/b = a \neq a'$, so / fails to be a post-divider.

Therefore, the converse of the previous lemma, which would be that every mid-divider is a post-divider, can fail. The following lemma essentially shows that the failure of this converse can only happen if a post-divider fails to exist.

Lemma B.8.3. *If an post-divider exists, then every mid-divider is a post-divider.*

Proof. Suppose that $\div$ is a post-divider, and that / is any (other) divider. For all a, b, we have

$$\begin{aligned}(ab)/b &= (((ab)/b)b) \div b \\ &= (ab) \div b \\ &= a\end{aligned}$$

where the first and third equations are due to $\div$ being a post-divider, and second is due to / be a divider. Hence, $(ab)/b = a$ for all a, b, making / a post-divider. □

From an algorithmic perspective, this lemmas means no extra effort needs to be expended to implement a post-divider, because if a post-divider exists, then all dividers are post-dividers.

Despite the lack of extra cost, post-division is sometimes significant for its greater impact on associative key agreement.

Diversion B.8.25. If a post-divider exists, then cancellation can be applied to show that a side-divider is a post-divider.

B.8.9.2 Partial post-dividers

To be completed.

A divider in which $(ab)/b = a$, only for $[a, b] \in T \subsetneq S^2$, it a **partial post-divider relative** to T.

Some results about post-divider carry through to partial post-dividers.

B.8.9.3 Post-divider in a finite semigroup

Having a post-divider is a very stringent and limiting condition on finite semigroups.

Lemma B.8.4. *If / is a post-divider in a finite semigroup, then*

$$d/b = db^{|S|-1} \tag{B.8.13}$$

for all $d, b \in S$.

Proof. Consider the elements db^i over the non-negative integers i, letting $db^0 = d$, by convention. The set

$$d\langle b\rangle = \{d, db, db^2, db^3, \dots\} \subseteq S$$

is finite, even though there are infinitely many i. So, not all db^i are distinct. Let m be the minimum non-negative integer such that

$$db^m = db^{m+p}$$

for some positive integer p. Suppose that $m \geq 1$. Dividing both sides by b gives

$$db^{m-1} = (db^m)/b = (db^{m+p})/b = db^{(m-1)+p}.$$

Since $m - 1 < m$, this would contradict m being the minimum such non-negative integer. Therefore $m = 0$, so

$$d = db^p$$

for a positive integer p. Now divide both sides by b, to get

$$d/b = (db^p)/b = (db^{p-1}b)/b = db^{p-1}.$$

Let u be an positive integer. Then

$$d/b = db^{up-1},$$

by induction on u, the case $u = 1$ being established above, the inductive step being from $db^{(u+1)p-1} = db^p b^{up-1} = db^{up-1}$.

Now, let p be the least positive integer such $d = db^p$.

Let $e \in S$ with $e \notin d\langle b\rangle$. By the same reasoning above, there exists a least positive integer q such that $e = eb^q$.

Consider the function $S \to Se : a \mapsto ae$. It is injective because / is a post-divider (if $ae = a'e$, then $a = (ae)/e = (a'e)/e = a'$). The sets and S and Se are finite. If Se were smaller than S, the function $a \mapsto ae$ could not be injective. There $S = Se$ and $d \in Se$, so $d = fe$ for some $f \in S$.

But then $db^q = feb^q = fe = d$. Since p is the smallest such positive integer, we have $p \leq q$.

Swapping d and e, shows that $q \leq p$. Therefore $p = q$.

The set $d\langle b\rangle$ and $e\langle b\rangle$ are disjoint, because if $db^i = eb^j$, for $0 \leq i, j \leq p-1$, then $e = eb^p = eb^j b^{p-j} = db^i b^{p-j} = db^{p+i-j} \in d\langle b\rangle$, a contradiction to the supposition that $e \notin d\langle b\rangle$.

Next, partition finite set S into subsets of the form above $d\langle b\rangle$ each with size p, by repeatedly taking elements not in previous subsets. There will be a finite number of such subsets, say u, and we will have $|S| = up$. □

So, in a finite semigroup with a post-divider, there is a exactly one divider / (it is peremptory, pursued further in this report). Furthermore, division can be implemented using power inversion.

Diversion B.8.26. The formula $d/b = db^{n-1}$ is not necessarily practical. For example, $n = |S|$ might be unknown, or n just too large (such as $n > 2^{2^128}$). Or, high powers of b might need representations (epigrams) so large that multiplication becomes impractical.

Diversion B.8.27. A post-divider seems to imply a more theoretical a specialized structure to a semigroup. It is necessarily a subsemigroup of the direct product of a left semigroup and a group (to be proven in section about constructing extension semigroups). In the finite case, it is whole the direct product (proven in the section on peremptory semigroups).

B.8.10 Pre-dividers

Definition B.8.11. *A divider / is a **pre-divider relative** to $T \subseteq S^2$, if*

$$(d/b)b = d, \tag{B.8.14}$$

for all $(d, b) \in T$.

Diversion B.8.28. The term **pre** in pre-divider refers to the division being performed before the multiplication in $(d/b)b$.

Other reasonable terms for a pre-divider / are a **fractional** divider, **preventive** divider, and **sectioning** divider.

Diversion B.8.29. If / is a pre-divider, then its divisible-set is S^2.

Diversion B.8.30. Generally, in terms of the divider's inputs d and b, verification for pre-dividers and (middle) dividers use the same equation:

$$(d/b)b = d.$$

The (middle) divider criteria differs in how d is chosen.

Essentially, the middle division problem is a **promise problem** [Kob98] version[4] version of the pre-division problem in which it is **promised** that the input d has the form $d = ab$ for some a.

A more formal comparison is provided below §B.8.10.1.

B.8.10.1 Comparison of pre-dividers and mid-dividers

On one hand, mid-divider and pre-divider are equivalent in the sense below.

Lemma B.8.5. *If $/$ is an pre-divider relative to T, then $/$ is a mid-divider relative to $T' = \{(a, b) : (ab, b) \in T\}$.*

Proof. If $(a, b) \in T'$, then let $(d, b) = (ab, b) \in T$. It follows that $(ab/b)b = (d/b)b = d = ab$, so $/$ is a mid-divider. □

Lemma B.8.6. *If $/$ is a mid-divider relative to T, then $/$ is an pre-divider relative to $T' = \{(ab, b) : (a, b) \in T\}$.*

Proof. If $(d, b) \in T'$, then $(d, b) = (ab, b)$ for some $(a, b) \in T$. Then $(d/b)b = (ab/b)b = ab = d$. □

Lemma B.8.7. *If there exists at least one pre-divider in S, then every mid-divider is an pre-divider.*

Proof. Let $\varnothing$ be a pre-divider and let $/$ a mid-divider. Then

$$\begin{aligned}(a/b)b &= (((a\varnothing b)b)/b)b \\ &= (a\varnothing b)b \\ &= a\end{aligned}$$

so $(a/b)b = a$, meaning that $/$ is a pre-divider. □

[4] I do not know how widely the term **promise problem** is used, but **premise** might be more neutral that **promise**, connotationally.

This lemma means that a mid-divider is pre-divider, if a pre-divider exists. (A similar result holds for post-dividers). No extra effort is required to find an pre-divider: if a pre-divider exists and mid-divider, then the mid-divider is already a pre-divider.

On the other hand, pre-dividers and mid-dividers are not equivalent, in the senses below.

Lemma B.8.8. *If $/$ is an pre-divider relative to set T and $(d, b) \in T$, then $d = ab$ for some a.*

Proof. Put $a = d/b$. By the pre-divider property, $ab = (d/b)b = d$. □

Therefore, if at least one pair $(d, b) \in T$ has the property that $d \neq ab$ for all a (in words, d is not left-divisible by a), then no $/$ can be an pre-dividers relative to a set T. In other words, a single instance of non-divisibility can cause non-existence of an pre-divider.

The existence of mid-dividers is much different from pre-dividers. Mid-dividers always exist, but a pre-divider exists only if every element is divisible (on the right) by every other element.

B.8.10.2 Left dividers from pre-dividers

A weak form of commutativity can be used to construct left dividers from a pre-divider.

Lemma B.8.9. *If $/$ is a pre-divider, and if d/d commutes with b for all $d, b \in S$, then $b\backslash d = ((d/d)/b)d$ is a left divider.*

Proof. Let $d = ba$ and compute:

$$\begin{aligned}
b(b\backslash(ba)) &= b(b\backslash d) \\
&= b(((d/d)/b)d) \\
&= b(((d/d)/b)(ba)) \\
&= b(((d/d)/b)b)a \\
&= (b(d/d))a \\
&= ((d/d)b)a \\
&= (d/d)(ba) \\
&= (d/d)d \\
&= d \\
&= ba
\end{aligned}$$

so that $\backslash$ meets the definition of a left mid-divider. □

B.8.10.3 Comparison of pre-dividers to post-dividers

This section compares pre-dividers to post-dividers.

Lemma B.8.10. *In a finite semigroup,*

- *every post-divider is a pre-divider, and*
- *every pre-divider is a post-divider.*

Proof. If a divider $/$ is a post-divider (pre-divider), then function $a \mapsto ab$ is injective (surjective).

In a finite set, a function is injective if only if it is surjective.

If $a \mapsto ab$ is surjective, there exists an pre-divider, say $\varnothing$. So, if $/$ is a post-divider, then there exists an pre-divider. Similarly, if there is an pre-divider, then there exists a post-divider.

Now suppose that $/$ is a post-divider and $\varnothing$ is an pre-divider. Then $a/b = ((a\varnothing b)b)/b = a\varnothing b$, so the two dividers are identical binary operations.

Therefore, the post-divider equals the pre-divider, thus it is also a pre-divider. Similarly, the pre-divider is also a post-divider. □

Lemma B.8.11. *In a commutative semigroup, every pre-divider is a post-divider.*

Proof. Let $/$ be a right pre-divider. Then $\backslash$ defined $(b\backslash d) = d/b$ is a left pre-divider (§B.8.5), essentially because the semigroup is commutative $(b(b\backslash d) = (d/b)b = d.)$

Lemma 6.2.3 (special case of Green's 1951 theorem that an H-class is either a subgroup or disjoint from its square), says that a semigroup that has both a left pre-divider and a right pre-divider is a group. In a group, cancellation holds, so a post-divider exists, and every divider is a post-divider. □

Recall that e is called **idempotent** if $ee = e$.

Lemma B.8.12. *In a semigroup with at least one idempotent, every pre-divider is a post-divider.*

Proof. Let $/$ be a pre-divider, let $e \in S$ be an idempotent, and let $q = e(e/b)$. Then $qbq = q$, because

$$\begin{aligned} qbq &= (e(e/b))bq \\ &= e((e/b)b)q \\ &= eeq \\ &= eq \\ &= ee(e/b) \\ &= e(e/b) \\ &= q. \end{aligned}$$

Use this property of q to compute:

$$\begin{aligned} (ab)/b &= (((ab)/b)/q)q \\ &= (((ab)/b)/q)(qbq) \\ &= ((((ab)/b)/q)q)(bq) \\ &= (((ab)/b)(bq) \\ &= ((ab)/b)b)q \\ &= (ab)q \\ &= a(bq) \\ &= ((a/q)q)(bq) \\ &= (a/q)(q(bq)) \\ &= (a/q)q \\ &= a \end{aligned}$$

which proves that $(ab)/a = a$, so $/$ is a post-divider. □

Diversion B.8.31. This proof is adapted from similar proofs in group theory, but likely unnecessarily long.

Diversion B.8.32. Every non-empty finite semigroup has an idempotent, so this proves again that every pre-divider in a finite semigroup is a post-divider.

(To see that non-empty finite semigroup has an idempotent, let $b \in S$, note there exists positive integers h, p with $b^h = b^{h+p}$, and the $e = b^{hp}$ is idempotent.)

A partial converse is the following.

Lemma B.8.13. *If semigroup has a divider $/$ that is both a pre-divider and a post-divider, then b/b is idempotent for all $b \in S$.*

Proof. Calculating directly

$$\begin{aligned}(b/b)(b/b) &= (((b/b)(b/b))b)/b \\ &= ((b/b)((b/b)b))/b \\ &= ((b/b)b)/b \\ &= (b/b),\end{aligned}$$

where the first equation is due to post-division, the second due to associativity, the third due to pre-division, while the fourth can be deduced from either pre-division (inside the parentheses) or post-division (outside the parentheses). □

Diversion B.8.33. The calculation in the proof above might become clearer writing $e = b/b$, noting that $eb = (b/b)b = b$, and then writing $ee = ((ee)b)/b = (e(eb))/b = (eb)/b = e$.

Diversion B.8.34. In non-empty semigroups, existence of an idempotent element is equivalent to the condition that every pre-divider is a post-divider.

Diversion B.8.35. Baer–Levi semigroups are known to have pre-dividers but no idempotents, as reviewed below.

Recall that N^N is the semigroup functions $f : N \to N$ under composition. The Baer–Levi semigroup S over an infinite space N, is a subsemigroup of N^N, consisting of injective functions f such that $N - f(N)$ is an infinite set.

To show that each $f \in S$ is not idempotent, consider that $f(N) \neq N$, since $N - f(N)$ is infinite. Since f is injective, we can apply f to both sides of inequality to get another strict inclusion $f(f(N)) \neq f(N)$, which means $ff \neq f$ (so f is not idempotent).

To see that the Baer–Levi semigroup S has a pre-divider (if N is countable), reason as follows. For each $a \in S$, define a bijective function $a' : N \to N - a(N)$, which exists since $N - a(N)$ is an infinite countable set. Define a divider / by the rule

$$(a/b)(n) = \begin{cases} a(m) & \text{if } n = b(m) \text{ for some } m, \\ a'(n) & \text{if } n \neq b(m) \text{ for all } m. \end{cases}$$

It just remains to show that $a/b \in S$, and $(a/b)b = a$.

Firstly, $(a/b)(n)$ is uniquely defined for all n, because either $n \in b(N)$ or not, and if $a \in b(N)$ then injectivity of b means that there exists exactly one m such $n = b(m)$.

To prove injectivity of a/b, suppose $(a/b)(n) = (a/b)(n')$. Injectivity means to show that $n = n'$. Since $a(N)$ and $a'(N)$ are disjoint, both n and n' must fall into the same case of the definition of a/b.

- In the first case, $(a/b)(n) = a(m)$ and $(a/b)(n') = a(m')$ for some $m, m' \in b(N)$. Since a is injective, we get $m = m'$, which implies $n = b(m) = b(m') = m'$, as desired.

- In the second case, $(a/b)(n) = a'(n)$ and $(a/b)(n') = a'(n')$, which means that $a'(n) = a'(n')$. But a' is bijective, so $n = n'$.

To prove that $N - (a/b)(N)$ is infinite, note that $N - (a/b)(N)$ contains the set $a'(b(N))$, which is infinite because b and a' are injective.

The latter two paragraphs show that a/b belongs to the Baer–Levi semigroup S.

To show that $(a/b)b = a$, evaluate at any $m \in N$, so $((a/b)b)(m) = (a/b)(b(m)) = a(m)$.

This pre-divider is not unique. For example, it can vary with the choice of a', and a' need not be bijective, injective suffices.

This pre-divider is not a post-divider, because of Lemma B.8.13 and the fact that the semigroup has no idempotent.

Diversion B.8.36. An alternative description of the Baer–Levi semigroups for the set $N = \{0, 1, 2, \dots\}$ of natural numbers can also be described as the set of function of the form $n \mapsto p(2n)$ where p is a permutation of the natural numbers.

Writing $N! \subseteq N^N$ for the set of permutations of N, and $2_N : n \mapsto 2n$ with $2_N \in N^N$, then the Baer–Levi semigroups is $(N!)2_N$.

Using the previous definition of a', define $p_a \in N^N$ by

$$p_a(n) = \begin{cases} a(n/2) & \text{if } n \text{ even} \\ a'((n-1)/2) & \text{if } n \text{ odd} \end{cases}$$

Then $a = p_a 2_N$.

This suggests find $p_{a/b}$ from p_a and p_b.

To be completed.

Diversion B.8.37. Many natural semigroups are commutative, finite, or have an idempotent. Arguably, it follows that, it is quite natural for a pre-divider to also be a post-divider.

The converse implication fails in many natural semigroups. For example, consider the positive integers under multiplication. Standard integer division is a post-divider, but it is not a pre-divider, because there are no fractions among the integers.

Diversion B.8.38. If $b \in bS$ for some $b \in S$, then every pre-divider is a post-divider, because if $b = bc$, then c is idempotent, with the lemma above applies. To see that c is idempotent, calculate $cc = ((c/b)b)c = (c/b)(bc) = (c/b)b = c$. Formalizing this:

Lemma B.8.14. *If S is a semigroup with an pre-divider $/$, and there exists $c \in S$ such that the function $\rho_c : a \mapsto ac$ has a fixed-point, then $/$ is a post-divider.*

Consequently, any semigroup which has a right multiplication meeting the hypotheses of a fixed-point theorem has the property that an pre-divider is a post-divider.

For example, Brouwer's fixed-point theorem applies if S is also a compact, convex set such that at least one right multiplication is continuous. This can be generalized using Lefshetz fixed-point theorem, a special case of which is if S has the topology of a CW complex, with a non-zero Euler characteristic, and at least one right multiplication operation is continuous in this topology.

Diversion B.8.39. Yet another observation that may help resolve the comparison of pre-dividers and post-dividers is the following.

Lemma B.8.15. *If a semigroup S has a pre-divider $/$ and there is a maximal subset of the form bS, then $/$ is a post-divider.*

Proof. Suppose that bS is maximal among subsets of that form. (This means that, for all $a \in S$, if $bS \subseteq aS$, then $bS = aS$.)

Since $/$ is a pre-divider, $(b/b)b = b$, which means that $bS = (b/b)bS \subseteq (b/b)S$, so $bS = (b/b)S$.

Let $e = (b/b)(b/b)$. We claim that e is idempotent.

Firstly, note that $e = (b/b)(b/b) \in (b/b)S = bS$. Therefore, $e = bs$ for some $s \in S$, consequently

$$ee = ebs = (b/b)(b/b)bs = (b/b)((b/b)b)s = (b/b)(b)(s) = ((b/b)b)s = bs = e.$$

By Lemma B.8.12, we are done. □

Diversion B.8.40. For any $b \in S$, write $1_b = b/b$. Then $bS \subseteq 1_bS \subseteq 1_{1_b}S \subseteq 1_{1_{1_b}}S \subseteq \dots$. The proof of the previous lemma also show that if S has no idempotent, then that sequence of sets is strictly increasing.

Diversion B.8.41. If S is a semigroup with a pre-divider but no idempotents, then the relation $a < b$ defined by $a \in bS$ is a strict partial order.

Diversion B.8.42. The resolution is to be completed.

B.8.10.4 Aside: quasi-groups

To be revised.

A **quasi-group** S is a magma with a left and right pre-divider.

The previous results show that if S both a quasi-group and a semigroup, then it is a group.

B.8.11 Aside: divisibility

Divisibility of integers can be generalized to any semigroup (or magma):

Definition B.8.12. *For b, d in a multiplicative semigroup S (or magma), say b* ***right-divides*** *d, and d is* ***right-divisible*** *by b, write*

$$b|d \tag{B.8.15}$$

precisely if there exists a such that $d = ab$.

The relation of divisibility is transitive in semigroups:

Lemma B.8.16. *In a multiplicative semigroup, if $b|d$ and $d|f$, then $b|f$.*

Proof. There exists a such that $d = ab$, and there exists e such that $f = ed$. Therefore $f = e(ab) = (ea)b$, so there exists $o = ea$ such that $f = ob$, meaning $b|f$. □

Diversion B.8.43. Divisibility can be considered a function $S^2 \to \{0, 1\}$, where 1 means true, or right-divisible, and 0 means false.

Definition B.8.12 is existential, referring to the existence of a such that $d = ab$.

This reports emphasizes constructions over existence. So, right divisibility can also be characterized somewhat more constructively by using a right divider operator.

Lemma B.8.17. *In a semigroup (or magma), $b|d$ if and only if*

$$(d/b)b = d,$$

for any divider $/$.

Proof. If $b|d$, then $d = ab$ for some a, so $(d/b)b = ((ab)/b)b = ab = d$.

If $(d/b)b = d$, then $d = ab$ for $a = (d/b)$, so $b|d$. □

Diversion B.8.44. The choice of divider $/$ does not matter in this characterization divisibility: any divider can be used.

Diversion B.8.45. The lemma above may be interpreted as a reductive algorithm to computes the Boolean divisibility function using a divider function, as in $(b|d) = (d/b)b \overset{\circ}{=} d$ (where $x \overset{\circ}{=} y$ is 1 if $x = y$ and 0 otherwise).

Left divisibility can also be defined, and is characterized by left division, so that b **left-divides** e if $(e\backslash b)b = b$.

B.8.11.1 Aside: division set

Tentative.

Definition B.8.13. *The* ***division-set*** *of d over b is defined,*

$$d//b = \{a : ab = d\}, \tag{B.8.16}$$

as an operator $// : S^2 \to 2^S$.

An operator / on semigroup S is a divider if and only if

$$(ab)/b \in (ab)//b$$

for all $a, b \in S$. In particular, given a division-set operator //, a divider can be constructed by setting d/b to be a random element of $d//b$.

A post-divider / exists if and only if:

$$|d//b| \leq 1$$

for all $d, b \in S$. A pre-divider / exists if and only if:

$$|d//b| \geq 1$$

for all $d, b \in S$.

If the set $d//b$ is small, and an algorithm can compute $d//b$ quickly, then the impact of // on key agreement is similar to the impact of a post-divider, because Alice's key $a \in d//b$. If Alice re-uses a outside of a single key agreement session, an attacker with run trial search on the set $d//b$ for a, reducing the security effective security of a. (The use of a in a single key agreement session is already compromised by any divider /.)

More generally, consider the set 2^S (of subsets of S) as a semigroup, under multiplication of sets. Then define an operator // on 2^S as follows:

$$D//B = \{a : aB \subseteq D\}. \tag{B.8.17}$$

The division-set operator on S can be re-defined in terms of this 2^S operator:

$$d//b = \{d\}//\{b\}. \tag{B.8.18}$$

The operator // on 2^S is a divider for 2^S. It is maximal in the sense that if / is any other divider on 2^S, then

$$(AB)/B \subseteq (AB)//B. \tag{B.8.19}$$

B.8.12 Aside: Green's relations

Green noticed that five definitions of mutual divisibility between semigroup elements lead to equivalence relations useful for analyzing the structure of semigroups. For multiplicative integers, the mutual divisibility equivalence classes are the sets:

$$\{0\}, \{1, -1\}, \{2, -2\}, \{3, -3\}, \ldots$$

where, for example, 3 is equivalent to -3, because they divide each other exactly.

Green defined five equivalence relations defined the relations using set operations including set-multiplication, but some of the relations can be stated using the divider notation, as used this report.

Definition B.8.14. *Define aLb precisely if $a = b$ or if both*

$$a|b, \tag{B.8.20}$$

$$b|a. \tag{B.8.21}$$

Diversion B.8.46. Green's choice of the letter L, as in **left**, is a mirror image in interpretation of this report's terminology of this being mutual **right** divisibility. In $a|b$ meaning $a = xb$, the value b is on the right, but the value x is on the left. For consistency, with this report other notations, such as right divider /, this has been called right divisibility.

Lemma B.8.18. *In a multiplicative semigroup, the relation L is an equivalence relation.*

Proof. To prove that relation L is an equivalence relation means to prove

$$aLa, \tag{B.8.22}$$

$$aLb \implies bLa, \tag{B.8.23}$$

$$aLb, bLc \implies aLc, \tag{B.8.24}$$

in other words, to prove that L is reflexive, symmetric and transitive.

By definition, aLa, since aLb if $a = b$, so L is reflexive.

If aLb, then either $a = b$, so $b = a$ so bLa, or $a|b$ and $b|a$, which means that bLa, since this system of two conditions is symmetric in the variables a and b. Hence L is symmetric.

If aLb and bLc, and $a = b$ or $b = c$, then aLc by substitution. Otherwise $a \neq b$ and $b \neq c$, so the four divisibility conditions hold. In particular, $a|b$ and $b|c$, so $a|c$, because divisibility is transitive in semigroups. Similarly, $c|b$ and $b|a$, so $c|a$. Therefore aLc. □

Diversion B.8.47. An equivalence class of L is called an L-class. The L-class of a is often written as L_a.

Diversion B.8.48. Reversing left and right divisibility gives Green's R-relation. Combining both these relations, L and R, gives the H-relation. So, aHb precisely if $a = b$ or if

$$\begin{aligned}(a/b)b &= b,\\ (b/a)a &= a,\\ a(a\backslash b) &= b,\\ b(b\backslash a) &= a.\end{aligned}$$

Diversion B.8.49. Green also defined two other relations, the D and J relations, which are not discussed in this report.

B.8.13 Co-multiples and co-divisors

TO BE RE-WRITTEN!

B.8.13.1 Co-multiples

Definition B.8.15. *The (right)* ***co-multiples*** *set of a set* $T \subseteq S$ *is*

$$T^{//} = \{d : (d/b)b = d, \forall b \in T\}. \tag{B.8.25}$$

Equivalently:

$$\begin{aligned}T^{//} &= \bigcap_{b \in S} Sb\\ &= \{d : b|d, \forall b \in T\}.\end{aligned}$$

The set $T^{\backslash\backslash}$ of left co-multiples is defined similarly:

$$T^{\backslash\backslash} = \{d : b(b\backslash d) = d, \forall b \in T\}.$$

In constructions of some algorithms, such as matrix divisions, it suffices to have an algorithm that computes one co-multiple.

Definition B.8.16. *A* ***co-multiple*** *operator is an operator* $2^S \to S : T \mapsto T^/$*, such that* $T^/ \in T^{//}$ *whenever* $T^{//} \neq \{\}$*.*

Diversion B.8.50. Some example co-multiple operators:

- If $0 \in S$, then $T^/ = 0$ for all T defines a co-multiple operator.
- If / is a pre-divider, then any operator $T \mapsto T^/$ is a co-multiple operator.
- If S is commutative, then for finite T, there exists a co-multiple operator with $\{b_1, \ldots, b_n\}^/ = b_1 b_2 \ldots b_n$.
- Given an existing co-multiple operator $T \mapsto T^/$, a second $T \mapsto T'$ may be defined as $T' = aT^/$ for any $a \in S$.

B.8.13.2 Ore semigroups

To be completed, or corrected.

Definition B.8.17. *A (right)* ***Ore*** *semigroup is one which $T^{//}$ is non-empty for every finite non-empty subset.*

B.8.13.3 Cross-multipliers

Given $a, b \in S$ and an co-multiple operator $T \mapsto T^{/}$, a **cross-multiplier** from a to b can be defined as:

$$a \ast/\, b = \{a, b\}^{/}/b \tag{B.8.26}$$

Diversion B.8.51. The action of co-multiple operator $\cdot^{/}$ on two element subsets can be recovered from the cross-multiplier: $\{a, b\}^{/} = (a \ast/\, b)b$.

The cross-multiplier has the characterizing property that

$$(a \ast/\, b)b = (b \ast/\, a)a,$$

which can be useful in matrix division to cancel variables, whenever there exists x, y with $xb = ya$. This property also provides an equivalent characterization of cross-multipliers.

A cross-multiplier $\ast/$ is **strict** if $a \neq b$ implies that $a \ast/\, b \neq b \ast/\, a$.

Diversion B.8.52.

Diversion B.8.53. Consider the problem of constructing a left double-base-tester (§3.4.3.3) τ in an associative scheme. The input is $[e, b, b']$ and the output is $[d, f, f']$ such that $f = d \wedge_b d$ and $f' = d \wedge'_b f'$.

To that end, let $[d, f, f'] = [(b' \ast/\, b)b, (b' \ast/\, b)e, (b \ast/\, b')e]$, where $\ast/$ is a strict cross-multiplier. If $f \neq f'$, then this provides a double-base-tester.

To be completed.

The Diffie–Hellman semigroup is commutative, so the $a \ast/\, b = a$ defines a strict cross-multiplier. But, this fails to yield a double-base-tester because $f = f'$ above.

B.8.13.4 Co-divisors

Definition B.8.18. *The right* ***co-divisors*** *set of a set $T \subseteq S$ is*

$$T_{//} = \{b : (d/b)b = d, \forall d \in T\}. \tag{B.8.27}$$

B.8.14 Aside: division by zero

The traditional view that division by zero is impossible can be rephrased as the impossibility of post-division by zero or pre-division by zero, made precise below.

Lemma B.8.19. *If semigroup S has absorbing element 0, and $/$ is a post-divider or a pre-divider, then $S = \{0\}$.*

Proof. Consider any $a \in S$. Let $b = 0$.

If $/$ is a post-divider and $a \in S$, then

$$\begin{aligned} a &= (ab)/b \\ &= (a0)/0 \\ &= (0)/0 \\ &= (00)/0 \\ &= (0b)/b \\ &= 0. \end{aligned}$$

If $/$ is a pre-divider , then $a = (a/b)b = (a/b)0 = 0$.

In either case, $a = 0$ for all $a \in S$, meaning $S = \{0\}$. □

In other words, if $0 \in S$, then a divider $/$ is a not post-divider or a pre-divider. But a divider, in the weak sense of middle division, still exists, as always, despite the existence of 0. So, middle division does not forbid division by zero. Furthermore, division by zero can take any value:

Lemma B.8.20. *Suppose that semigroup S has absorbing element 0, and that $e, z \in S$. There exists a divider $/$ such that such $z/0 = e$.*

Proof. Let $\div$ be a divider for S, which we know exists. Define $/$ by

$$d/b = \begin{cases} e & \text{if } [d, b] = [z, 0], \\ d \div b & \text{if } [d, b] \neq [z, 0]. \end{cases} \tag{B.8.28}$$

Consider any $a, b \in S$. We aim to prove that $((ab)/b)b = ab$. Write $d = ab$.

If $[d, b] \neq [z, 0]$, then $((ab)/b)b = (d/b)b = (d \div b)b = ((ab) \div b)b = ab$, as required.

If $[d, b] = [z, 0]$, then $((ab)/b)b = (z/0)0 = e0 = 0 = a0 = ab$. □

Diversion B.8.54. Perhaps, for some special semirings and for some stricter definitions of division (such as those distributive over addition), the intuitive idea that $1/0 = \infty$, in the sense that ∞ is an additively absorbing element, might hold.

B.8.15 Aside: traditional division examples

Examples of the various dividers under traditional arithmetic may help illustrate this report's notions of division.

- Positive rational numbers under standard multiplication have a standard divider / that is both post-divider and pre-divider.

- Positive integers under standard multiplication have a post-divider /, but have no pre-divider. (For example, no integer result for the division 4/3 meets the pre-divider rule that $(4/3)3 = 3$.)

- Non-negative integers (natural numbers) under standard multiplication have a divider / but have no pre-divider or post-divider, because 0 is included.

B.8.16 Peremptory dividers

A special case is a semigroup with a unique divider.

Definition B.8.19. *A **peremptory** divider in a semigroup S is a divider such that there are no other dividers in S.*

*A **peremptory** semigroup is a semigroup with only one divider.*

Diversion B.8.55. A left semigroup L (where $ab = a$ for all $a, b \in L$) is peremptory, because $d/b = (db)/b = ((db)/b)b = d$, for any divider / and any $d, b \in L$.

Diversion B.8.56. Any group G is a peremptory, because $d/b = ((db^{-1}b)/b)(bb^{-1}) = ((db^{-1})b)b^{-1} = db^{-1}$, for any $d, b \in G$ and any divider G. (The inverse element b^{-1} is uniquely defined in a group, so the division $d/b = db^{-1}$ is uniquely defined.

Diversion B.8.57. Non-zero rationals under multiplication is a group from elementary arithmetic, and therefore a peremptory semigroup.

Diversion B.8.58. Positive reals under multiplication is a group used often in applied mathematics with its peremptory divider is read **per**, as in kilometers per hour (km/h).

Lemma B.8.21. *A divider / is peremptory if and only if / is both a post-divider and a pre-divider.*

Proof. Suppose that / is both a pre-divider and a post-divider (so $(ab)/b = a$ and $(a/b)b = a$ for all a, b). Suppose that $\div$ is a divider (so $((ab) \div b)b = ab$

for all a, b). Then

$$\begin{aligned} a \div b &= ((a \div b)b)/b \\ &= ((((a/b)b) \div b)b)/b \\ &= ((a/b)b)/b \\ &= a/b \end{aligned}$$

for all a, b: so, $\div = /$. All dividers are $/$, so $/$ is the only divider, so $/$ is peremptory.

Suppose that $/$ is peremptory. Consider any $a', b' \in S$. Define an operator $\div$ by the rule:

$$d \div b = \begin{cases} a' & \text{if } [d, b] = [a'b', b'] \\ d/b & \text{if } [d, b] \neq [a'b', b'] \end{cases}$$

Then $\div$ is a divider, because:

$$\begin{aligned} ((ab) \div b)b &= \begin{cases} a'b & \text{if } [ab, b] = [a'b', b'] \\ ((ab)/b)b & \text{if } [ab, b] \neq [a'b', b'] \end{cases} \\ &= \begin{cases} ab & \text{if } [ab, b] = [a'b', b'] \\ ab & \text{if } [ab, b] \neq [a'b', b'] \end{cases} \\ &= ab. \end{aligned}$$

But $/$ is peremptory, so $/ = \div$ and therefore $(a'b')/b' = (a'b') \div b' = a'$. This holds for any $a', b' \in S$, so $/$ is a post-divider.

Next, let $f : S \to S$ be any function, and define a binary operator $\div$ by the rule:

$$d \div b = \begin{cases} d/b & \text{if } (d/b)b = d \\ f(d/b) & \text{if } (d/b)b \neq d \end{cases}$$

Now $\div$ is a divider, because

$$\begin{aligned} ((ab) \div b)b &= \begin{cases} ((ab)/b)b & \text{if } ((ab)/b)b = ab \\ (f((ab)/b))b & \text{if } ((ab)/b)b \neq ab \end{cases} \\ &= ((ab)/b))b \\ &= ab. \end{aligned}$$

Because $/$ is peremptory, $/ = \div$. Suppose $(d/b)b \neq d$, then choose $f : S \to S$ such that $f(d/b) \neq d/b$, which is possible if $|S| \geq 2$. Then $d \div b = f(d/b) \neq$

d/b, giving $/ \neq \div$, a contradiction. Since this holds for all d, b, we conclude that / is a pre-divider.

If $|S| \leq 1$, then any divider, including /, is a pre-divider.

Therefore, if / is peremptory, it is both a post-divider and pre-divider. $\square$

Lemma B.8.22. *Let S be peremptory. For $a, b \in S$,*

$$a1_b = a(b/b) = a,$$

writing $1_b = b/b$, call this the element ***ridentity*** *of b.*

Proof. Compute

$$\begin{aligned} a(b/b) &= (a(b/b)b)/b \\ &= (ab)/b \\ &= a. \end{aligned}$$

$\square$

Peremptory semigroups are closed under direct products of semigroups.

Lemma B.8.23. *If S and T are peremptory, then so is $S \times T$.*

Proof. Recall that $S \times T = \{[s,t] : s \in S, t \in T\}$ with multiplication rule $[s,t][u,v] = [su, tv]$.

Define a divider in $S \times T$ by

$$[d,e]/[b,c] = [d/b, e/c]$$

where the dividers / inside the brackets are the unique dividers of S and T.

The operator / in $S \times T$ is both a pre-divider and a post-divider, because in each coordinate it is both a pre-divider and post-divider:

$$\begin{aligned} ([a_1,a_2][b_1,b_2])/[b_{1,2}\,] &= [(a_1b_1)/b_1, (a_2b_2)/b_2] = [a_1,a_2] \\ ([a_1,a_2]/[b_1,b_2])[b_{1,2}\,] &= [(a_1/b_1)b_1, (a_2/b_2)b_2] = [a_1,a_2], \end{aligned}$$

and therefore it / is peremptory in $S \times T$, so $S \times T$ is a peremptory semigroup. $\square$

In particular, if L is a left semigroup and G is a group, then $L \times G$ is peremptory semigroup. A converse of this is true, up to an isomorphism (efficient relative to division and multiplication).

Lemma B.8.24 (Rees?)**.** *A peremptory semigroup S is isomorphic to $L \times G$ where L is a left semigroup and G is group.*

An isomorphism can be given by

$$m : S \to L \times G : a \mapsto [(a/a), (g/g)a] = [1_a, 1_g a],$$

where g is some fixed element of S, and $L = \{a/a : a \in S\}$ and $G = \{(g/g)a : a \in S\}$ are subsemigroups of S, with L being a left semigroup, and G being a group.

Proof. Sets L and G are subsemigroups of S because, being closed under multiplication, as shown next. For $1_a, 1_b \in L$, we have $1_a 1_b = 1_a \in L$. For $1_g a$ and $1_g b$ in G, we have that $(1_g a)(1_g b) = (1_g c) \in G$, for $c = a(g/g)b$, so G is also closed in S.

Map m is a semigroup morphism, because it is multiplicative, as computed below:

$$\begin{aligned}
m(a)m(b) &= [1_a, 1_g a][1_b, 1_g b] \\
&= [1_a 1_b, 1_g a 1_g b] \\
&= [1_a, 1_g ab] \\
&= [(1_a(ab))/(ab), 1_g ab] \\
&= [(((a/a)a)b)/(ab), 1_g ab] \\
&= [(ab)/(ab), 1_g ab] \\
&= m(ab)
\end{aligned}$$

Map m is injective, since its input is determined by its output, as shown next. If $m(a) = [b, c]$, then $a = bc$, because $bc = 1_a 1_g a = 1_a = (a/a)a = a$.

Map m is surjective, because any $h \in G$ can be put in the form $h = [1_b, 1_g c] \in L \times G$, for some $b, c \in S$, and then for $a = 1_b c$, we have

$$\begin{aligned}
m(a) &= [1_a, 1_g a] \\
&= [a/a, 1_g(1_b c)] \\
&= [(1_b c)/a, (1_g 1_b)c] \\
&= [((1_b 1_b)c)/a, 1_g c] \\
&= [(1_b(1_b c))/a, 1_g c] \\
&= [(1_b a)/a, 1_g c] \\
&= [1_b, 1_g c] \\
&= h.
\end{aligned}$$

Map m is an isomorphism because it is a bijective morphism.

Semigroup L is a left semigroup, because that $1_a 1_b = 1_b$ for all $1_a, 1_b \in L$.

Semigroup G has an identity element, namely $1_g = 1_g 1_g \in G$. For any $1_g a$, we have $1_g(1_g a) = (1_g 1_g)a = a$ and $(1_g a)1_g = 1_g(a 1_g) = 1_g a$.

Semigroup G has a group inverse operation defined

$$h^{-1} = 1_g/h$$

for all $h \in G$. To see that this defines an inverse compute as follows:

$$\begin{gathered} h^{-1}h = (1_g/h)h = 1_g \\ hh^{-1} = h(1_g/h) = (h(1_g/h)h)/h = (h1_g)/h = h/h \end{gathered}$$

and then note that $h = 1_g k$ for some k, so $h = 1_g h$, mean $h/h = (1_g h)/h = 1_g$. □

Diversion B.8.59. More abstractly, we can let L be the set of distinct subsets aS, and set G be the set of functions defined by right multiplication in S.

Diversion B.8.60. In a peremptory semigroup, every element b has a co-mutual inverse $q = (b/b)/b$.

In a finite peremptory semigroup, every element b has torsion, meaning $b = b^{p+1}$ for some positive integer p called the period. If the entirety of S can be implemented by a quantum computer, then Shor's period-finding algorithm might apply to find p, and therefore its co-mutual inverse $q = b^{2p-1}$, and therefore, division by b.

In other words, a finite peremptory semigroup is likely insecure for key agreement.

B.8.17 Co-associativity

To be revised.

Dividers are generally not commutative or associative. Nevertheless, nice dividers are co-associative with multiplication in the sense of the next lemma.

Lemma B.8.25. *If / is both a pre-divider and post-divider, then*

$$a/(b/c) = (ac)/b, \tag{B.8.29}$$

$$(a/b)/c = a/(cb), \tag{B.8.30}$$

$$(ab)/c = a(b/c). \tag{B.8.31}$$

Proof. Computing directly:

$$\begin{aligned} a/(b/c) &= ((a/(b/c))b)/b \\ &= ((a/(b/c))(b/c)c)/b \\ &= (ac)/b, \end{aligned}$$

and

$$\begin{aligned}(a/b)/c &= ((a/b)/c)(cb))/(cb)\\ &= ((a/b)b)/(cb)\\ &= a/(cb),\end{aligned}$$

and

$$\begin{aligned}(ab)/c &= (a((b/c)c))/c\\ &= ((a(b/c))c)/c\\ &= a(b/c),\end{aligned}$$

using associativity of multiplication and the pre-divider and post-divider rules. □

Diversion B.8.61. The lemma holds even if / is only a post-divider, provided that we restrict all the inputs to / to be relevant inputs, meaning each division d/b has the form $d = eb$ for some e.

- Suppose $b = ec$ and $a = ue$. Then $a/(b/c) = (ue)/((ec)/c) = (ue)/e = u = (ub)/b = (uec)/b = (ac)/b$.
- Suppose $a = ecb$. Then $(a/b)/c = ((ecb)/b)/c = (ec)/c = e = (e(cb))/(cb) = a/(cb)$.
- Suppose $b = ec$. Then $(ab)/c = (aec)/c = ae = a((ec)/c) = a(b/c)$.

Diversion B.8.62. Co-associativity also include the equation $a/(b/c) = a(c/b)$, since both sides are $(ac)/b$.

The associativity and co-associativity rules from the lemma cover all but one expressions of the form $a \circ_1 (b \circ_2 c)$ and $(a \circ_1 b) \circ_2 c$ for $\circ_1, \circ_2 \in \{\times, /\}$. Each co-associativity matches pairs of these, sometimes swapping the order of two adjacent variables.

The missing expression from the co-associativity rules is $(a/b)c$. It can be shown that this expression is closely related to the wedge operator. In some cases, we can also relate it to the left divider, via the rule $(a/b)c = a(b\backslash c)$.

Diversion B.8.63. If / is only a mid-divider, not a post-divider, then the co-associativity can fail. For example, in a null semigroup, $ab = 0$ for all a, b and allowing any binary operation / is a mid-divider, because $((ab)/b)b = 0 = ab$. The co-associativity rules can fail, because / can be arbitrary.

Diversion B.8.64. Co-associativity can be viewed as arithmetic rules for somewhat simplifying complicated expressions involving multiplication and division. For example simplifications, upon encountering two adjacent divisions, one division can be replaced by a multiplication, thereby reducing the number of divisions in the expression. Similarly, factors in the numerator of division can be pulled out, leaving only the rightmost factor.

Diversion B.8.65. A **co-associative** divider $/$ is a binary operation on a semigroup such that

$$ab = ((ab)/b)b, \tag{B.8.32}$$

$$a/(bc) = (a/c)/b, \tag{B.8.33}$$

$$a(b/c) = (ab)/c = a/(c/b), \tag{B.8.34}$$

for all $a, b, c \in S$. Lemma B.8.25 can then be re-stated to say: if $/$ is both a pre-divider and post-divider, then it is co-associative. (The converse is likely false.)

If a co-associative divider exists, then the semigroup can be said to be **co-associatively divisible**.

Diversion B.8.66. Non-negative (or positive) integers under multiplication is not a co-associatively divisible semigroup. Let $a, b > 1$. Then $ab = ((ab)/b)b = (((ab)1)/b)b = ab(1/b)b$. This implies $(1/b)b = 1$, which would be possible over the rationals, but not in the non-negative (or positive) integers.

Diversion B.8.67. A commonly defined operation on non-negative integers, generally known as the **quotient**, defines d/b for $b > 0$, to be the smallest integer q such that $(1 + q)b > d$. Extend this operator to $b = 0$, by defining $d/0 = 0$. (Within positive integers, a different operator $/$, defined as the least integer such that $(d/b)b \geq d$, is a divider, though this operator is not used widely.)

Then $/$ is a mid-divider and it is partially co-associative, in that it obeys the rule $a/(bc) = (a/c)/b$. (As noted previously, no divider for integers can be co-associative. Moreover, the standard quotient $/$ does not obey any of three equalities $a(b/c) = (ab)/c = a/(c/b) = a(b/c)$.)

This suggests that the equality $a/(bc) = (a/c)/b$ is easier to meet, or at least special, or more natural, than full co-associativity. Inspecting the equality more carefully reveals that the functions $/_b$ for $b \in S$ are closed under composition, and form a semigroup, say $/_S$, and that the map $S \to /_S : b \mapsto /_b$ is a semigroup morphism. This property of $/$ may thus deserve a name, perhaps **closed**. A semigroup with a closed divider is **divisibly closed**. So the non-negative (or positive) integers are divisibly closed.

B.8.18 Wide social dividers?

This subsection looks at recovering multiplication from division.

Suppose that S is a set and $/ : S^2 \to S$ is a binary operator, which is initial assumed only to be a **notational** divider, meaning merely that it uses the notation $/$. (At this point, we have not said that S is a multiplicative semigroup, so an actual divider would be undefined.)

For each $b \in S$, let $1_b = b/b$ and let $b' = 1/b$. Define a multiplication operator on S by letting $ab = a/b'$. We call this multiplication the **denominal**

multiplication derived from /. (We also call $1_b = b/b$ the **denominal identity** of b, and b' the **denominal inverse** of b.) This makes S into a magma (not yet a semigroup, because multiplication might not be associative).

Operator / is a **wide divider** if / is a divider for the denominal multiplication derived from /. Equivalently, expanding all the definitions, an operator / is a **wide divider** on S if and only if

$$((a/((b/b)/b))/b)/((b/b)/b) = a/((b/b)/b) \tag{B.8.35}$$

for $a, b \in S$.

If the denominal multiplication derived from / is an associative operation, then we say that / is a **social divider**.

Diversion B.8.68. If multiplicative semigroup S has a peremptory divider, meaning only one divider, then / is a post-divider and pre-divider. Multiplication in S is necessarily denominal, with $ab = a/((b/b)/b)$ for all $a, b \in S$. In other words, there exists a unique multiplication operation $\times$ such that / is peremptory divider for $\times$.

Other multiplication operations such as null multiplication ($ab = 0$ for all a, b) may have / as a divider, but it will not be peremptory for the semigroup.

B.9 Wedge operators

To compute a wedge in a semigroup means the following definition.

Definition B.9.1. *A **wedge** operator is a function*

$$\wedge : S^3 \to S : [d, b, e] \mapsto d \wedge_b e \tag{B.9.1}$$

such that

$$(ab) \wedge_b (bc) = abc \tag{B.9.2}$$

for all $a, b, c \in S$.

Diversion B.9.1. The cryptographic significance of computing a wedge is given in §3.1.4: computing a wedge in semigroup amounts to a watcher breaking the seclusion security of key agreement scheme associated with the semigroup.

Diversion B.9.2. In the semigroup of even positive integers under multiplication,

$$48 \wedge_8 80 = 480,$$

for example, where $[a, b, c] = [6, 8, 10]$ in the notation above.

Diversion B.9.3. Some cryptography implementers might appreciate that Montgomery's modular multiplication a wedge operator.

B.9.1 Notational justification

The name and notation for the wedge operator is not at all standard, so warrants some justification.

- The subscript b to the symbol $\wedge$ helps distinguish from other meanings of the symbol $\wedge$.
- The wedge operator has close connections to division, and the symbol $\wedge$ suggest a consolidation of the right / and left \ divider symbols.
- The term to **drive a wedge** often means to thwart what would otherwise be agreement.

B.9.2 Terminology and uniqueness

To be revised.

Diversion B.9.4. For discussion purposes, name the variables involved as follows.

- The **problem** is to compute abc from $[ab, b, bc]$.
- The **input** to the wedge problem or operator is the array $[d, b, e]$.
  - The **left** input is d.
  - The **middle** input is b.
  - The **right** input is e.
- The **seed** is $[a, b, c]$.
- An **instance** of the wedge problem is the input $[d, b, e] = [ab, b, bc]$ derived from a seed $[a, b, c]$.
- A **valid** input has the form $[d, b, e] = [ab, b, bc]$ for some (seed) $[a, b, c]$.
- An **invalid** input has form $[d, b, e] \neq [ab, b, bc]$ for all seeds $[a, b, c]$.
- An **output** for a given input $[d, b, e]$ to the wedge operator is $d \wedge_b e$.
- A **relevant** output is an output from a valid input.
- A **irrelevant** output is an output from an invalid input.
- A **correct** output $d \wedge_b e$ is a relevant output for an instance $[d, b, e]$ from a seed $[a, b, c]$ such that $d \wedge_b e = abc$.
- A **solution** is a correct output.

Diversion B.9.5. An instance of $[ab, b, bc]$ of the wedge problem may have multiple different seeds (if the semigroup does not have the cancellation property).

Lemma B.9.1. *Each instance $[ab, b, bc]$ of the wedge problem has a unique solution abc.*

Proof. For each seed $[a, b, c]$, there is a unique solution abc, so the only case to handle is if two different seeds, say $[a, b, c]$ and $[a', b', c']$ lead to the same instance of the wedge problem.

Two such seeds have $[ab, b, bc] = [a'b', b', b'c']$. But

$$abc = (a'b')c = a'(b'c) = a'(b')c = a'(b)c = a'(bc) = a'(b'c') = a'b'c', \quad \text{(B.9.3)}$$

using associativity, and equations $ab = a'b'$, $b = b'$ and $bc = b'c'$. □

Diversion B.9.6. The wedge problem can be defined for any magma (set with a binary operation), not just semigroups. The lemma above uses associativity, so does not establish uniqueness of solutions in non-semigroups.

Because of the unique solution of the wedge problem, we can usually safely speak of **the wedge** operator, tacitly ignoring irrelevant outputs.

B.9.3 Incomplete wedges

To be revised.

Definition B.9.2. *A* ***wedge operator*** *is a ternary (three input) operator on a semigroup S, written*

$$\wedge : S^3 \to S : [d, b, e] \mapsto d \wedge_b e. \quad \text{(B.9.4)}$$

A wedge operator is ***correct*** *for seed-set $T \subset S^3$ if*

$$ab \wedge_b bc = abc \quad \text{(B.9.5)}$$

for all seeds $[a, b, c] \in T$.

So, the wedge operator provides the solution $ab \wedge_b bc$ to the wedge problem $[ab, b, bc]$ instance.

Diversion B.9.7. We also say that $\wedge$ is a wedge operator **relative** to seed-set T to mean that it is correct for set T.

B.9.4 Binary wedge

An important special type of incomplete wedge operator involves a fixed value of b.

In the context of associative key agreement, the middle input to the wedge problem is often a fixed value b. In this case, we can view $\wedge_b$ as a binary operation:

$$\wedge_b : S^2 \to S : (d, e) \mapsto d \wedge_b e. \tag{B.9.6}$$

B.9.5 Wedge algorithms

To be revised.

Definition B.9.3. *A **wedge problem solver** (for seed-set $T \subset S^3$) is an algorithm to implement a wedge operator (correct for T).*

B.9.6 Probabilistic wedges

To be revised.

Diversion B.9.8. We may wish to consider probabilistic wedge problem solvers. These algorithms receive inputs drawn from a probability distribution on the set T. The algorithm itself may make further probabilistic choices (formally, via random tape or other source of randomness). The resulting output of the wedge problem solver may be viewed as a random variable. We then the speak of the **success rate** as the probability of the equation (B.9.5).

Strategies to compute wedges are discussed in Chapter 4.

B.10 Aside: Polarizable functions?

To be completed!

This section defines a generalization of polynomial functions.

Polynomial functions are of basic importance in algebra. The hope in generalizing polynomials is that the basic importance generalizes usefully to wider settings.

Polynomials are defined by using both addition and multiplication. But polynomials also obey some functional equations that only involve addition (no multiplication or even subtraction). These functional equations in some cases are strong enough to characterize polynomial functions.

This motivates the generalization: **polarizable** functions are those that obey the same additive functional equations. These generalizations can be defined for functions between commutative semigroups. We use additive notation for the commutative semigroups, to draw out the similarity to polynomials functions.

Diversion B.10.1. The term **polarizable** is chosen here for its similarity to the terms **polynomial** and **polarization**, some related notions.

There may already exist different terminology for polarizable functions: the notion behind it is well-known.

The term **polarizable** is not related to the notion of polarized semigroups.

B.10.1 Free commutative semigroups

For any set X, let $[X]^+$ indicate the commutative additive semigroup freely generated by elements of the form $[x]$ with $x \in X$, which are formal copies of the elements of X. Let $[X]_0^+$ be the same, with an identity element 0.

For $s \in [X]^+$, we write $|s|$ (or $\#(x)$ or $\#x$) for the number of terms, so

$$|[x_1] + \cdots + [x_n]| = n, \tag{B.10.1}$$

and call this the **length** of s. The length function is actually a semigroup morphism from $[X]^+$ to $\mathbb{P}^+ = \{1, 2, 3, \dots\}^+$ (the positive integers under addition). For $0 \in [X]_0^+$, let $|0| = 0$.

For positive integer n, write

$$n[x] = \underbrace{[x] + [x] + \cdots + [x]}_{n \text{ copies of } [x]} = \sum_{i=1}^{n} [x] \tag{B.10.2}$$

as is usual in additive semigroups. In $[X]_0^+$, we also have we have $0[x] = 0$. In $[X]^+$, term of the form $0[x]$ can appears in a sum, unless all terms have that form. Terms for the form $0[x]$ are simply omitted from the sum. If all terms have this form, then the sum is not defined in $[X]^+$. Equivalently, consider everything to happen $[X]_0^+$, but then carefully track whether or not 0 results.

For $b \in \{0, 1\}$, define the b-**exploratory** function,

$$\varepsilon_b : [X]^+ \to [[X]_0^+]^+ : \left(\sum_{i=1}^{n} [x_i] \right) \mapsto \sum_{\substack{b_1, \dots, b_n \in \{0,1\} \\ \sum b_i \equiv b \bmod 2}} \left[\sum_{i=1}^{n} b_i [x_i] \right]. \tag{B.10.3}$$

Also, let $\varepsilon(s) = \varepsilon_0(s) + \varepsilon_1(s)$.

Diversion B.10.2. For example:

$$\varepsilon_0([2]+[2]+[3]) = [0]+[[2,2]]+[[2,3]]+[[2,3]],$$
$$\varepsilon_1([2]+[2]+[3]) = [[2]]+[[2]]+[[3]]+[[2]+[2]+[3]],$$

where the 0 in inside the [0] of the first term in the first equation is the additive identity of $[X]^+_0$.

Diversion B.10.3. Function ε_1 has image in the subsemigroup $[[X]^+]^+ \subsetneq [[X]^+_0]^+$, but $\varepsilon_0([x]) = [0]$, so zeros are necessary on the right.

Diversion B.10.4. Length grows exponentially under the exploratory function: $|\varepsilon_b(x)| = 2^{|x|-1}$.

Diversion B.10.5. The term **exploratory** is chosen to here for similarity to terms **exponential**, **parity**, and **polarization**.

For any function $f : X \to Y$, define an **induced** function $[f] : [X]^+ \to [Y]^+$ by:

$$[f]([x_1]+\cdots+[x_n]) = [f(x_1)]+\cdots+[f(x_n)]. \tag{B.10.4}$$

Diversion B.10.6. If clear from context, one can write $[f]$ as f. Clarification from context is sometimes needed. For example, in $[[X]^+]^+$, care is needed in how many layers of brackets to pass through.

B.10.2 Summation maps

Now let $X = S^+$, a commutative additive semigroup. Define the **summation** function $\sigma : [S^+]^+ \to S^+$ by

$$\sigma([a_1]+\cdots+[a_n]) = a_1+a_2+\cdots+a_n. \tag{B.10.5}$$

The summation function σ is a semigroup morphism. Extend the domain of σ to the $[S^+]^+$ be defining $\sigma(0) = 0$. If $0 \notin S^+$, then the extended σ is taken to have range S^+_0.

Diversion B.10.7. If the set S^+ also has a multiplication operation, for example, if S is a ring, then we can also define a function π similarly by $\pi([a_1]+\cdots+[a_n]) = a_1a_2\ldots a_n$.

For any $z \in S^+$, define the translation functions $t_z : S^+ \to S^+ : a \mapsto a+z$. Extend the domain of the each translation function to S^+_0 by setting $t_z(0) = z$.

B.10.3 Polarity functions

Again, let S^+ and T^+ be additive commutative semigroups.

Definition B.10.1. *If f is a function $f : S^+ \to T^+$, then define the* ***polarity function*** *of f as a function $\{0,1\} \times [S^+]^+ \times S^+ \to T^+$, written $[b, x, z] \mapsto f_{b,x}(z)$, and defined by*

$$f_{b,x}(z) = \sigma([f]([t_z]([\sigma](\varepsilon_b(x))))). \tag{B.10.6}$$

Equivalently,

$$f_{b,[x_1]+\cdots+[x_n]}(z) = \sum_{\substack{b_1,\ldots,b_n \in \{0,1\} \\ \sum b_i \equiv b \bmod 2}} f\left(z + \sum b_i x_i\right), \tag{B.10.7}$$

using the usual conventions that $1x = x$ and terms $0x$ vanish from the sum.

Diversion B.10.8. For example, $f_{1,[x]}(z) = f(x+z)$ and $f_{0,[x]}(z) = f(z)$, while:

$$\begin{aligned} f_{0,[x]+[y]}(z) &= f(z) + f(x+y+z), \\ f_{1,[x]+[y]}(z) &= f(x+z) + f(y+z). \end{aligned}$$

Diversion B.10.9. If $0 \in T$, then extend the polarity function to a larger domain $\{0,1\} \times [S^+]_0^+ \times S^+$ if $0 \in T$, by letting $f_{0,0}(z) = f(z)$ and $f_{1,0}(z) = 0$.

B.10.4 Recursions for polarity functions?

Polarity functions obey some recursions

$$f_{b,[0]+x}(z) = f_{0,x}(z) + f_{1,x}(z), \tag{B.10.8}$$

and

$$f_{b,x+y}(z) = (f_{0,x})_{b,y}(z) + (f_{1,x})_{1-b,y}(z), \tag{B.10.9}$$

and

$$f_{b,[u]+[v]+x}(z) + f_{b,[0]+x}(z) = f_{1-b,[u+v]+x}(z) + f_{b,[u]+x}(z) + f_{b,[v]+x}(z). \tag{B.10.10}$$

Recursions like this are useful for proving properties by induction.

B.10.5 Polarizable functions

Definition B.10.2. *Let d be an integer. A function $f : S^+ \to T^+$ is d-**polarizable** if*

$$f_{1,x}(z) = f_{0,x}(z) \tag{B.10.11}$$

*for all $x \in [S^+]^+$ with $|x| > d$ and for all $z \in S^+$. Say that d is a **degree** of polarizability of f.*

Diversion B.10.10. A constant function $f : S^+ \to T^+$ is 0-polarizable.

(For $f(s) = c$ and $|x| > 0$, we have $f_{b,x}(z) = 2^{|x|-1}c$.)

Conversely, a 0-polarizable function $f : S^+ \to T^+$ is constant.

(For any $x, z \in S^+$, we have $f(z) = f_0([x], z) = f_1([x], z) = f(x + z) = f(z + x) = f_1([z], x) = f_0([z], x) = f(x)$.)

Diversion B.10.11. The identity function $f : S^+ \to S^+ : z \mapsto z$ is 1-polarizable.

(For any $x \in [S^+]^+$ with $|x| > 1$, we have $f_{b,x}(z) = 2^{|x|-2}(2z + \sigma(x))$.)

Diversion B.10.12. If we extend the domains of the polarity function to $[S^+]_0^+ \times S^+$, which, recall, requires that $0 \in T$, then we get an extended notion for polarizable functions.

The only difference this extension makes for the set of d-polarizable functions is for integer $d < 0$, because then x with $|x| = 0$ becomes relevant. But $|x| = 0$, means $x = 0$. So, if $d < 0$ and f is d-polarizable in the extended sense, then $f(z) = 0$ for all $z \in S$. In the non-extended sense, d-polarizable for $d < 0$ means that same as 0-polarizable.

We will generally assume that $d \geq 0$, in which case, this distinction is not important.

Diversion B.10.13. If functions p and q are d-polarizable, then so is function $p + q$. Consequently, d-polarizable functions of degree d form a additive semigroup. (This is also true for $d = -1$.)

Diversion B.10.14. Let $S^+ = \{0, 1, 2, 3, \dots\}^+$. The function $f : S^+ \to S^+$ defined $f(n) = 2^n$ is not polarizable.

(For example, we have $f_{0,m[1]}(0) + f_{1,m[1]}(0) = \sum_{b_i} 2^{\sum b_i} = \sum_{k\geq 0} \binom{m}{k} 2^k = 3^m$, which is odd, so the two integer terms cannot be equal.)

Diversion B.10.15. If $0 \in S^+$ and there also exists an element $b \in S^+$ such that all nonempty sums of b are nonzero, so $b, b + b, b + b + b, \dots \neq 0$, then the function $f : S^+ \to S^+$ defined:

$$f(a) = \begin{cases} b & \text{if } a = 0 \\ 0 & \text{if } a \neq 0 \end{cases} \tag{B.10.12}$$

is not d-polarizable for any d, because $f_{1,n[b]}(0) = 0 \neq b = f_{0,n[b]}(0)$.

B.10.6 Differential polarity function?

Again, let S^+ and T^+ be a commutative additive semigroups. Fix a subtraction operator $-$ for T.

Definition B.10.3. *The* ***differential polarity function*** *of function* $f : S^+ \to T^+$ *is a function* $[S^+]^+ \times S^+ \to T^+$ *written* $[x, z] \mapsto f_x(z)$*,*

$$f_x(z) = f_{|x| \bmod 2,\ x}\ (z) - f_{|x|+1 \bmod 2,\ x}\ (z). \tag{B.10.13}$$

Diversion B.10.16. Let $\mathbb{R}$ be the ring of real numbers, and let $S^+ = \mathbb{R}^+$ be its additive semigroup. Let $R^\times$ be the subset of nonzero numbers, viewed as a multiplicative semigroup. Let $f : \mathbb{R} \to \mathbb{R}$ be a function that d times differentiable at 0. Then the d^{th} derivative of f at z can be defined using the differential polarity function as

$$f^{(d)}(z) = \lim_{x \to 0} \frac{f_x(z)}{\pi(x)},$$

where the limit runs over $x \in [R^\times]^+$ with $|x| = d$, and the limit $x \to 0$ refers to all terms approaching 0. (Recall $\pi(x)$ is the product of the entries in the multiset.)

B.11 Aside: productive semigroups?

Recall that in, a (multiplicative) semigroup S, multiplication is a binary operation having two inputs, and the result is called the **product**. Associativity implies any sequence of values can be multiplied together, which can be viewed as a multiple-input function, **product**.

Diversion B.11.1. For non-associative binary operations, multiple-input products can be defined, but the inputs must be arranged into a binary tree structure, not just a sequence structure.

In commutative semigroups, the multiple inputs to a product can be arranged into a multi-set.

Sometimes, semigroup multiplication can be generalized to allow for products of infinitely many elements, in a way formally defined below.

Diversion B.11.2. Such infinite-input products are not always uniquely determined by the semigroup multiplication, so must be considered an extra structure on the semigroup.

Given any multiplicative semigroup S, we form a semigroup $\mathrm{Fac}(S)$ of **(formal) factorizations**, whose elements are equivalence classes of functions $f : T \to S$, where T is a total order. Two functions $f : T \to S$ and $g : U \to S$ are considered equivalent if there exists an isomorphism $h : T \to U$

such that $f(t) = g(h(t))$ for all $t \in T$. We write $[f]$ for the equivalence class of a function.

Recall that two total orders, say T and U, can be combined to form new total order $T \triangleleft U$, consisting of disjoint copies of T and U, with all elements of T less than those of U.

Formal factorizations multiply as follows:

$$[f][g] = [h]$$
$$h : T \triangleleft U : v \mapsto h(v) = \begin{cases} f(v) & \text{if } v \in T, \\ g(v) & \text{if } v \in U. \end{cases}$$

A **solitary** factorization $[f]$ is a factorization $f : T \to S$ such that $|T| = 1$. A **finite** factorization $f : T \to S$ is such that T is finite. A factorization is **empty** if T if T is empty, and otherwise it is **non-empty**. A factorization is **binary** if $|T| = 2$.

A **general product** on the semigroup S is a semigroup morphism:

$$\prod : F \to S \tag{B.11.1}$$

where F is a subsemigroup of $\mathrm{Fac}(S)$ such that F contains all the solitary factorization, and for each solitary factorizations obeys:

$$\prod([f]) = f(t),$$

where t is the sole elements of T in $f : T \to S$.

The semigroup's original multiplication is directly recoverable from general products as defined over a binary factorization, since $\prod[0 \mapsto a, 1 \mapsto b] = ab$.

As per usual conventions, when applying the morphism $\prod$, we often write

$$\prod([f]) = \prod_{t \in T} f(t).$$

The subsemigroup F is called the **product scope** of the general product $\prod$. For set of all T (up to isomorphism), is call the **index type** of the general product. A **product(ion) semigroup** is a semigroup equipped together with a specific general product.

Every semigroup S has general product where the product scope consists of all non-empty, finite factorizations, which is the **minimum product**

scope, which is the non-empty, finite index type. Every product scope contains this minimum product scope. The value of $\prod$ on the minimum product scope is uniquely determined. (The empty factorization can be included in the product scope if and only if the semigroups has a 1.)

Diversion B.11.3. Some additional ideas that need to be corrected, or completed.

Product semigroups might form a category, with morphisms preserving the structure (such that not all semigroup morphism preserve the general product.)

General products can also be defined for formal factorizations too, but one particular general product is most natural. Consider a factorization of factorizations, which we can write as

$$[f] = [f : t \mapsto [f_t : T_t \to S]],$$

where $t \in T$ for some total order T, and $\{T_t\}_{t\in T}$ is a family of other total orders. The natural general product is then

$$\prod[f] = \left[\biguplus_{t\in T} T_t \to S : u \mapsto f_t(u)\right],$$

where $\biguplus_{t\in T} T_t$ is the ordered union of the family $T_{tt\in T}$ of total orders, and each u in the ordered is mapped to $f_t(u)$ where t is such that $u \in T_t$, for the copy of T_t inside the ordered union.

Consistency of a general product on the semigroup and the natural general product of the formal factorizations can be called **super-associativity**, which looks something like:

$$\prod\left(\prod_{t\in T}[f_t]\right) = \prod\left[t \mapsto \prod([f_t])\right].$$

On the left side of the equation, the inner product is the natural general product of factorization. The three product in the equation are the general product whose super-associativity is in question. Either:

- super-associativity can be proven to hold for any general product, or
- else super-associativity can fail for some general products, and thereby used to define an extra condition, and we can speak **super-associative** product semigroups.

We can also ask whether a super-associative general product meets a further closure property. If its index type I is closed under infinite ordered unions, meaning that $\biguplus_{t\in T} T_t \in I$ if $T, T_t \in I$ for all $t \in T$, then we say that the general product is **recursively super-associative**.

For an additive semigroup S^+, we re-name factorization as **series**, we re-name a products as a **sum**, or a **summation**, we write the morphism $\prod$ as $\sum$. From series to sum.

Examples of product semigroups:

- Let S be any well-ordered set and define multiplication on S to be the minimum. The a natural general product defines the product of any non-empty factorization $f : T \to S$ as the minimum element of the non-empty set $f(T)$.

- Let S be a multiplicative semigroup with a zero. Define a general product by setting the product of all infinite factorizations to be zero.

- Let S be a multiplicative semigroup with both a zero and a one. Define a general product the product of a empty factorization to be one, the product of a factorization $f : T \to S$ to be zero if $1_f = \{t : f(t) \neq 1\}$ is an infinite set, and otherwise $\prod[f] = \prod[f']$ where $f' : T_f \to S : t \mapsto f(t)$ is the restriction of f to the domain 1_f.

- Let $S^+ = (\mathbb{R}^+_+)_\infty$ be the semigroup of positive real numbers under addition, with the point extension by an absorbing element, written infinity ∞. We recall the natural ordering of the real numbers and place ∞ to right of all reals. Define a general summation whose scope is the smallest subsemigroup of series containing series of the form $[f : \mathbb{P} \to S]$, where $\mathbb{P}$ is the set of the positive integers. The infinite sum $\sum[f] = \sum_{i \in \mathbb{P}} f(i)$ is the least element S^+ greater than or equal to all partial sums, $\sum_{i \in \mathbb{P}, i \leq n} f(i)$. Any other infinite series is sum of $\mathbb{P}$-types series and finite series, so is defined accordingly.

  Diversion B.11.4. This general product is super-associative, but not recursively super-associative, because it its index type does not include $\mathbb{P} \triangleleft \mathbb{P} \triangleleft \mathbb{P} \triangleleft \dots$, also known as ω^2 to those familiar with ordinal numbers, whereas this total order must appear in recursively super-associative index type containing $\mathbb{P}$.

- Let $S^+ = (\mathbb{R}^+_{\geq 0})_\infty$. (A point extension of the same semigroup, with the addition of identity element 0). For any non-empty series $[f : T \to S]$, define the sum $\sum[f]$ as the least element greater than or equal to all sums of finite sub-series of f, meaning all $[g : U \to S : u \mapsto f(u)]$ where U is a finite subset of T. Let $\sum[] = 0$, since 0 is the least element greater than or equal to all members of the empty set.

Diversion B.11.5. It is natural ask about the relevance of infinite products to cryptography, despite this chapter aiming for the basic theory of semigroups. One possibility is that a product semigroup (or summation semigroup), can be used to build yet other infinite semigroups, which in turn might have useful subsemigroups or images. For example, there is little doubt that infinite series, such as the Weierstrass and Jacobi functions, were instrumental in the development of elliptic curve theory.

B.12 Semirings and realms

Multiplicative semigroups sometimes arise from rings, which are equipped with two binary operations. The extra operations can enable constructions of new semigroups (such as via matrices) new division algorithms (such as long division, which uses subtraction).

This section examines some definitions and facts generalizing the setting of ring theory.

Diversion B.12.1. A large part of semigroup theory is motivated by ring theory, because each ring has a multiplicative semigroup. So, despite the name semigroup theory is more a generalization of ring theory, than of group theory.

Diversion B.12.2. Ring theory is more widely taught than semigroup theory, perhaps because its greater applicability to natural problems. Ring theory is perhaps deeper, since the extra operations and axioms allows more sophisticated theorems.

Ring theory is such a rich area that it is usually divided into three large topics: field theory, commutative algebra, and non-commutative algebra, which can be studied somewhat separately.

It would be impossible and inappropriate to review all of ring theory in this section, so the discussion will be confined to reviewing some notations, and definitions, axioms, that generalize those of ring theory. For full ring theory, the reader should consult several standard textbooks, such as Lang's *Algebra* (or surveys like Hutchins [Hut81] for lists of specific examples).

B.12.1 Realms

Definition B.12.1. *A **realm** is a set with two binary operations.*

A realm's operations default to addition (written $+$) and multiplication (written silently), and the realm and its underlying are indicated as R.

Otherwise, notation of the form $R^{+,*}$ indicates a realm with set R, and binary operations $+$ and $*$.

Diversion B.12.3. A realm could also be called a **bi-magma**.

Diversion B.12.4. In some general theory of universal algebra, a realm is an algebra with signature $[2, 2]$.

Diversion B.12.5. In this report, the focus is on realms with addition and associative multiplication, meeting some further axioms, whereby the addition operation helps either to provide new secure semigroups, or to devise new attacks.

In a few cases, the focus will be on the additive semigroup.

If R is a realm, with addition and multiplication as binary operators, then R^+ can indicate its additive magma and $R^\times$ or R can indicated its magma, if clear from context. (If the relevant operation is associative, then the relevant magma is a semigroup, hence the interest in considering these in this report.)

B.12.2 Subsets of realms

The definition of realm, though abstract, enables the identification of notable subsets, generalizing some of notable subsets from ring theory (include the prime spectrum topology of rings). To avoid confusion with existing terminology, the generalized definition below uses a distinct geographic jargon.

Definition B.12.2. *A **subrealm** is a subset closed under both operations.*

*A **submagma** is a subset closed under one of the operations.*

*A **country** is a subrealm whose complement is a subrealm.*

*A **province** is subrealm whose complement is a submagma.*

*A **region** is an intersection of provinces.*

*A **bastion** is a union of finitely many provinces.*

*A **cloister** is an intersection of bastions.*

*A **territory** is a submagma whose complement is a submagma under the other the operation.*

*An **idyl** is a subrealm whose combination with its complement under one operation remains confined to the subrealm.*

Diversion B.12.6. For example, positive elements in an ordered ring also form a province. An ideal of a ring is an idyl. A prime ideal of a ring is both a province and idyl.

Assuming the default operations of addition and multiplication, the sets of above can be further qualified with the following definitions.

Definition B.12.3. *A **additive** set is closed under addition.*

*A **multiplicative** set is closed under multiplication.*

*Two **equi-operative** sets are closed under the same operation.*

*A **co-additive** set has an additive complement.*

*A **co-multiplicative** set has an multiplicative complement.*

*Two **co-operative** sets whose complements are equi-operative.*

Diversion B.12.7. For example, a set is a prime ideal of a ring if and only if it is both co-multiplicative provinces and an idyl.

Finally, using these subsets suggests ideas similar to the dimension theory of rings.

Definition B.12.4. *A* ***stratification*** *is a maximal chain of co-operative provinces.*

The ***distension*** *of a realm is one less than the maximum cardinality of a stratification.*

Diversion B.12.8. A **chain** above has the usual meaning, a set totally ordered under set inclusion.

The next jargon is occasionally useful:

Definition B.12.5. *An* ***element*** *of a realm is any element of its set.*

A ***member*** *of a realm is any element or operator of a realm.*

B.12.3 Function between realms

The most general function of interest between realms is the following.

Definition B.12.6. *A* ***chart*** *from a first realm to a second realm is a function from the members of the first realm to members of second realm, sending elements to elements, and operators to operators.*

Charts can be nearly arbitrary, but the following type of charts define the morphisms of a category.

Definition B.12.7. *A* ***map*** *is a chart that preserves the action of the operators.*

For example, if $m : R \to S^{[\oplus,\otimes]}$ is a map, then

$$\begin{aligned} m(a+b) &= m(a)m(+)m(b), \\ m(ab) &= m(a)m(\times)m(b). \end{aligned}$$

The equations are between before chart application and after chart application, for all inputs the to the operators.

Diversion B.12.9. Consider the realm $\mathbb{Z}^{\sqcup,\sqcap}$, integers with minimum and maximum operators. The map $m : n \mapsto -n$ and $m(\sqcup) = \sqcap$ and $m(\sqcap) = \sqcup$ is an isomorphism of the realm (taking the category-theoretic meaning of isomorphism).

B.12.3.1 Conservative realms, congruences, and valuators

Definition B.12.8. *An* ***relative*** *realm is a realm equipped with a binary relation (on its elements)*

The more interesting relative realms are the following:

Definition B.12.9. *A* ***conservative*** *realm where the operators conserve the binary relation.*

The most important case of a conservative realm is when the relation is an equivalence relation.

Definition B.12.10. *A* ***congruence*** *on a realm is an equivalence relation on the elements of realm that makes it into a conservative realm.*

Diversion B.12.10. Any map $m : R \to S$ of realms defines a congruence $\sim$ on S by the rule $a \sim b$ if and only if $m(a) = m(b)$.

The congruence equivalence classes also from a realm, the congruence realm, as discussed later in the next Appendix.

Definition B.12.11. *An* ***ordered*** *realm is a relative realm where the relation is a partial ordering.*

Assume by default that the binary relation is written as $\leq$.

So, in a conservative ordered realm (with default notations), whenever, $a \leq b$ and $c \leq d$, it will also be true that:

$$a + b \leq c + d,$$
$$ab \leq cd.$$

Non-negative real numbers form a conservative ordered realm.

For relative realms, another class of chart has some significance, with equations in the map definition, replaced by order relations.

Definition B.12.12. *A* ***valuator*** *is a chart from a realm to a relative realm, where each operator obeys the relations (either before chart application, or after chart application).*

A map is a valuator when the relation is the equality relation.

One valuator from the realm of complex numbers to the ordered realm of real numbers is the standard absolute value function (magnitude) $z \mapsto |z|$

on elements (and identity on operators), because

$$\begin{aligned}|a+b| &\leq |a|+|b|\\ |a||b| &\leq |ab|.\end{aligned}$$

The latter relation of the example could also be written $|ab| \leq |a||b|$.

B.12.3.2 Types of realm elements

To be corrected and revised.

In commutative algebra, some elements of rings can be classified as integral or transcendental (or negative-like). In the more lenient category of realms, the following weakenings of these properties can be given.

Definition B.12.13. *An **additively** (or **multiplicatively**) **internal** element a has subrealm $\langle a\rangle$ finitely generated as an additive (or multiplicative) submagma.*

*A **transcategorical** element a, relative to a subcategory C of realms, is such that for any realm B in C, and any $b \in B$, there is a map $\langle a\rangle \to B : a \mapsto b$.*

*A **negligible** element a has $a \in \langle\langle a\rangle - \{a\}\rangle$.*

Diversion B.12.11. The set of internal rational numbers is the set of integers. The set of negligible rational numbers is $\{r : r \in \mathbb{Q}, r < 0\} \cup \{\frac{1}{n} : n \in \mathbb{Z}, n > 1\}$. The set of non-negligible internal rational numbers is the set of non-negative integers $\{0, 1, 2, 3, \dots\}$.

Diversion B.12.12. The number $\sqrt{2}$ is non-negligible and internal. The number $1/\sqrt{2}$ is negligible and non-internal.

Diversion B.12.13. The additively internal elements of a semiring form a subsemiring.

Diversion B.12.14. If a subcategory C of realms contains at least one realm with at least one additively (or multiplicatively) non-internal elements, then no additively (or multiplicatively) internal element is transcategorical element relative to C.

Diversion B.12.15. The transcendental number π is transcategorical relative to the subcategory of semirings. Relative to larger categories with non-associative or non-distributive operations, π is not transcategorical.

B.12.4 Semirings

The special class of realms defined below, includes rings, and enables construction of semigroups using matrices.

Definition B.12.14. *A **semiring** is realm with medial addition, associative multiplication, and multiplication distributive over addition (both left and right).*

So, in semiring R, for a, b, c, d the following equations hold:

$$\begin{aligned}(a+b)+(c+d) &= (a+c)+(b+d),\\ a(bc) &= (ab)c,\\ a(b+c) &= (ab)+(ac),\\ (a+b)c &= (ac)+(bc).\end{aligned}$$

Of course, a ring is semiring:

Definition B.12.15. *A **ring** is a semiring R such that its additive magma R^+ is a commutative (abelian) group.*

Diversion B.12.16. The semiring condition of having medial addition is redundant, given the ring's condition that R^+ is a commutative group.

B.12.5 Nearrings

Various sources use the term **nearring** for our realms, where the addition is commutative, and multiplication is left-distributive, or right-distributive over addition.

Nearrings arise more easily than semirings, usually by way of endomorphisms, but less easily yield yet other semirings.

B.12.6 Why addition is often commutative

This section explores the notion that left and right distributivity are nearly enough to imply medial addition. To that end, we drop the condition of medial addition from a semigroup, as follows.

Definition B.12.16. *A **quasi-ring** is defined just like a semiring, except that addition need not be medial.*

Diversion B.12.17. As is shown later in this report, there exist quasi-rings with associative non-commutative addition, and therefore non-medial addition.

Therefore stronger conditions are needed to imply medial addition (let alone commutative addition).

Lemma B.12.1. *If R is a quasi-ring such that $R^\times$ is a group, and R^+ is a semigroup with a pre-subtracter $-$, then R is a semiring.*

Proof. Consider any $a, b, c, d \in R$. Choose any $z \in R$. Let

$$\begin{aligned} y &= d/z, \\ w &= b/z, \\ x &= y\backslash c, \\ e &= a - wx, \end{aligned}$$

using the pre-dividers / and \ available in any group. Then

$$\begin{aligned} a + b + c + d &= (a - wx) + wx + (b/z)z + y(y\backslash c) + (d/z)z \\ &= (a - wx) + wx + wz + yx + yz \\ &= (a - wx) + (w + y)(x + z) \\ &= (a - wx) + wx + yx + wz + yz \\ &= (a - wx) + wx + y(y\backslash c) + (b/z)z + (d/z)z \\ &= a + c + b + d. \end{aligned}$$

□

Diversion B.12.18. The result above can perhaps be strengthened by weakening the condition that $R^\times$ is a group, because the proof uses only three division d/z and b/z and $y\backslash c$.

It suffices that for one z (for each choice of a, b, c, d) to exist in S such that these three divisions are correct.

Lemma B.12.2. *If R is quasi-ring, with R^+ a semigroup having both left and right post-subtraction, and $R^\times$ has a left identity, then R is a semiring, and furthermore, R^+ is a commutative.*

Proof. Let $-$ be a right post-subtracter, $\boxminus$ a left post-subtracter, and 1 a left multiplicative identity. Let a, b be any elements of S. Compute as follows:

$$\begin{aligned}
b + a &= a \boxminus (a + (b + a)) \\
&= a \boxminus (((a + (b + a)) + b) - b) \\
&= a \boxminus ((((a + b) + a) + b) - b) \\
&= a \boxminus (((a + b) + (a + b)) - b) \\
&= a \boxminus (((1(a + b)) + (1(a + b))) - b) \\
&= a \boxminus (((1 + 1)(a + b)) - b) \\
&= a \boxminus ((((1 + 1)a) + ((1 + 1)b))) - b) \\
&= a \boxminus ((((1a) + (1a)) + ((1b) + (1b))) - b) \\
&= a \boxminus (((a + a) + (b + b)) - b) \\
&= a \boxminus ((((a + a) + b) + b) - b) \\
&= a \boxminus ((a + a) + b) \\
&= a \boxminus (a + (a + b)) \\
&= a + b.
\end{aligned}$$

So, R^+ is commutative. Commutative and associative implies medial. □

Appendix C

Semigroup examples: some sketches

This chapter sketches constructions of some semigroups, for the purpose of concretely illustrating examples of semigroups affected by the division and wedge algorithms described earlier in this report.

Each semigroup implied by this chapter is very likely at least one of:

- insecure, or
- impractical, or
- incorrect, or
- old and well-known.

The reader is **cautioned** is to consider each semigroup sketched here as insecure – until strong evidence suggests otherwise. No such evidence is implied by this report.

C.1 Semigroups with almost arbitrary multiplication

The four binary operations of any key agreement can be embedded into the multiplication of a semigroup, as shown §2.20.

Even totally arbitrary binary operations can be embedded into a semigroup. One approach for this is similar to that used in §2.20, which is fill in the rest of the semigroup multiplication table with zeros, resulting in a 3-nilpotent semigroup – which is well-known method, see Distler [Dis10], for example. Another approach is to fill up the semigroup freely with sequences, subject to the congruence implied by the given binary operation.

C.1.1 Quick review: Disjoint unions of sets

This section makes use of a formality called a **disjoint union of sets**, which we review:

Definition C.1.1. *A* ***disjoint union*** $A \uplus B$ *of given sets* A *and* B*, is any set which is a union of disjoint subsets that are copies of* A *and* B*.*

For a specific formal example, let $A \uplus B = \{(t,u) | t \in \{1,2\}; t = 1 \implies u \in A; t = 2 \implies u \in B\}$*.*

By convention, because of the existence $A \uplus B$, we can safely assume that A and B are disjoint, without worrying about the formally making the distinct copies.

We can form a disjoint union $A_1 \uplus \dots A_n$ of several sets, whose elements are formally (i, a_i) with $a_i \in A_i$. The disjoint union also allows us to add an arbitrary formal symbol.

C.1.2 Embedding in a 3-nilpotent semigroup

Definition C.1.2. *The* ***semigroup closure*** $S^{\langle m \rangle}$ *of binary operation* $m : L \times R \to M$ *consists of the set* $(L \cup R) \uplus M \uplus \{0\}$ *with a multiplication operation*

$$st = \begin{cases} m(s,t) & \text{if } (s,t) \in L \times R; \\ 0 & \text{if } (s,t) \notin L \times R. \end{cases} \tag{C.1.1}$$

Diversion C.1.1. We can also denote the binary operation of S of $\langle m \rangle$ if necessary to distinguish from other forms of multiplication.

Lemma C.1.1. *The semigroup closure* $S^{\langle m \rangle}$ *is a semigroup.*

Proof. It suffices to prove multiplication defined in (C.1.1) is associative.

We claim that $a(bc) = (ab)c = 0$ for all a, b, c, which will prove that $S^{\langle m \rangle}$ is a semigroup.

Consider $a(bc)$.

If $(b,c) \notin L \times R$, then $bc = 0 \notin R$. Therefore $(a, bc) \notin L \times R$, so $a(bc) = 0$.

If $(b, c) \in L \times R$, then $bc = m(b, c) \in M$. Then $bc \notin R$, since the bc belongs to the copy of M which is made to be disjoint from the copy of R. Therefore $(a, bc) \notin (L, R)$, so $a(bc) = 0$.

The argument for $(ab)c$ is similar. □

Diversion C.1.2. As a concrete example from cryptography, both the block cipher AES and message authentication code HMAC can be viewed as binary operators $L \times R \to M$: with the left domain L being keys (fixed-length bit strings), the right domain R being message (fixed-length for AES, or variable-length for HMAC, bit strings), and the range being ciphertexts or tags (fixed-length bit strings). Therefore, semigroup closures $S^{\langle \mathrm{AES} \rangle}$ and $S^{\langle \mathrm{HMAC} \rangle}$ can be formed.

Diversion C.1.3. If m in $S^{\langle m \rangle}$, has $M \subseteq L \cup R$, then $S^{langlem\rangle}$ has a disjoint copy M' of M is used, which has $M' \nsubseteq L \cup R$. Consequently, $S_{\langle m \rangle}$ can have associative multiplication even if m the multiplication of a non-associative magma.

Diversion C.1.4. If a semigroup S is some arbitrary semigroup, with binary operation $\times$, then S is not generally the same, or even isomorphic, is to the semigroup closure $S^{\langle \times \rangle}$.

Diversion C.1.5. Semigroups with the property that the product of any three elements is a constant (which may be re-labelled 0) are called 3-nilpotent.

In this sense, the semigroup closure $S^{\langle m \rangle}$ is formally degenerate, and not very interesting.

Diversion C.1.6. It can be shown that any 3-nilpotent semigroup can be constructed in the same manner as above.

The semigroup $S^{\langle m \rangle}$, being 3-nilpotent, has an easy wedge problem (§**??**.)

C.1.3 Embedding into a non-nilpotent semigroup

To be completed.

See §4.5.

C.2 Semigroups from hard homogeneous space

To be clarified.

Couveignes [Cou06] described an idea called a **hard homogeneous space** (HHS).

A HHS is a special case of a **permutation group**, where various operations must be implemented by an algorithms, and some operations (or problems) should be infeasible to implement by algorithms.

Couveignes shows how to define a key exchange scheme from any given hard homogeneous scheme. Under the report's terminology, the key exchange scheme of Couveignes can be easily transformed into an asynchronous key agreement scheme. (Precisely, it requires that the value h_0 is fixed.)

Consequently, a hard homogeneous space can be lead to an associative key agreement, since the previous observation in this report show that any asynchronous key agreement scheme is effectively associative.

For completeness, a direction reduction of a hard homogeneous space (G, H) to a semigroup S is now given, using the notation of Couveignes and this report.

Let $S = \{0\} \uplus H \uplus G$. As usual, let $0s = s0 = 0$ for all $s \in S$. Let $h_1h_2 = 0$ for all $h_i \in H$. For g_1, g_2, define the product g_1g_2 in S to be same as the product in G. Let $hg = gh = g.h$ for $g \in G$ and $h \in H$, where the dot in $g.h$ signifies the group action.

It is a good exercise to verify the associativity of S.

Diversion C.2.1. The semigroup S constructed above is commutative.

Couveignes requires efficient some algorithms for his Problems 1–5. Algorithms for these problems almost imply that S is what we call a **practical** semigroup. What might missing is monography, since Couveignes says "We assume that elements in G and H are represented by strings in a non necessarily unique way." This seems to be a slight mistake, since it implies that the agreement will fail if the two parties in the key exchange cannot arrive at the same string representation of the value K.

Couveignes defines his Problem 6, named "**Vectorization**", computing $g \in G$ such $g.h_1 = h_2$ for given h_1, h_2. The vectorization problem is an instance of post-division in S. Using our standard notation, put $b = h_1$ and $d = h_2$ and $g = d/b$. This should be expected, because Couveignes describes the vectorization problem as generalization the discrete logarithm problem, while division (in a semigroup) also generalizes the discrete logarithm problem.

Couveignes writes $\delta(h_2, h_1)$ for the solution of the vectorization problem. By the observation above, this amounts to $\delta(h_2, h_1) = h_2/h_1$ in the semigroup S, using post-division.

Diversion C.2.2. The notation $\delta(h_2, h_1)$ for the solution of the vectorization problem matches Hamilton's original definition of a vector as a difference (a delta) between two points in the space.

If we had used additive notation for our semigroup, then it would have been written $h_2 - h_1$.

Couveignes defines his Problem 7, named "**Parallelization**", which amounts to computing, with three inputs $h_1, h_2, h_3 \in H$ yielding a fourth value $h_4 \in H$. The success condition is expressed using δ, but translate to a division operation, it can be written $h_2/h_1 = h_4/h_3$. It can be shown that $h_4 = h_2 \wedge_{h_1} h_3$, so the parallelization is an instance of the wedge problem. Couveignes also remarks the parallelization can be solved by vectorization: which this report has observed as division is usable to solve the wedge problem.

Hard homogeneous spaces are a simple abstraction, much like semigroups. Couveignes not only identifies the abstract value of generalizing Diffie–Hellman key agreement, but gives concrete examples of potential hard homogeneous spaces, using elliptic curves and isogenies. Subsequent work generalized these examples.

It seems clear (to me) that semigroups are far more general than hard homogeneous spaces, but it is not yet clear (to me) whether semigroups outside those that can be constructed from hard homogeneous can also satisfy basic security.

Diversion C.2.3. I would argue that semigroups, with only one operation and only one axiom, are simpler abstraction than the abstraction of hard homogeneous spaces.

Semigroups also seem to more **widely** generalize Diffie–Hellman key agreement. The semigroup S constructed from a hard homogeneous space is commutative, but a general semigroup need be commutative to be used in associative key agreement. Indeed, associative key agreement covers all asynchronous key agreement schemes.

C.3 Semigroups from orderings

Though the term **semigroup** suggests a modest generalization of groups, semigroups generalize more diverse objects such as certain kinds of partial orderings (called semilattices).

C.3.1 Semilattices

A **semilattice** a set L with a partial order relation $\leq$ in which every pair of elements a, b has a infimum (greatest lower bound), say $a \sqcup b = \inf\{a, b\}$, meaning that $c \leq a, b$ if and only if $c \leq a \sqcup b$.

The infimum binary operation is associative, so a semilattice forms a semigroup. If no confusion arises, then we will write ab for $a \sqcup b$.

Diversion C.3.1. A semilattice is sometimes called **lower semilattice**, with an **upper semilattice** being a partially ordered set with unique pairwise supremums. Then a semilattice means either an upper or lower semilattice, usually defaulting to a lower semilattice.

Any upper semilattice can be converted to a lower semilattice by reversal of the order relation.

Diversion C.3.2. The notation $a \vee b$ is a more common notation than $a \sqcup b$ for infimum operation in a semilattice. The name **meet** is a more common name. The name **meet-lattice** is often used for lower semilattice.

Similarly, the term **join** is more common than supremum, and the notation $a \wedge b$ is more common than $a \sqcap b$.

Unfortunately, the more common notation for join conflicts with this report's notation for wedge. To minimize the confusion, the alternative notation $a \sqcap b$ is used in this report. For symmetry of notation, the notation $a \sqcup b$ is used as notation for the meet.

Diversion C.3.3. A lattice (order) is a partially ordered set that is both a lower semilattice and an upper semilattice, meaning pairwise infimums and supremums exist.

The semigroup of a semilattice is commutative and idempotent. (Consequently, division is easy.)

Conversely, any semigroup that is both commutative and idempotent defines a semilattice by the relation $a \leq b$ if and only if $a = ab$. This is a well-known theorem of (Clifford, Preston, Liapin, or earlier?) [**?**].

For any semigroup S, the subset of idempotent elements is often written $E(S)$. The set $E(S)$ of idempotents may be partially ordered by a relation $\leq$ defined by $e \leq f$ if and only $e = ef = fe$.

Diversion C.3.4. There is also notion called a **residuated semilattice**, in which some operators / and \ are considered part of the axiomatic structure. If these operators meet the definitions of right and left dividers used in this report, then at least some consistency would be achieved.

C.3.2 Orderly semigroups

In the notation of the previous section, a total ordering, also known as linear ordering, is a partial ordering with the extra property that $a \sqcup b \in \{a, b\}$ for all a, b. Consequently, a total ordering is a semilattice.

An **orderly** semigroup is the semigroup associated with total ordering, as described in the previous section. So,

$$ab = a \sqcup b = \min(a, b) \tag{C.3.1}$$

Again, we generally use notation $\sqcup$ only when needed to avoid confusion with other multiplication operations.

Diversion C.3.5. We make a distinction between orderly semigroup and **ordered** semigroup, further below.

An orderly semigroup has the property that $ab \in \{a, b\}$, which we call the **idoperant** property.

Any semigroup that is both commutative and idoperant is an orderly semigroup. As noted in the previous section, the ordering can be recovered from the semigroup multiplication by the rule:

$$(a \leq b) \iff (a = ab), \tag{C.3.2}$$

Suppose $a \leq b$ and $b \leq c$. Then $a = ab = abc = ac$, so $a \leq c$.

Diversion C.3.6. In an orderly semigroup, the minimum element, if any, acts like 0, while the maximum element, if any, acts like 1 (serving as the multiplicative identity).

Diversion C.3.7. An idoperant semigroup is obviously idempotent: $a^2 = aa \in \{a, a\} = \{a\}$, so $a^2 = a$ for all $a \in S$. Therefore, division is easy in any idoperant group.

Diversion C.3.8. For the semigroup $\{0, 1\}$ of non-negative idempotent integers under standard multiplication, the multiplication operation is identical the minimization operation.

Diversion C.3.9. For each non-negative integer n, there is only one semigroup isomorphism class of finite orderly semigroups of cardinality n. Perhaps the most natural labelling of elements of such a finite orderly semigroups is a subset $\{\frac{a}{n-1} | 0 \leq a \leq n-1\}$ of the rational numbers.

For infinite cardinalities, there are many orderly semigroup classes.

Diversion C.3.10. An **ordered** semigroup is traditionally considered a semigroup that has consistency with a linear (total) ordering. However, the linear ordering can be made into its own semigroup: an orderly semigroup.

In this case, we can require the notation $\sqcup$ to disambiguate confusion with the multiplication of the ordered semigroup. The consistency between the orderly semigroup and ordered semigroup can then be expressed using the distributive law:

$$ab \sqcup ac = a(b \sqcup c) \tag{C.3.3}$$

$$ac \sqcup bc = (a \sqcup b)c \tag{C.3.4}$$

Essentially, this is to say that we have a semiring, which is discussed much later in the report. In order words, a an ordered multiplicative semigroup is essentially a semiring whose additive semigroup is an orderly semigroup (equivalently, addition is commutative and idoperant).

Diversion C.3.11. In an ordered semigroup, there is a conflict in notation between the 0 and 1: the notation in the ordered semigroup might not match the notation in the associated orderly semigroup.

C.4 Small semigroups

This section discusses some of the smallest semigroups.

C.4.1 Empty semigroup

There is only one semigroup of cardinality 0, namely an empty semigroup. We could write this semigroup $\varnothing$, with the usual notation for its underlying set.

In many case, the notation will be clearer if write this semigroup as 0, provided context avoids any confusion with an element 0 within a semigroup. (This will be because, semigroups can be grouped combined with associative operations, in which case semigroup 0 usually acts just like 0 in a multiplicative semigroup.)

The empty semigroup has an important role in category theory as the **initial object**. For any semigroup S, there exists a unique morphism $m : 0 \to S$.

C.4.2 Unit semigroup

All semigroups of cardinality 1 are isomorphic – they share the same structure.

When dealing with isomorphism classes of semigroups, we simply write 1 for this semigroup class.

As a representative semigroup in the semigroup class 1, we can choose an underlying set $\{1\}$, with multiplication $1 \cdot 1 = 1$. This matches the usual convention of writing 1 for identity element of multiplication. To belabor the obvious, we can write out the multiplication table of the unit semigroup.

$$\begin{array}{c|c} \cdot & 1 \\ \hline 1 & 1 \end{array} \tag{C.4.1}$$

Diversion C.4.1. Usually, the context making clear whether 1 means a semigroup class, its semigroup representative, or some elements 1 of a semigroup.

The naive notation above would imply $1 = \{1\}$, which in turn, implies that $1 = \{\{1\}\} = \{\{\{\ldots\}\}\}$, so that 1 would not be well-founded as a set. So, strictly this is a notational convention.

The unit semigroup has an important role in category theory as the **terminal object**. For any semigroup S, there exists a unique morphism $m : S \to 1$.

C.4.3 Semigroups of size 2

See the Wikipedia page.

To be completed.

C.4.4 Semigroups of size 3

See the Wikipedia page.

To be completed.

C.4.5 Abundance of small semigroups

For general information on the numbers of finite semigroups, see:

http://oeis.org/wiki/Index_to_OEIS:_Section_Se#semigroups.

For example, there are 1,627,672 isomorphism classes (structures) of semigroups of size 7. By comparison, there is only 1 group isomorphism class of size 7.

Diversion C.4.2. If we ignore isomorphisms, instead just count (labeled) semigroup operations on the set $\{1, 2, 3, 4, 5, 6, 7\}$, there are $840 = 7!/6$ groups, and 7,743,056,064 semigroups, which is a nearly 10 million times as many. There are on average, about 4700 semigroups per isomorphism class (of size 7), which is a little less than the $7! = 5040$ possible re-labellings of each semigroup of size 7.

The abundance suggests that semigroups can have rather arbitrary structure. Perhaps most of these are 3-nilpotent, or 4-nilpotent. For large semigroups, such as semigroups of size near 2^{256}, there are very many possible semigroups (and semigroup structures).

C.5 Semigroups from sets

Given an arbitrary set X, we can form semigroups in various ways.

C.5.1 Zero semigroups

Let X be any non-empty set, and let $0 \in X$ (re-labeling some element of X as 0). Define a semigroup S whose set is X, with multiplication $xy = 0$ for all $x, y \in X$. Multiplication is associative because $(st)u = 0 = s(tu)$.

This is called the **zero** semigroup on set X. We can write this as 0_X.

C.5.2 Left and right semigroups

Let X be any set. The **left semigroup** on X is the semigroup $S = L(X)$ whose underlying set is X and whose multiplication is defined by $st = s$ for all $s, t \in X$.

In words, the multiplication equals the left input.

The converse semigroup is the right semigroup. We focus on the left semigroup, with the discussion of the right semigroup being the mirror image.

The left semigroup is associative, because $(st)u = s = s(tu)$.

C.5.3 Boolean semigroups

The **Boolean** semigroup on a set X is the set of $B(X)$ of all subsets of X with multiplication defined as set-intersection.

Equivalently, the Boolean semigroup $B(X)$ can be defined as the set of boolean-valued functions on X. Point-wise multiplication is used. For example, $(0110)(0011) = (0010)$.

Boolean semigroups are idempotent semigroups. An element b is idempotent if $b^2 = b$. A semigroup is idempotent if all its elements are idempotent.

C.5.4 Semigroups of words

Given a non-empty set X, the **word semigroup** on X is the the set $W(X)$ of non-empty sequence $[x_1, \dots, x_n]$, with semigroup multiplication definition by simple concatenation of sequences:

$$[x_1, \dots, x_m][y_1, \dots, y_n] = [x_1, \dots, x_m, y_1, \dots, y_n]. \tag{C.5.1}$$

Elements of X are called **symbols** and elements of $W(X)$ are called **words**.

Diversion C.5.1. We also call $W(X)$ the **free semigroup** on set X.

If $|X| > 1$, then $W(X)$ is a non-commutative semigroup.

Let $W(X)_1$ be the extension of $W(X)$ which allows the empty word $[]$, so that $W(X)_1$ becomes a monoid with identity $[]$. If clear from context, we write $[]$ as 1. We call $W(X)_1$ the **word monoid** or **free monoid** of X.

If $|X| = 1$, then $W(X)$ is isomorphic to $\mathbb{P}^+$ (positive integers under addition). If $|X| = 0$, then $W(X)$ is the empty semigroup 0.

Diversion C.5.2. When the symbols in X are easily distinguishable from variables names, then the notation for words can be simplified by omitting the square brackets and commas. For example, if $X = \{0, 1\}$, then the symbols are numbers, which are not confused with letters using variables names. In this case, can we identity $0 = [0]$ and $1 = [1]$. Then $[0, 1, 1] = [0][1][1] = 011$.

Diversion C.5.3. The word monoid $W(X)_1$ is sometimes written using the Kleene star notation X^*.

Diversion C.5.4. Every semigroup S is an image of a word semigroup, namely $W(S)$, with the morphism:

$$[s_1, \dots, s_n] \mapsto s_1 \dots s_n. \tag{C.5.2}$$

In other words, word semigroups have a universal property: there is a surjective morphism from some word semigroup to any given semigroup.

C.5.5 Semigroup of functions

This section elaborates on §C.8.7.1.

Let $S = X^X$ be the set of functions from X to itself. Then S a semigroup under function composition, which means that, if $f, g \in S$, then fg is the function defined by

$$(fg)(x) = f(g(x)) \tag{C.5.3}$$

for all $x \in X$.

We call this function (or transformation) semigroup of X.

Note that X^X may have subsemigroups.

Indeed, every semigroup S is isomorphic to a subsemigroup of a semigroup X^X, where $X = \{1\} \cup S$. Note X is a semigroup consisting of S and an extra element 1, disjoint from S such that $1x = x1 = 1$ for all $s \in X$. Map $s \in S$ to the function $l_s \in X^X$ defined by $l_s(x) = sx$. Then $l_s l_t = l_{st}$, and $l_s = l_t$ if and only if $s = t$.

In other words, function (transformation) semigroups have a universal property: there is injective morphism from any given semigroup into some function semigroup.

In fact, we already saw a similar embedding of S into S^S, which was used to translate between mid-dividers in S and semigroup inverses in S^S. The embedding from S and S^S might be injective if S is not a monoid.

If X is finite, there seem be two possible ways to representing elements in X^X.

- If X is finite, then the listing representation uses a fixed enumeration $(x_1, \ldots, x_n)$ and the representation of f is just $(f(x_1), \ldots, f(x_n))$, or alternatively $((x_1, f(x_1), \ldots, (x_n, f(x_n)))$.

- If elements of X have finite representations, then f could be represented by some program, the programmatic representation. If X is infinite, then only represents a proper subset of X can be so represented (because X^X is uncountable, and the set of programs if countable.

Note that the listing representation can be viewed a special case of the programmatic representation, with the program consisting of looking up $f(x_i)$ in the list.

Note that programmatic representation of functions is non-unique. For the purposes of key agreement, a unique representation is needed. Generally, listing representations are unique if X (once X sorted).

If X is a finite field of size q, then unique representations via polynomials of a degree at most $q-1$, can be used as unique programmatic representations.

Multiplying two functions f and g given in listing representation is quite fast if X is sortable, and could take about $|X| \log(|X|)$ steps. Multiplying functions in programmatic form is nearly free, since it can be define as the concatenation of the programs. However putting these representations into a unique form may require X steps.

If we describe programs with a set of symbols Y, then the description above may be viewed as an image of a subsemigroups of the word semigroup $W(Y)$.

Every $f \in X^X$ has a semigroup inverse g (which, recall, means that $fgf = f$). A semigroup inverse g may be found in $2|X|$ as follows. Run through $x_i \in X$ in order, from $x_1, \ldots, x_n$. Compute $y_i = f(x_i)$. If $g(y_i)$ has not yet been defined previously, then define $g(y_i) = x_i$. After reaching x_n, then run a second loop through x_i, as follows. If $g(x_i)$ has not been defined previously, then define to be some arbitrary element of X.

Recall that a semigroup inverse implies a solution to the wedge problem and the (middle) division problem. So, for X^X to be secure, we need that $|X|$ to be large enough to make the algorithm above infeasible.

Unfortunately, making $|X|$ infeasibly large generally makes the key agreement infeasible for the users, because they must also take $|X|$ steps to multiply (into a unique representation form).

If Alice and Charlie use a programmatic representations, with a maximum program length, then they are really using only a subset of X^X. In this case, they would still want X to be large.

C.5.6 Semigroups of relations

Let X be a set. Let $\mathrm{Rel}(X)$ be the set of relations on X. A relation r is merely a subset of $X \times X$. We define composition of $r, s \in \mathrm{Rel}(X)$ as follows:

$$rs = \{[x,z] : [x,y] \in r, [y,z] \in s, \text{ for some } y \in X\}. \tag{C.5.4}$$

Relation composition is associative.

The **transpose** of $r \in \mathrm{Rel}(X)$ is defined as $r^t = \{[y,x] : [x,y] \in r\}$. The **complement** of $r \in \mathrm{Rel}(X)$ is defined as $r^c = X^2 - r = \{[x,y] : x, y \in X, [x,y] \notin r\}$.

Semigroup $\mathrm{Rel}(X)$ has a subsemigroup anti-isomorphic to X^X. Suppose $f \in X^X$, let r_f be its graph:

$$r_f = \{[x, f(x)] : x \in X\}. \tag{C.5.5}$$

Then $r_{fg} = r_g r_f$, and $r_f = r_g$ if and only $f = g$. Transposing each function graph, as $l_f = r_f^t = \{[f(x), x] : x \in X\}$, gives a subsemigroup of $\mathrm{Rel}(X)$ isomorphic to X^X.

Diversion C.5.5. Any directed graph with vertex set X defines a relation on X.

Diversion C.5.6. The semigroup of relations can also form a semiring, with addition given by taking the union, as described in §C.15.2. In fact, the semiring of relations better describes the structure. The semiring of relations can also be described as a semiring of matrices with boolean entries.

C.6 Semigroups: smaller from larger

Many interesting semigroups are most easily defined by referring to some larger semigroup. This section briefly describes constructions.

C.6.1 Subsemigroups

See §B.4 for definitions of subsemigroups. Essentially, a subsemigroup T of S is a subset $T \subseteq S$ closed under the binary operation of S. Write $T \leq S$ to indicate that T is a subsemigroup of S (except when the notation $\leq$ applied to subsets of T has some other, more important meaning).

C.6.1.1 Green's subsemigroup

Green's equivalence relations can be defined for any semigroup S. There are various relations, including **Green's H-relation**, defined as follows:

$$a \sim_H b \iff (a \cup Sa = b \cup Sb)\&(a \cup aS = b \cup bS) \tag{C.6.1}$$

Green's theorem states each equivalence class H for the relation $\sim_H$ satisfies one of:

$$\begin{aligned} H^2 \cap H &= \varnothing \\ H^2 &= H \end{aligned}$$

In the latter case, H is a subsemigroup of S. Furthermore, H is actually a group (to be verified).

C.6.1.2 Subsemigroup of invertible elements in a medial semigroup

In a medial semigroup S (where, recall, $abcd = acbd$ for all $a, b, c, d \in S$), the middle invertible elements are closed under multiplication, so they form a subsemigroup.

C.6.1.3 Subsemigroup of cancellative elements

Recall that b is **right-cancellative** if $(ab)/b = a$ for all a (for any divider $/$), or equivalently, $ab = eb$ implies that $a = e$, for all a, e.

Right-cancellative elements form a subsemigroup. To see this, suppose that b, d are right-cancellative and $abd = ebd$. Then $ab = eb$ because d is right-cancellative, so $a = e$, because b is cancellative – hence bd is cancellative. (To see this from the divider definition, compute $(abd)/(bd) = (((abd)/(bd))b)/b = ((((abd)/(bd))bd)/d)/b = ((abd)/d)/b = (ab)/b = a$.)

C.6.1.4 Subsemigroup of fixed elements

See §B.6.3 for the definition of the subsemigroups $S_{\subset e}$ and $S_{\supset e}$ which can be defined for each $e \in S$.

C.6.1.5 Subsemigroup of multiples

The **set of multiples** of b is the set $Sb = \{ab : a \in S\}$. Equivalently, $Sb = \{d : b|d\}$, using divisibility to characterize being a multiple. The set Sb of multiples of b is a subsemigroup of S (so $Sb \leq S$). To see this, note that $(ab)(eb) = (abe) \in Sb$.

Diversion C.6.1. The set Sb is also a **right ideal** of S, not just a subsemigroup, because $S(Sb) \subseteq Sb$.

C.6.1.6 Subsemigroups of non-divisors

For $d \in S$, define the **set of non-divisors** of d to be set $N_d = \{b : b \nmid d\}$. Equivalently, $N_d = \{b : ab \neq d, \forall a \in S\} = \{b : d \notin Sb\}$. The set of non-divisors is N_d is a subsemigroup. To see this, suppose that $b \in N_d$. Then $alb \neq d$ for all a, l. Therefore, $lb \in N_d$.

Diversion C.6.2. The set N_d is also a **right ideal** of S, not just a subsemigroup, because $S(N_d) \subseteq N_d$.

C.6.1.7 Subsemigroup of non-nullifiables?

If $0 \in S$, and $s \in S$, then we say that s is **zero-divisor** if there exists a $t \in S$ with $0 \neq t$ and $0 \in \{st, ts\}$. If s is not a zero divisor, then we say that s is **non-nullifiable**. Let N be the set of non-nullifiable elements.

Diversion C.6.3. If $0 \in S$ and $|S| > 1$, then the element 0 is a zero divisor, so $0 \notin S$. If $|S| = 1$, then $0 \in S$ but element 0 is not zero divisor (because there are no nonzero elements to multiply against), so 0 is non-nullifiable and $N = S$.

Lemma C.6.1. *Then N is a subsemigroup of S*

Proof. For $s, t \in N$, it is necessary to prove that $st \in N$.

Suppose otherwise, so that $st \notin N$. Suppose that $u \in S$ and $u \neq 0$. Then $tu \neq 0$ because $t \in N$. Therefore, $s(tu) \neq 0$, because $s \in N$ and $(tu) \neq 0$. Therefore $(st)u \neq 0$. Similarly, $u(st) \neq 0$. This holds for all $u \neq 0$, so $st \in N$. □

If $N = S \backslash \{0\}$, then we might call S an **non-nullifiable** semigroup. In this case $S_{(0)} = 1 \triangleleft N$.

C.6.1.8 Nominalization of a subset?

Let S be a semigroup, and let U be a subset. A subsemigroup can be defined as:

$$S_U = \{s \in S : sU \subseteq Us\}, \tag{C.6.2}$$

which we call the **left nominalization** of U. (Right nominalization can be defined similarly.)

To set that this is a subsemigroup, suppose $s, t \in S_U$. Then $(st)U = s(tU) \subseteq s(Ut) = (sU)t \subseteq (Us)t = U(st)$.

Diversion C.6.4. The nominalization is similar to the normalization of a subgroup in a group, hence the name.

C.6.2 Image semigroups and congruences

Image semigroups, or equivalently, congruence semigroups can be practical for associative key agreement if:

- the base (original) semigroup is efficient,
- test equivalence $s \sim t$ is efficient, and
- finding the a unique representative of each congruence class is efficient.

C.6.2.1 Semigroup image of an arbitrary magma

To be revised!

Recall that a **magma** is a set with binary operation. Every semigroup is a magma. But an arbitrary binary operation is not required to be associative, so a magma is not necessarily a semigroup.

As with semigroups, we default to multiplicative notation for magmas.

Magmas form a category, with morphisms being functions that respect the magma multiplication. For each magma M, there exists a semigroup S, unique up to isomorphism, and a morphism $f : M \to S$, with the following (universal) property. For any semigroup T and any morphism $g : M \to T$, there exists a unique morphism $h : S \to T$ such that $g = h \circ f$. We call this S the **semigroup image** of M.

One way the to show the existence of S above is through the use of a **congruence** on M: a congruence is an equivalence relation $\sim$ on M such that $a \sim c$ implies $ab \sim cb$ and $ba \sim bc$. Any intersection of any (infinite) family of congruences is a congruence. Each congruence $\sim$ defines a **quotient magma** $M/\sim$ on the equivalence classes. The semigroup image is $M/\sim$ where $\sim$ is the intersection of all congruence magmas whose quotient magma is a semigroup.

The semigroup image, as described above, is generally not at all practical, because it may be quite difficult to determine equivalency of the equivalence class representations.

C.6.2.2 Mediality (internal commutation)

We can define a relation ρ on S called **mediality** (or **internal commutation**), by saying $(abcd)\rho(acbd)$ for all $a, b, c, d \in S$ (not necessarily distinct). Then $T = S/\sim_\rho$ will be **medial** (or **internally commutative**), meaning $abcd = acbd$ for all $a, b, c, d \in T$. It there exists a multiplicative identity $1 \in T$, and T is medial (internally commutative), then T is commutative (put $a = d = 1$).

Diversion C.6.5. The congruence semigroup above is a generalization of the derived group $G/[G, G]$ in group theory.

Diversion C.6.6. I learned the term **medial** from the Wikipedia page about a medial magma. The term **internally commutative** is what I previously used before learning of the term **medial**.

According to Wikipedia, there are many other terms, even including **entropic**, for the very same idea.

Mediality (internal commutativity) of a semigroup is equivalent to the to the ability of switching of the order of production (or summation in additive groups):

$$\prod_{i=1}^{m}\prod_{j=1}^{n} a_{i,j} = \prod_{j=1}^{n}\prod_{i=1}^{m} a_{i,j}. \tag{C.6.3}$$

To see this: put $m = n = 2$ and $[a_{1,1}, a_{1,2}, a_{2,1}, a_{2,2}] = [a, b, c, d]$ to deduce mediality from commutative production; expand both sides of the products using internal commutative (mediality) to re-order the internal factors, while the keeping the external factors $a_{1,1}$ and $a_{m,n}$ the same on both sides.

C.6.2.3 Merged subsets

For any subset $T \subseteq S$, the merger semigroup S/T, defined in §B.5.5, renders all elements of T equivalent.

Diversion C.6.7. As an example, let $S = \mathbb{P}_0^+$, the semigroup of non-negative integers under addition, and $T = \{2, 3\}$. Then S/T has three elements with representatives $\{0, 1, 2\}$, and addition table:

+	0	1	2
0	0	1	2
1	1	2	2
2	2	2	2

(C.6.4)

because if $2 \sim 3$, then $3 = 2 + 1 \sim 3 + 1 = 4$, so $2 \sim 3 \sim 4 \sim 5 \sim \ldots$ and this infinite class is represented by 2.

C.6.3 Converse (reversal) semigroups

If S is a semigroup, with multiplicative notation. Let S' be converse defined as follows, with binary operation $*$. Semigroup S' shares the same underlying set of S. Then $s * t = ts$.

C.7 Semigroup from presentations

The group theory notion of a presentations generalizes to the setting of semigroups.

A semigroup presentation has the form

$$\langle a, b, c, \dots | d = e, f = g, \dots \rangle = W_+(a, b, c, \dots)/ \sim_{\{(d,e),(f,g),\dots\}}, \tag{C.7.1}$$

meaning that is the semigroup of words in alphabet $\{a, b, c, \dots\}$ under the minimal congruence such that $d = e$, $f = g$, where each of $d, e, f, g, \dots,$ is word in the alphabet $\{a, b, c, \dots\}$.

All semigroups have a presentation, although the presentation may require infinitely many generators and relations.

C.8 Extending a semigroup

If S and T are semigroups with $S \subseteq T$, then T is said to be an **extension** of S. If clear from context, if T is also said to be **extension** of S, if $S \subseteq T' \cong T$, or, equivalent, that T has a subsemigroup isomorphic to S. For practical reasons, we focus on such extensions where the necessary isomorphisms are efficient (in both directions), such as minor notational changes.

To avoid confusion, we may refer to the subsemigroups S as **base** semigroup, or **given** semigroups, or **original** semigroup, to distinguish it from the constructed the extension semigroup T.

Diversion C.8.1. A more precise meaning of an extension, at least in the category theory setting, is a semigroup T together with injective semigroup morphisms $f : S \to T$, which makes clearer which the copy of S in T is used to define the extension.

The constructions here are a little more practical than this, and simply describe T and implicitly describe how the S_i are embedded into the T constructed.

Diversion C.8.2. To consider two different extensions T and U of S formally equivalent, means to ask for an isomorphism $h : T \to U$ such that the action h on the copies of S in each of T and U is the identity function.

C.8.1 Solitary extensions

A **solitary extension** of S is a semigroup T such that $T = S \cup t$ for some $t \notin S$.

Two solitary extensions are special: if t the identity in T, and if t is absorbing in T. When S uses multiplicative notation, write S_0 for the solitary extension with $t = 0$ absorbing, and write S_1 for the solitary extension with $t = 1$ the identity element. Call S_0 the **extension by zero** (or **extension by**

an absorbing element). Call S_1 the **extension by one** (or an **extension by an identity element**).

Diversion C.8.3. The standard notation in semigroup theory S^1 and S^0. This report uses subscripts in place of superscripts instead for a few reasons. One is to avoid inconsistency with the notation for Cartesian powers. Another is that this report saves the superscript position to indicate the binary operation (unless it is unwritten).

For additive semigroup S^+, a different notation will be used: S^+_0 for an extension by an identity element, and S^∞ for

Diversion C.8.4. Extension by an absorbing element and by an identity element are each special cases of the order union (see §C.9.1.2). For example,

$$S_0 = 1 \triangleleft S, \tag{C.8.1}$$

$$S_1 = S \triangleleft 1. \tag{C.8.2}$$

Diversion C.8.5. If S already has an absorbing element, say o, then S_0 has a new absorbing element 0, with $0 \neq o$, and $0o = 0 = o0$, so the old absorbing element is no longer absorbing in the extension. (We could say that 0 usurps the role of absorbing element from o.

Similarly, if S already has an identity i, then i is no longer an identity element in S_1.

Diversion C.8.6. A notational conflict naturally arises when extending by an absorbing or identity element. For example, if S already has an identity element, then it would have been natural to label the identity of S as 1. To avoid conflict with the new identity element in S_1, we can write 1_S for the old identity of S.

Diversion C.8.7. If we start from an arbitrary semigroup S, then $(S_0)_1$ and $(S_1)_0$ are isomorphic. In other words, the two types of extensions essentially commute. So, iterations extensions of S by zero or one, all that matters up to isomorphism is the number of extensions by each type, zero or one. This suggest we can write $S_{0^a 1^b}$ to indicate a general iteration of such extensions.

Assuming that existence of an identity element or of an absorbing element in a semigroup often greatly simplifies their analysis. Indeed, semigroups with an identity element are called monoids, and have a simpler theory. For example, any medial monoid is commutative, whereas there exists non-commutative medial semigroups. Extensions by zero or one can thus be seen as a small price to pay for brining the semigroup into a nicer class of algebraic structures.

When a semigroup already has an identity element (or an absorbing element), added a new identity element (or a new absorbing element) often seems unnecessarily seems redundant, at least if one's aim is just to simplify an analysis. In this case, we can conditionally extend S. Let $S_{(0)} = S$ if $0 \in S$, and let $S_{(0)} = S_0$ if $0 \not S$. In other words, $S_{(0)}$ is guaranteed to have a zero, but does not create a new zero unless needed.

Diversion C.8.8. In some semigroups, there exists solitary extension which are intermediate in nature between zero and one.

Suppose that S partitioned into disjoint sets O and I such that $OI \subseteq I$, and $OO \subset O$ and $II \subseteq I$. Equivalently, O is an ideal of S, and S/O has no zero-divisors, so O is kind of like a prime ideal. Equivalently, S is an ordered union $S = O \lhd I$, as defined in §C.9.1.2.

Add a new element $e = e_{O,I}$ to S such that

$$eo = o = oe$$
$$ie = e = ei$$

for all $o \in O$ and $i \in I$. Because $I \cup e \cong I_0$ and $O \cup e \cong O_1$, the new element e acts like both types of solitary extensions (by zero and by one). The resulting semigroup can be written as S_e. In terms of ordered unions, we have $S_e = O \lhd \{e\} \lhd I$.

Another type of generic solitary extension is a **shadow extension**. For each $u \in S$, define a solitary extension S_y, the **shadow extension at** u, such that:

$$yy = uu$$
$$ay = au$$
$$ya = ua$$

for all $a \in S$. In other words, whenever y appears in any multiplication operation it gets immediately replaced by u as an input to the multiplication operation, so that y does whatever u does, as if a shadow.

Diversion C.8.9. Another way to view the shadow extension is as a subgroup of a Cartesian product semigroup $S \times \{0, \epsilon\}^*$, see §C.9.1.4, where $*$ is defined on $\{0, \epsilon\}$ to mean $b * c = 0$ for all $b, c \in \{0, \epsilon\}$. In other words, $\{0, \epsilon\}^*$ is a **zero** or **null** semigroup of size two, with 0 as the zero. Equivalently, 0 is an absorbing element, and ϵ is nilpotent with $\epsilon^2 = 0$.

The subsemigroup is $T = \{[s, 0] : s \in S\} \cup \{u, \epsilon\}$. Elements $[s, 0]$ in T act as a copy of S, since $[a, 0][b, 0] = [ab, 0 * 0] = [ab, 0]$, while $[u, \epsilon]$ acts as the shadow of $[u, 0]$, since $[a, 0][u, \epsilon] = [au, 0 * \epsilon] = [au, 0] = [au, 0 * 0] = [a, 0][u, 0]$, and, similarly, $[u, \epsilon][u, \epsilon] = [uu, \epsilon * \epsilon] = [uu, 0] = [uu, 0 * 0] = [u, 0][u, 0]$.

C.8.2 The semigroup of subsets

Subsets of S can be multiplied. The resulting semigroup can be written 2^S.

Diversion C.8.10. The copy of S inside 2^S consists of singleton subsets, so the copy of a is $\{a\}$.

Diversion C.8.11. The empty step $\{\}$ is an absorbing element of 2^S.

Semigroup 2^S has an identity element if and only if S does, in which case, the singleton set $\{1\}$ is the identity of 2^S.

Diversion C.8.12. The set $T = 2^S_{\neq\{\}} = 2^S_{\#>0}$ of non-empty subsets is a subsemigroup of 2^S. If $0 \in S$ (is absorbing in S), then $\{0\}$ is absorbing in T. The subset S is an absorbing element of T if and only if S is a group.

Diversion C.8.13. When S is infinite, one can consider the subsemigroup $2^S_{\#<\infty}$ consisting only of finite subsets of S.

Diversion C.8.14. The semigroup 2^S can be made into a semiring, by defining addition as the union operation. With this view, 2^S is the multiplicative semigroup of a more general construction, a semigroup semiring $\mathbb{B}[S]$, discussed elsewhere in this report.

Diversion C.8.15. It is also possible to consider finite multisets (sets with repeated elements). In this case, we may view this as the multiplicative semigroup semiring $\mathbb{N}[S]$.

C.8.3 Stickel semigroups?

The Stickel key exchange protocol can be translated into an extension of a given a semigroup S.

To be completed.

Let L and R be two commutative subsemigroups of S. Typical examples of L and R are $L = \langle l \rangle$ and $R = \langle r \rangle$ for some $l, r \in S$, where preferably $lr \neq rl$.

Form a set $T = \{0\} \uplus S \uplus (L \times R)$, with elements written 0, or $[s]$ for $s \in S$, or $[u, v]$ for $u \in L$ and $v \in R$. A generic element of T is written as an un-bracketed variable, such as t. The set T is larger than S in the sense that T contains a copy of S as a strict subset.

Define multiplication on T by:

$$0t = t0 = 0, \tag{C.8.3}$$

$$[s][u] = 0, \tag{C.8.4}$$

$$[s][u, v] = [u, v][s] = [usv], \tag{C.8.5}$$

$$[u, v][w, x] = [uw, vx] \tag{C.8.6}$$

Lemma C.8.1. *T is a semigroup.*

Proof. The aim is to prove associativity, $a(bc) = (ab)c$ for all $a, b, c \in T$.

Multiplication in T is commutative, so one of the following cases analysis has been simplified.

- If $0 \in \{a, b, c\}$, then $a(bc) = (ab)c = 0$.
- If two of a, b, c belong to S, then $a(bc) = (ab)c = 0$.
- If $b \in S$ with $b = [s]$ and $a, c \in L \times R$ with $a = [u, v]$ and $c = [w, x]$, then $a(bc) = [u, v][wsx] = [uwsxv] = [wusvx] = [usv][w, x] = (ab)c$.
- If $a \in S$ with $a = [s]$ and $b, c \in L \times R$ with $b = [u, v]$ and $c = [w, x]$, then $a(bc) = [s][uw, vx] = [uwsvx] = [wusvx] = [usv][w, x] = (ab)c$.
- If $c \in S$ and $a, b \in L \times R$, then apply the argument above with a and c swapped, with $a(bc) = (cb)a = c(ba) = (ab)c$.
- If $a, b, c \in L \times R$, then associativity follows the associativity of the Cartesian product of semigroups L and R.

□

C.8.4 Action extensions?

A semigroup S acts on itself by left multiplication. This creates a morphism from S into the semigroup of functions S^S. A semigroup S also acts on $T = S_{(1)}$ faithfully, meaning that semigroup T^T contains S as an isomorphic copy.

So T^T is a natural extension of S, though not necessarily practical.

The semigroup T^T has two copies of S, one, λ_S from left multiplication, one, ρ_S, from right multiplication. These semigroups pairwise commute with each other. The smallest semigroup contains both these copies of S is therefore $\lambda_S \rho_S$, and is a natural extension of S.

The functions in T^T can also be viewed as relations on T. The set of relations on T forms a semiring. If R is the smallest semiring containing the copy of S, then the semigroup $R^\times$ is a natural extension of S.

Relations can also be described as Boolean matrices, using standard matrix multiplication with infinitary Boolean sums.

Diversion C.8.16. In some cases, the Boolean matrices can be replaced with matrices over some other semiring instead of the Boolean semiring.

C.8.5 Semigroups of ratios?

To be completed.

C.8.5.1 Free ratios

In this section, we consider semigroups intermediate in nature between two notions: free semigroups and free groups.

Let X be any set. Begin with free semigroup F generated by X^2, whose elements have the form $[a, b][c, d] \dots [e, f]$, for $a, b, \dots, f \in X$. We will then consider various congruences on F. The congruence class of $[a, b]$ will be written $[a : b]$. The congruences are generated by relations that make the pairs multiply as though they were like ratios. An element $[a : b]$ will be called a **ratio** over S, with the type of ratio depending of course on the congruence $\sim$.

A first relation is **middle cancellation**:

$$[a : b][b : c] = [a : c]. \tag{C.8.7}$$

A second relation is **right identity**:

$$[a : b][c : c] = [a : b]. \tag{C.8.8}$$

The congruence generated by the middle cancellation and right identity relations defines a **right ratio**, $[a : b]$.

Diversion C.8.17. As a congruence class, the right ratio $[a : b]$ is larger than the middle ratio $[a : b]$.

The semigroup of right ratios might have the following structure: a Cartesian of a left semigroup on X (with $xy = x$ for all $x, y \in X$, and a subsemigroup of the free group generated by X, where the subsemigroup is defined as product that strictly alternate between generator and an inverse of a generator. Right ratio $[a : b]$ maps to $[a, ab^{-1}]$. The product $[a : b][c : d] \dots [y : z]$ maps to $[a, ab^{-1}cd^{-1} \dots yz^{-1}]$. Conversely, $[a, bc^{-1} \dots yz^{-1}]$ can be converted to a product of right ratios as $[a : a][b : c] \dots [y : z]$. The semigroup of right ratios has a monography: each product can be written as $[a_0 : a_0][a_1 : a_2] \dots [a_{2n-1} : a_{2n}]$ where $a_i \neq a_{i+1}$ for $0 < i < 2n$.

The semigroup generated by right ratios is peremptory: it has a single divider $/$, this is is both a pre-divider and post-divider, given by

$$d[b_1 : b_2] \dots [b_{2m-1} : b_{2m}] = d[b_{2m} : b_{2m-1}] \dots [b_2 : b_1].,\tag{C.8.9}$$

for any d.

A third relation is **middle identity**:

$$[a : b][c : c][d : e] = [a : b][d : e]. \tag{C.8.10}$$

The congruence generated by the middle cancellation and middle identity defines a the **middle ratio** $[a : b]$.

Diversion C.8.18. Right identity implies middle identity. Consequently, the congruence for middle ratios is finer, the congruence class for middle ratio $[a : b]$ is contained with the congruence for the right ratio $[a : b]$.

The semigroup generated by middle ratios might have a similar structure to the semigroup of right ratios. Indeed, perhaps it just a Cartesian with the semigroup of right ratios with the a right semigroup on X, mapping product of middle ratios $[a : b] \dots [y : z]$ to the pair $[[a : b] \dots [y : z], z]$ whose first term is a product of right ratios.

Diversion C.8.19. Yet another weaker type of cancellation is **mirror cancellation**:

$$[a : b][c : c][b : a] = [a : a] \tag{C.8.11}$$

Mirror cancellation is implied by middle identity and middle cancellation. The mirror ratio $[a : b]$ is the congruence class of $[a, b]$ for the congruence generated by the middle cancellation and mirror cancellation.

The congruence generated middle and mirror cancellation is therefore finer the congruence for middle ratios. For example, if $a \neq b$, then mirror ratio $[a : b]$ congruence class is just a singleton set $\{[a, b]\}$.

Despite these rather fine congruences classes, the semigroup of mirror ratios is close to having quite good inverses. Given $b = [c : d] \dots [y : z]$, let $q = [z : y] \dots [d : c]$. Then $bq = [c : c]$ and $qb = [z : z]$. Also, b and q are mutual inverses.

C.8.5.2 Multiplicative ratios

Now we take $X = S$, a semigroup, and try to relate the multiplication in S in a meaningful to the various semigroups of ratios from the previous sections (which treat X as arbitrary set).

We do this imposing the further relations that

$$[ab : b] = [ac : c] \tag{C.8.12}$$

for all a, b, c and extending the congruence used for ratios, as necessary. The purpose is to make the map

$$a \mapsto [ac : c]$$

well-defined, and also a semigroup morphism. To that is as semigroup morphism, compute $[ac : c][bc : c] = [abc : bc][bc : c] = [abc : c]$, using the the middle cancellation law in the second equation.

To be completed.

C.8.5.3 Group of fractions (Grothendieck)

If S is any semigroup, it is natural to generate its **group of fractions** as follows, following an idea of Grothendieck.

We start with free sum $S \boxplus S'$ of the semigroup S, where S' is the converse S' (with element s' with $s \in S$, and multiplication switching left and right, so $s't' = (ts)'$). We then a make a solitary extension by one. (For example, this 1 can be represented by the empty sequence if a sequence representation of the free sum is used.)

Then form a congruence $\sim$ on $(S \boxplus S')_1$ generated by relations of the form $ss' = 1 = s's$ $s \in S$. The result is a group. Writing $s'' = s$ for $s \in S$, then have $(ab \dots c)^{-1} = c' \dots b'a'$ for $a, b, \dots, c \in S \cup S'$.

To make S/S practical for associative key agreement, there needs to be a monography (a way to find a canonical representation). This depends very much on S.

Diversion C.8.20. If S is commutative, then its Grothendieck group is commutative. For example, we can prove that $ab = ba$ for $a = [s, t']$ and $b = [t]$, as follows $[s, t'][u] = [s, t', u] = [s, u, u', t', u] = [su, u't', u] = [us, t'u', u] = [u, s, t', u', u] = [u, s, t'] = [u][s, t']$.

Consequently, if S is commutative, every element of the group of fractions in can represented in one of the forms $1 = []$, or $[s]$ or $[t']$ or $[s, t']$, where $s \in S$ and t' is in the converse (reversal) of S.

So, if S is commutative, then S/S is quite close to have a a unique canonical form for each element.

Diversion C.8.21. The semigroups of multiplication ratios of the previous section map into the group of fractions, by sending each ratio $[a : b]$ to ab' in the group of fractions.

The group of fractions can perhaps be equivalently describing by imposing some further relations on the ratios, such as the relation $[a : ca] = [b : ba]$ for all a, b, c.

C.8.6 Rees matrix semigroups

Recall the definition of a Rees matrix semigroup.

Definition C.8.1. *Let S be a semigroup. Let I, J be non-empty sets. Let $m : J \times I \to S$ be any (binary) function, which we call the **matrix**. The*

Rees matrix semigroup $\mathrm{Ree}_m(S)$ *has underlying set* $I \times S \times J = \{[i, s, j] : i \in I, s \in S, j \in J\}$ *and semigroup multiplication operation:*

$$[i, s, j][k, t, l] = [i, sm(j, k)t, l]. \tag{C.8.13}$$

We call S *the* ***base semigroup*** *of* $\mathrm{Ree}_m(S)$*, and call the* I *the* ***left index set*** *and* J *the* ***right index set*** *of* $\mathrm{Ree}_m(S)$*.*

Rees proved that every **completely simple** semigroup was a Rees matrix semigroup whose base semigroup was a group.

Division in the Rees matrix semigroup is discussed in §5.10; wedges in §4.7.

C.8.6.1 Generalization to Munn algebras?

To be completed.

Rees semigroups generalize to Munn algebra, also called Rees algebra. Rees semigroups are subsemigroups of the multiplicative semigroup of a Munn algebra.

A Munn algebra M are defined as the set of matrices of a fixed rectangular shape, with entries in some base ring. Addition is usual matrix addition. To define multiplication in M, pick a fixed matrix P in M, and then define M-multiplication $A * B$ via the equation $A * B = AP^t B$, using standard matrix multiplication and P^t means matrix transpose.

The Rees semigroup corresponds to a subset of M for a special base ring. The base ring is a semigroup semiring $R[S]$. The subset of M are the elementary matrices, where at most one matrix entry is nonzero, and the nonzero entry belongs to S (assuming R has an element 1). Finally, the matrix P can have many nonzero entries, but they all belong to S (the copy of S inside $R[S]$). The Rees semigroup multiplication is then the Munn algebra multiplication.

Note that P is often called the sandwich matrix in a Munn algebra, presumably because it sandwiched between the two other matrices. This is somewhat related to the wedge operation, as explained below.

Consider the larger semiring of all matrices of arbitrary shape. Defining addition and multiplication as usual, when the matrix shapes are compatible. When the matrix arguments are shapes incompatible for addition of multiplication, define the result to be some error value, such as ∞, or an empty matrix. Then the Munn product is a sandwich product $(A, C) \mapsto APC$.

The sandwich product is similar to the wedge product as though P were the inverse matrix of B.

C.8.7 Semigroups of special functions on a semigroup

In the following constructions, we begin with a single base semigroup S, and construct another semigroup, which is usually larger.

C.8.7.1 Semigroup of transformations

Let S be a semigroup. (Or, more generally, let S be any set.)

The semigroup S^S consists of all functions from S to itself. Its multiplication is standard functions composition. This means that $h = fg$ is defined by $h(s) = f(g(s))$ for all $s \in S$.

It is easy to see that S^S is a semigroup. In some sense, S is on the archetypal semigroups, after integer arithmetic.

Diversion C.8.22. The semigroup S^S is unital (a monoid), multiplicative identity 1 being the function $1(s) = s$ for all $s \in S$.

Diversion C.8.23. The semigroup S^S does not depend directly on the multiplication in S. It only depends on the underlying set of S.

For further discussion, in the context of more general sets, see §C.5.5.

C.8.7.2 Semigroup of endomorphisms

Let S be a semigroup. Form its semigroup of **endomorphisms** as follows.

Diversion C.8.24. Endomorphisms sometimes form semiring, not just a semigroup, as described in §C.17.2.

Let $E(S)$ be the set of semigroup morphisms $h : S \to S$. (A morphism from S to itself is an **endomorphism** of S.) Multiplication in $E(S)$ is just function composition.

Lemma C.8.2. *$E(S)$ is a semigroup.*

Proof. Endomorphisms are closed under function composition because $(gh)(st) = g(h(st)) = g(h(s)h(t)) = g(h(s))g(h(t)) = (gh)(s)(gh)(t)$. Multiplication of endomorphisms is associative because function composition is associative. □

Diversion C.8.25. $E(S)$ is a subsemigroup of S^S.

Diversion C.8.26. Semigroup $E(S)$ is also a monoid: its multiplicative identity is the identity function $1 : s \mapsto s$.

Diversion C.8.27. Later we will that is S is commutative, and written additively, then $E(S)$ can be made into a semiring.

If S is unital multiplicative identity 1, then we define subset $E_1(S)$ of **unital endomorphisms** to be all endomorphism satisfying $h(1) = 1$. Unital endormorphisms are closed under multiplication, so $E_1(S)$ is a subsemigroup of $E(S)$.

Diversion C.8.28. If S is unital, and $u, v \in S$ satisfy $vu = 1$, then the function $h = h_{u,v} : s \mapsto usv$ is an endomorphism, since $h(st) = ustv = us1tv = usvutv = h(s)h(t)$.

Diversion C.8.29. The endomorphism construct generalizes. Consider any category C and any object X in the category. Then $\mathrm{Mor}(X, X)$ is the set of morphisms from X to itself, so it is the set endomorphisms of X. By the definition a category, the morphism can be multiplied associatively, and the endomorphisms of a single object are closed under multiplication. Hence $\mathrm{End}(X) = \mathrm{Mor}(X, X)$ is a semigroup, for any object in any category.

C.8.7.3 Semigroup of affine functions on a semigroup

Let S be a unital semigroup (a monoid). The semigroup $A(S)$ of **right-affine** functions on S is defined as follows.

Let $A(S)$ be the subset of semigroup S^S consisting of all **right-affine** functions, satisfying the rule:

$$f(ab)f(1) = f(a)f(b) \tag{C.8.14}$$

for all $a, b \in S$.

Lemma C.8.3. *$A(S)$ is a semigroup.*

Proof. It is a subset of S^S, so it suffices to prove that right-affine functions are closed under composition. If f and g are affine, then

$$f(g(ab))f(g(1)) = f(g(ab)g(1))f(1) = f(g(a)g(b))f(1) = f(g(a))f(g(b)). \tag{C.8.15}$$

□

Diversion C.8.30. An element $c \in S$ is **central** if $cs = sc$ for all $s \in S$, in other words if it commutes with all of S. The function $t_c : s \mapsto sc$ is right-affine.

Diversion C.8.31. A unital endomorphism is right-affine, so $E_1(S) \subseteq A(S)$.

Diversion C.8.32. A right-affine function f is unital if $f(1) = 1$. A unital right affine function is an endomorphism.

Diversion C.8.33. If we use additive notation for S, then endomorphism can be thought as multiplicative, and affine functions can be thought as linear (multiplication plus an translation).

C.8.7.4 Semigroups of metamorphic functions?

Probably wrong.

Let S be semigroup. Let $F = S^S$ be semigroup of endofunctions on S. The semigroup E of endomorphisms of S is a subset of F. The left nominalization $M = F_E = \{m : S \to S : mE \subset Em\}$ (see §C.6.1.8) is a subsemigroup of F, which we call the set of **metamorphic** or **monoform** functions of S.

Diversion C.8.34. The term **metamorphic** refers to the following. Let $me = \mu(e)m$, for some function μ. Then μ is almost anti-endomorphism of E, since, for $e, f \in E$, and $\mu(ef)m = mef = (me)f = \mu(f)me = \mu(f)me = \mu(f)\mu(e)m$. This suggest $\mu(ef) = \mu(f)\mu(e)$, unless right multiplication by m does not cancel.

For example, suppose $S = \mathbb{P}^+$. For positive integer n, there is an endomorphism $e_n : p \mapsto np$, and these are all endomorphisms.

For positive integers, consider the function $m_{a,d} : p \mapsto ap^d$. Then $m_{a,d}$ is metamorphic (monoform) because, $m_{a,d}e_n = e_{n^d}m_{a,d}$, for all n.

Diversion C.8.35. Functions $m_{a,d}$ are usually called **monomials**, which weakly justifies the name **monoform**.

C.9 Combining semigroups

Some semigroups are built from smaller semigroups. In particular, sometimes the smaller semigroups remain as subsemigroups of the constructed semigroup: this section discusses such constructions.

C.9.1 Combining two semigroups

Let S and T be semigroups.

We general assume that S and T are disjoint as sets. If not, then we work with disjoint copies of S and T, if needed.

C.9.1.1 Free sum of semigroups

The **free sum** $S \boxplus T$ of semigroups S and T is the semigroup whose underlying set of elements are finite non-empty ordered sequences whose entries belong the disjoint union $S \uplus T$, with the restriction that entries alternate between (the copies) S and T. Multiplication consists of concatenating sequences, except that if that results in consecutive sequence entries from the same base semigroup (either both in S or both in T), then the two entries by one entry, being product in the base semigroup. More formally,

$$\begin{aligned}&[u_1, \ldots, u_m][v_1, \ldots, v_n] \\ &\qquad = \begin{cases} [u_1, \ldots, u_m, v_1, \ldots, v_n] & \text{if } [u_m, v_1] \notin (S \times S) \cup (T \times T), \\ [u_1, \ldots, u_m v_1, \ldots, v_n] & \text{if } [u_m, v_1] \in (S \times S) \cup (T \times T), \end{cases}\end{aligned} \tag{C.9.1}$$

where the notation $S \times S$ and $T \times T$ refers the standard Cartesian products of sets.

Diversion C.9.1. To clarify, $S \boxplus S \neq S$, because the copies of S used are considered disjoint adjacent entries from different copies do not coalesce into their product.

Diversion C.9.2. An alternative notation might be clearer. For this, use addition notation for the base semigroups S^+ and T^+ (but allow otherwise arbitrary semigroups). Then we write the free sum as $\langle x^{S^+}, y^{S^+} \rangle$ (instead of $S^+ \boxplus T^+$), using multiplicative notation, with the understanding that $x^a x^b = x^{a+b}$ and $y^c y^d = y^{c+d}$, but with formal powers of x and y not interacting other than by multiplication.

An example element is like $x^a y^b x^c$, where a normal form is used, to make the formal bases alternate between x and y. The formal exponents in the example have $a, c \in S^+$ and $b \in T^+$. In this case, $x^a y^b x^c$ is the alternative notation for $[a, b, c]$.

Diversion C.9.3. The free sum $S \boxplus T$ contains a copy of S and a copy of T, in the form one-element sequences. Therefore, $S \boxplus T$ may be viewed as an extension of both S and T (and S and T may be viewed as subgroup of S and T). More precisely, there exists natural semigroup morphisms $n_S : S \to S \boxplus T : s \mapsto (s)$, and n_T. These semigroup morphisms are injective.

Diversion C.9.4. An alternative definition $S \boxplus T$ is as a congruence semigroup.

Start with the free semigroup $F^+(S \uplus T)$ generated by the set $S \uplus T$, which is the set of all sequences with entries in set $S \uplus t$, with multiplication being sequence concatenation.

Then define a congruence $\sim$ on $F^+(S \uplus T)$ such that adjacent sequence entries from the same base semigroup (the copy of S or the copy of T) are to be multiplied into a single sequence entry.

In this case, $S \boxplus T = F^+(S \uplus T)/\cong$. The alternating sequences are the canonical representatives of the congruence classes (and serve a practical monogram).

Diversion C.9.5. The free sum is actually categorical co-product, in the sense of category theory. Specifically, if U is any semigroup and both $f : S \to U$ and $g : S \to U$ are semigroup morphisms, then there is a unique semigroup morphism $h : S \boxplus T \to U$, such that $f = hn_S$ and $g = hn_T$. (To be verified.)

Diversion C.9.6. If we modify the definition by allowing the copies S and T to intersect, then we might lose the ability to find a canonical normal form with alternating entries from S and T.

In category theory, this would be an attempt at what is known as the **fibered co-product** or **pushout** or the **amalgamated sum**.

To be completed.

Diversion C.9.7. Other possible names for the free sum are **compositum** (which I borrow from field theory), **free product**, and **ordered product** (for its parallel to ordered union).

Diversion C.9.8. The free sum is associative, up to semigroup morphisms: $S \boxplus (T \boxplus U) \cong (S \boxplus T) \boxplus U$. (With suitable opening of nested sequences, then this can be considered an equality, not just an isomorphism.)

Diversion C.9.9. The empty semigroup class 0 obey $S \boxplus 0 = 0 \boxplus S = S$ for any semigroup class S.

Diversion C.9.10. The free sum is commutative: $S \boxplus T = T \boxplus S$.

The alternative notational form of free sum has $\langle x^{S^+}, y^{T^+}\rangle = \langle y^{T^+}, x^{S^+}\rangle$, so could be considered commutative too. However, we also have $\langle x^{S^+}, y^{T^+}\rangle \neq \langle x^{T^+}, y^{S^+}\rangle$ because of the notational change (in otherwise isomorphic semigroups).

Diversion C.9.11. It is possible to have a semigroup such that $S \cong S \boxplus 1$, where 1 is the unit semigroup (of one element). In this case, the elements can be most natural represented as sequences of positive integers such that no adjacent entries are equal. To multiply, concatenate except that adjacent repeated entries are replaced by a single copy. Essentially, this S is generated by infinitely many idempotents, as freely as possible.

If S and T are monographic, then so is $S \boxplus T$: just apply the monographies of S and T to each entry.

Efficiency of multiplication in $S \boxplus T$ is the minimum of efficiency of multiplication in S and T, plus also the cost of accommodating of potentially long sequences.

C.9.1.2 Ordered union of semigroups

The **ordered union** $S \triangleleft U$ of semigroups S and T is semigroup whose underlying set is their disjoint union of sets $U = S \uplus T$. Multiplication in $S \triangleleft T$ is to be the multiplication in S and T, if both inputs belong to the same copy of a semigroup, and otherwise the output is the same as whichever the input belongs to S. That is:

$$uv = \begin{cases} uv & \text{if } u, v \in S, \\ uv & \text{if } u, v \in T, \\ u & \text{if } u \in S, v \in T, \\ v & \text{if } u \in T, v \in S. \end{cases} \tag{C.9.2}$$

Associativity follows from the following general observation. To compute $u_1 \dots u_n$ as follows. If all $u_i \in T$, then compute in T, where associativity holds (so the parentheses can be placed anywhere in the product). If at least $u_i \in S$, then ignore all $u_j \in T$, and keep only those $u_k \in S$, say $u_{k_1} \dots u_{k_m}$. Then compute in S, where associativity hods (so the parentheses can be placed anywhere in the full product).

Diversion C.9.12. In any subsemigroup, a relation $\subset$ can be defined such that $o < i$ if $oi = io = o$, see §B.6.3 for details.

In $S \triangleleft T$, we have $s \subset t$ for all $s \in S$ and $t \in T$. Hence the term **ordered union**.

Diversion C.9.13. There are natural morphisms $S \to S \triangleleft T$ and $T \to S \triangleleft T$.

Diversion C.9.14. The ordered union operation respects the isomorphism class of semigroups.

Diversion C.9.15. On semigroup isomorphism classes, it is associative. Semigroup classes form a semigroup under zero-one merger. On semigroup classes, the empty semigroup 0 obeys the rule $0 \triangleleft S = S \triangleleft 0 = S$.

So, semigroup classes form a unital semigroup (a monoid) under the ordered union operation.

Diversion C.9.16. An ordered union of orderly semigroups is orderly. Hence orderly semigroup classes form a subsemigroup of the semigroup of semigroup class under the ordered union operation.

Diversion C.9.17. There are likely already different names for the ordered union.

Diversion C.9.18. The ordered union $S \triangleleft T$ is an image of the free sum $S \boxplus T$, through a natural morphism $[u_1, \ldots, u_m] \mapsto u_1 \ldots u_m$.

If S and T have monographies (unique canonical element representations), then so does $S \triangleleft T$. Similarly, efficiency of multiplication is the minimum of the efficiencies in S and T.

C.9.1.3 Disjunction of semigroups

Let S and T be disjoint semigroups, form a **disjunction** $S \uplus_0 T$, a semigroup whose underlying set is the disjoint union $S \uplus \{0\} \uplus T$ (a copy of each S and T, plus a formal zero element, distinct from any zero elements already in S or T). Product within the copy S use multiplication in S, those within T use multiplication from T, an all products evaluate to 0 (in the term $\{0\}$ of the disjoint union).

The disjunction $D = S \uplus_0 T$ may also be defined as a congruence semigroup, taking an image of the free sum $U = S \boxplus T$ (defined in §C.9.1.1) as follows. Let I be the subset of U consisting of all sequence of length two or more. This is a semigroup ideal (see §B.5.7), and thus defines a congruence $\sim_I$. Then the disjunction $D = S \uplus_0 T$ is isomorphic to $U/ \sim_I = U/I$. The elements of I represent the formal 0 of $S \uplus_0 T$.

Diversion C.9.19. If one or both of S and T already have a zero element, then a **reduced disjunction** $S \uplus T$ can be defined, just the same as $S \uplus_0 T$, except that there is no new formal zero, and instead the zero is re-used.

Actually $S \uplus_0 T = S_0 \uplus T$, where S_0 is the extension of S by the zero element.

C.9.1.4 Cartesian products of semigroups

Let S and T be semigroups. Let $S \times T$ be the semigroup whose set is the Cartesian product consisting of pairs $[s, t]$ and whose multiplication is defined as $[s, t][s', t'] = [ss', tt']$.

Diversion C.9.20. The same terminology **Cartesian product** and the same notation $S \times T$ will be used for both sets and semigroups. Usually this overlap should result in no confusion, because the meaning overlap considerably: after forgetting the multiplication the two notions become the same. If confusion nonetheless results from this overlap of meanings, then clarification can be added.

Diversion C.9.21. In some cases, a semigroup U might be isomorphic to $S \times T$, but evaluating one or both directions of the isomorphism might be difficult.

For example, let p and q be distinct large primes, and take $S = \mathbb{Z}_p$ and $T = \mathbb{Z}_q$ and $U = \mathbb{Z}_{pq}$ (where $\mathbb{Z}_m$ is the semigroup $\{0, 1, \ldots, m-1\}$ under multiplication modulo m). Given $U = \mathbb{Z}_m$, finding p and q amounts to the integer factorization problem, which can be difficult (without a quantum computer).

Most of this section discusses the case where the structure of $S \times T$ is evident (not hidden difficult by a difficult isomorphism).

Multiplication in $S \times T$ has cost which the sum of its cost in S and T. (So, efficiency is the inverse sum of inversed efficiencies, which is usually at least half the minimum of the two based efficiencies.)

Monography in $S \times T$ is achieve through monography in each of S and T individually.

Diversion C.9.22. If semigroup T contains an element e that is idempotent, meaning $e^2 = e$, then $S \times T$ contains a copy of S as subsemigroup, namely the set $S \times e = \{[s, e] : s \in S\}$.

Diversion C.9.23. The Cartesian product $S \times T$ is an image of a subsemigroup of the free sum $S \boxplus T$.

Let I be the subset of $S \boxplus T$ consisting of sequence of length two or greater. Then I is a subsemigroup. Define a semigroup morphism $h : I \to S \times T$ by the rule

$$[u_1, \ldots, u_m] \mapsto [u_{s_1} u_{s_2} \ldots, \quad u_{t_1} u_{t_2} \ldots] \tag{C.9.3}$$

for s_i and t_i such that $s_1 < s_2 < \ldots$ and $t_1 < t_2 < \ldots$ and $u_{s_i} \in S$ and $u_{t_i} \in T$ and $\{s_1, s_2, \ldots, t_1, t_2, \ldots\} = \{1, 2, \ldots, m\}$.

Diversion C.9.24. If S^+ and T^+ are additive semigroups, then an alternative description of Cartesian product $S^+ \times T^+$ is to use the notation of formal powers, so that all elements have of the $x^s y^t$, for $s \in S^+$ and $t \in T^+$. Multiplication is then $(x^a y^b)(x^c y^d) = x^{a+c} y^{b+d}$, which may be described as saying that formal powers of x and y commute with each other.

Diversion C.9.25. The Cartesian product is commutative up to isomorphism, with the obvious isomorphism $T \times S \to S \times T : [t, s] \mapsto [s, t]$.

The Cartesian product is associative under the convention the sequence get un-nested with $[s, [t, u]] = [s, t, u] = [[s, t], u]$. Without this convention, the Cartesian product is associative up to isomorphism, with isomorphism $[s, [t, u]] \mapsto [[s, t], u]$.

For the empty semigroup, we have $0 \times S = 0$. For the unit semigroup 1, we have $1 \times S \cong S \cong S \times 1$.

Diversion C.9.26. If K and L are associative key agreement schemes, and S_K and S_L their associated semigroups, then $S_{K \times L}$ is essentially the semigroup associated with a key agreement scheme $K \times L$, in which Alice and Charlie use K and L in parallel and independently, using ordered pairs of secrets, deliveries and keys.

Diversion C.9.27. The Cartesian product of additive semigroups S^+ and T^+ is perhaps more naturally written as $\oplus$.

C.9.1.5 Non-commutative tensor products of semigroups

A **non-commutative tensor product** of semigroups can be defined is a few ways, equivalent except for notation (up to efficient isomorphism).

A major practicality issue with non-commutative tensor product can arise: finding a monography (a practical unique representation of elements) might be difficult.

Diversion C.9.28. The definitions with additive notation do not follow a common tradition of using additive notation only for commutative operations.

The first additive notation for the tensor product uses subscripted variables. Let S and T be semigroups with multiplicative notation. Let $\langle x_S y_T \rangle^+$ the (additive) semigroup generated by elements written as formal products $x_a y_b$ with $a \in S$ and $b \in T$. The generators obey the following two relations:

$$x_a y_b + x_a y_c = x_a y_{bc} \tag{C.9.4}$$

$$x_a y_c + x_b y_c = x_{ab} y_c. \tag{C.9.5}$$

If we view the formal symbols x_a as a formal logarithm of a to the base a, as though $x_a = \log_x(a)$, then relations above may be viewed as instances of the distributive law (of multiplication over addition). For example:

$$x_a y_b + x_a y_c = x_a(y_b + y_c) = x_a(\log_y(b) + \log_y(c)) = x_a \log_y(bc) = x_a y_{bc},$$

where only the first and last expression strictly belong to the tensor product. (The intermediate expression may be well-define in a richer algebraic structure.)

Diversion C.9.29. The subscripted notation of formal logarithms has advantage of being analogous to the superscripted notation for formal powers, which was used for semigroups in the free sum (co-product) and the Cartesian product. (Formal power notation is also used widely in mainstream algebra, such as in polynomial rings and power series rings.)

A second additive notation can be defined, and is to be used when taking the tensor product of additive semigroups. We write this, $(S^+ \boxtimes T^+)^+$. It is generated by elements that are formal products $a \boxtimes b$ with $a \in S^+$ and

$b \in T^+$, obeying the relations:

$$a \boxtimes b + a \boxtimes c = a \boxtimes (b + c), \tag{C.9.6}$$

$$a \boxtimes c + b \boxtimes c = (a + b) \boxtimes c. \tag{C.9.7}$$

Diversion C.9.30. The similarity between the two notations is that $x_a y_b$ can be rewritten written as $a \boxtimes b$, and adjusting the binary operator notations for S and T as needed.

Other more compact additive notations can be used, if clear from context. Instead of $a \otimes b$, one might write $[a][b]$ or even ab.

Diversion C.9.31. There may be universal properties of the tensor product of semigroups.

Function $b : S^+ \times T^+ \to U^+$ is a **bimorphic** function of semigroups if $b(s + s', t) = b(s,t) + b(s',t)$ and $b(s, t + t') = b(s,t) + b(s,t')$ for all $s, s' \in S$ and $t, t' \in T$.

Then there is a natural morphism $(S^+ \boxtimes T^+ \to U^+$ that sends column $s\boxtimes$ to $b(s,t)$, and extends to formal sums arrays additively.

Also, there is a natural bimorphism $S^+ \times T^+ \to (S^+ \boxtimes T^+) : [s,t] \mapsto s \otimes t$. The natural morphism and bimorphisms can be put into a commutative diagram with b, and may well be the case that the natural morphism is unique.

To be completed.

A multiplication notation for the non-commutative tensor products uses two-dimensional arrays. Given multiplicative semigroups, let $\binom{S}{T}$ be the semigroup defined as the following (congruence) semigroup.

The elements can be represented 2-row arrays, with the first row with entries in S and the second row with entries in T, but certain arrays are considered equivalent, by a series of steps called **column splitting**, or its opposite **column joining**. A single column $\binom{ac}{b}$ may be split into two columns $\binom{a}{b}$ and $\binom{c}{b}$ with this two columns replacing the original column. Splitting is also allowed in the second row: $\binom{a}{bd} \mapsto \binom{a}{b}\binom{a}{d}$. For example, a typical column-splitting looks like:

$$\begin{pmatrix} s_1 & \dots & s_i s_i' & \dots & s_n \\ t_1 & \dots & t_i & \dots & t_n \end{pmatrix} \sim \begin{pmatrix} s_1 & \dots & s_i & s_i' & \dots & s_n \\ t_1 & \dots & t_i & t_i & \dots & t_n \end{pmatrix}.$$

So, the relation $\sim$ can relate any two arrays A and B reachable by arbitrary sequence of column splittings, with each individual splitting occurring either in the first or second row, and each individual splitting occurring in either direction, splitting towards A or towards B.

Multiplication simply concatenation of the arrays, side-by-side. So, multiplication looks like this:

$$\begin{pmatrix} s_1 & \dots & s_m \\ t_1 & \dots & t_m \end{pmatrix} \begin{pmatrix} s_1' & \dots & s_n' \\ t_1' & \dots & t_n' \end{pmatrix} = \begin{pmatrix} s_1 & \dots & s_n & s_1' & \dots & s_n' \\ t_1 & \dots & t_m & t_1' & \dots & t_n'. \end{pmatrix}$$

Diversion C.9.32. The non-commutative tensor product is associative, up to isomorphism. The empty semigroup 0 is absorbing, and the unit semigroups is the identity. The non-commutative tensor product is commutative up to isomorphism: $S \boxtimes T = T \boxtimes S$.

The non-commutative tensor product seems to be distributive over the free-sum: $S \boxtimes (T \boxplus U) = (S \boxtimes T) \boxplus (S \boxtimes U)$, because it seems the 2-row matrix form, taking the top row with entries in S and the bottom row with entries in T or U.

C.9.1.6 Commutative tensor product of semigroups

The commutative tensor product of semigroups is the commutative image of the non-commutative tensor product.

We can use notation $\otimes$ to distinguish the commutative tensor product from the non-commutative tensor product (which uses $\boxtimes$).

Diversion C.9.33. The tensor product of vector spaces (or modules) is a standard definition from linear algebra. A vector space V includes an additive semigroup structure, which we could write as V^+. A vector space also has an extra scalar multiplication structure beyond this semigroup structure.

The vector space tensor product and commutative tensor product of semigroups are consistent enough to share the same notation, without much confusion.

If U and V are vector spaces (over the same field), then the vector space tensor product $U \otimes V$ has additive semigroup $(U \otimes V)^+ = (U^+ \otimes V^+)^+$, which the semigroup tensor product of their additive semigroups.

The commutative tensor product can also suffer from the same practicality issues as the non-commutative tensor product: no easy monography.

In some cases, the semigroups S and T themselves have a special monography. For example, consider the case of vector spaces, in which monography would be via coordinates as coefficients in a certain basis. This representation can be carried over to commutative tensor product as a matrix representation.

Diversion C.9.34. The commutative tensor product might be distributed over the Cartesian product of commutative semigroups.

C.9.2 Combining large families of semigroups

A **family** of semigroups, is a function f from a set, called the **index set** and often written I, to a set of semigroups. We usually omit the function name f and just write the family as $\{S_i\}_{i\in I}$, where S_i is a semigroup. (Formally, it would be $S_i = f(i)$.)

This section deals with constructions that turn a family of semigroups into a semigroup.

Diversion C.9.35. One may naturally ask of what cryptographic practicality is it to consider the generality of infinite families? In this regard, the underlying structure of more constructible subsemigroups may be related to such infinite families.

For example, the positive integers under multiplication, certainly a practical semigroup, is a subsemigroup of a Cartesian product of an infinitely family of semigroups (one semigroup for each prime), each a copy of the additive semigroup $\mathbb{N}^+ = \{0, 1, 2, \ldots\}$ of non-negative integers. The isomorphism involves factorization, so $40 = 2^3 \times 5 \mapsto [3, 0, 1, 0, 0, \ldots]$.

C.9.2.1 Free sums of arbitrary families

The **free sum** of an arbitrary family of semigroups has virtually the same definition as the free sum of two semigroups.

Take non-empty sequences with entries in the disjoint union of the family of semigroups as sets. Two adjacent sequences in the same semigroup can be replaced by one entry, their product. Each sequence has shortest length representation such no two adjacent entries belong to the same copy of a semigroup in the family.

Multiplication of sequences uses concatenation. When the factors are in shortest length form, either their concatenation is in shortest length form too, or the rightmost of the left factor and the leftmost entry of the right factor belong to the same copy of a semigroup in the family. In this case, the two entries can be combined into one entry using the required semigroup operation, so that the length of the concatenated is reduced by one, and the sequence will be the shortest length representative.

C.9.2.2 Cartesian products of large families

The **Cartesian product** of an arbitrarily family of semigroups is the following semigroup. Suppose $(S_i)_{i\in I}$ is any family of semigroups for some arbitrary index set I, possibly infinite. The product semigroup $\bigotimes_{i\in I} S_i$ has elements which are functions $s : I \to \bigcup_{i \in I} S_i$, with property that $s(i) \in S_i$. Multiplication is the usual rule $(st)(i) = s(i)t(i)$.

The **Cartesian power** S^I is a Cartesian semigroup in which each S_i is an isomorphic copy of a given semigroup S. Elements of the Cartesian power are simply functions $\sigma : I \to S$. Multiplication of these functions is point-wise S-multiplication, so $\sigma\tau : I \to S : i \mapsto \sigma(i)\tau(i)$.

Diversion C.9.36. When $I = S$, another associative binary operation can be defined on the set S^I of functions $I \to S$: function composition (see §C.5.5) – whose associativity is unrelated to that of S, since is function composition is associative for any set.

These two operations on S^S, when S is a semigroup, taken together, form a nearring, as explained in §C.17.1.

C.9.2.3 Direct products of semigroups

If each S_i is a unital semigroup (a monoid), then we can form a special subsemigroup of $\bigotimes_{i \in I} S_i$, written $\bigoplus_{i \in I} S_i$, and known as the **direct product**, as follows. Let 1_i be the identity element of S_i. For any $s \in \prod_{i \in I} S_i$, define the support $U(s) = \{i \in I : s(i) \neq 1_i\}$. Then let

$$\bigoplus_{i \in I} S_i = \left\{ s \in \prod_{i \in I} S_i : |U(s)| < \infty \right\}, \tag{C.9.8}$$

in other words, the elements of finite support.

For example, consider the semigroup $\mathbb{P}^*$ of positive integers under multiplication. Let $\mathbb{N}^+ = \mathbb{P}_0^+$ be the semigroup of non-negative integers under addition, which is a monoid with identity 0. For each $i \in \mathbb{P}$, let $\mathbb{N}_i^+$ be copy of $\mathbb{N}^+$. Then

$$\mathbb{P}^* \cong \bigoplus_{i \in \mathbb{P}} \mathbb{N}_i^+, \tag{C.9.9}$$

with the isomorphism using the prime factorization exponents.

C.9.2.4 Orderly unions of large families

Let $\{S_i\}_{i \in O}$ be a family of semigroups, indexed by the elements of an orderly semigroup O (§C.3.2). Then the **orderly union** is a semigroup

$$\biguplus_{i \in S} S_i \tag{C.9.10}$$

with elements represented by pairs $[i, s]$ with $i \in O$ and for $s_i \in S_i$ and multiplication rule:

$$[i, s_i][j, s_j] = \begin{cases} [ij, s_{ij}] & \text{if } i \neq j, \\ [ij, s_i s_j] & \text{if } i = j. \end{cases} \tag{C.9.11}$$

The orderly union generalized the ordered union. An ordered union is an orderly union where the index semigroup is $O = \{0, 1\}$ (the integers zero and one under multiplication). More generally, the ordered union is an associative operation up to isomorphism, being isomorphic to an orderly union with finite index set.

If O orderly semigroups, and each S_i in a family indexed by O is isomorphic to given semigroup S, then we call $\biguplus_{i \in O} S_i$ the **lexicographic** product $O \rtimes S$. Its operation can be written:

$$[i, s][j, t] = \begin{cases} [i, s] & \text{if } ij = i \neq j, \\ [i, st] & \text{if } ij = i = j, \\ [j, t] & \text{if } ij = j \neq i. \end{cases} \tag{C.9.12}$$

If each of the S_i is an orderly semigroup, then their orderly union $\biguplus_{i \in O} S_i$ is an orderly semigroup too. Therefore, if O and S are orderly, then so is their lexicographic product. The lexicographic product is associative on orderly semigroup classes (up to isomorphism).

The lexicographic product can also be generalized, in several interesting ways, to larger orderly families of orderly semigroups. However, orderly semigroups are idempotent, which makes division easy, which makes them not very useful for associative key agreement. So, we do not look further into these constructions.

C.10 Semigroups from semiautomata

A semiautomaton is a function $t : X \times Y \to X$, representing a device that maintains an internal state $x \in X$ which gets updated in response to an external input $y \in Y$.

For each fixed external input y, define a function $t_y \in X^X$ mapping states to states, by defining $t_y(x) = t(x, y)$.

The semigroup X^X of all transformations from X to X has a subsemigroup generated as $\langle t_y \rangle_{y \in Y}$ called the **transition semigroup** of the semiautomaton t. (A **transition monoid**, defined as $\langle 1_X, t_y \rangle_{y \in Y}$ can also be define, where $1_X \in X^X$ is the identity function with $1_X(x) = x$ for all $x \in X$.)

Diversion C.10.1. In cryptography, we can think of Y as space of keys, and then t is a set of keyed functions from the state space X to itself. Some symmetric-key primitives fit this model including block ciphers, and the compression functions of chained hash functions.

Diversion C.10.2. Some theory in the area characterizes a semiautomaton according to the algebraic structure of its transition semigroups (or monoid).

C.11 Semigroups from combinatorial graphs

There are several ways to combine two graphs. Some of these are commutative, forming a semigroup of graphs. Some of these graph operations can be found on Wikipedia:

[https://en.wikipedia.org/wiki/Graph_operations]

For example, the **Cartesian graph product**, written $G \Box H$ corresponds to the Kronecker sum of the adjacency graphs. If the graphs are labelled, then division is just Kronecker subtraction, which is easy.

If the graphs are un-labelled, or randomly re-labelled, then division might be a little more difficult.

To be completed.

C.12 Semigroups from combinatorial games

Recall that a (**combinatorial**) **game** is an ordered pair of sets of (combinatorial) games. Strictly, this is a recursive definition, but can be well-founded, since we can start with the game $0 = [\varnothing, \varnothing]$, for example, and then form other games.

In game $x = [L, R]$, write x^L for a typical element of L and x^R for a typical member of R. These elements are called **options** of x, with x^L being a **left option**, and x^R a **right option** . Write $x = \{x^L | x^R\}$, and thus write $0 = \{|\}$.

The **positions** in a game x are itself, and all the positions of its options (if any).

Games must be well-founded, so that there is no sequence of options nested infinitely deep. Also the positions form the vertices of a direct acyclic graph, with edges marking the option relationship.

Diversion C.12.1. There also exists a theory of **loopy games** in which the directed graph of positions has cycles.

Two games x and y can be added using the rule:

$$x + y = \{x^L + y, x + y^L | x^R + y, x + y^R\}. \qquad (C.12.1)$$

Diversion C.12.2. Intuitively, the game $x + y$ represents the player playing both games x and y simultaneously, but with the rule that the player can only play one option from the two given games.

Diversion C.12.3. There are other ways to add games. (See Conway's book *On Numbers and Games*, and earlier papers such as that of C. A. B. Smith.)

Addition of games is commutative and associative, both of which can be proven by induction (using well-foundedness of games).

The game $0 = \{|\}$ is the additive identity of addition, which can be proven by induction.

A game can be be represented in a computer by a tree, but this is generally inefficient. For example, it can require an number of nodes that is exponential in the number of moves remaining the in game.

Instead, it is often more efficient to represent a game by a directed acyclic graph (with multiple edges allowed) and edges painted blue (for left) and red (for right). The binary tree representation can be viewed as a special case of this representation. But a directed acyclic graph representation would allow all the leaves of the tree to be represented by a single vertex, a sink. For many games, the graph representation requires less storage than the binary tree representation, and permits more efficient addition.

This more efficient representation has the downside of not being immediately monographic. (Maybe their is an efficient monography algorithm?)

There is a general theory for determining the outcome (the winner) of various types of combinatorial games, under the rule of last play to move wins. Under this outcome rule, many different games become equivalent. This equivalence is a congruence in the semigroup of games. Therefore, equivalence classes of games form another semigroup. The equivalence classes can be quite large. In fact, all games in which the first players are equivalent to the game $0 = \{|\}$.

The opposite rule, first player unable to move wins, is much more complicated. The equivalence classes are much smaller.

The left and right graphs of a sum of game are the Cartesian products the left and right graphs of the games.

So, if the graphs are labelled, then division (game subtraction) should be easy.

Some games have an infinite number of positions, and thus they cannot be represented on a computer by the methods above. Nonetheless, a rich theory on the equivalence classes of games includes some interesting semigroups.

- The **ordinal** numbers, under various operations:
  - Cantor addition and multiplication (which pre-dates combinatorial game theory and considers ordinal as isomorphism classes of well-ordered sets),
  - Normal addition and multiplication (which is a special case of surreal numbers defined below),
  - Nimber addition and multiplication (which extends the Sprague–Grundy theorem),
- The **surreal** numbers, explained in Conway's book *On Numbers and Games*, for example. These form semigroups under addition and multiplication.

Although is impossible to represent all transfinite ordinals on a computer, since they are uncountable in cardinality, certain countable collections of these objects can be represented with a finite amount of information.

For example, the ordinal numbers up to $\omega^{\omega^{\omega^{\cdot^{\cdot^{\cdot}}}}}$ can be represented in Cantor normal form, which in turn can be represented to a binary tree.

C.13 Semigroups from topology?

To be completed.

Oriented knots can be joined by the so-called connected sum. This forms a semigroup on the knot isomorphism classes.

C.14 Semirings with given addition or multiplication

This section considers some constructions of a semiring R such that either R^+ or $R^\times$ is isomorphic to a given semigroup S.

Diversion C.14.1. The constructions in the section require rather modest conditions on the given semigroup, but choose the remaining semigroup operation specifically, simplisticly, and arguably artificially.

Equivalently, the constructions in this section show that specific, special (simplistic) semigroups (null and left semigroups) are distributive over, or under, almost any other semigroup.

Diversion C.14.2. The semiring constructions in this section are not directly useful as methods to directly construct new semigroups, because they build semirings from semigroups. It is possible, though, the the semirings built could then, in turn, to build yet other semigroups.

As stated elsewhere, many of the most natural semigroups are most easily constructed as the additive or multiplicative semigroup of a semiring. In most case, these semirings are not the constructions in this section.

C.14.1 Left addition, any multiplication

Lemma C.14.1. *Let S be a multiplicative semigroup. Define an addition on S as left addition: $a + b = a$ for all $a, b \in S$. Then S is a semiring.*

Proof. Associativity multiplication is because S is a semigroup, Additions is associative, since every sum equals the leftmost term. Left and right distribution follow by the calculations

$$a(b + c) = ab = ab + ac,$$
$$(a + b)c = ac = ac + bc.$$

□

Diversion C.14.3. This semiring has non-commutative addition, which is atypical for a natural semiring. It is well-known that rather mild condition on a semiring imply commutative addition.

Diversion C.14.4. A similar semiring can be constructed using right addition.

C.14.2 Left multiplication, any idempotent addition

Lemma C.14.2. *Suppose that S^+ is an additive semigroup, and that S^+ is idempotent (meaning $s + s = s$ for all $s \in S$.) Define multiplication in S as left multiplication: $ab = a$ for all a, b. Then S is a semiring.*

Proof. Associativity of addition is S^+ is a given semigroup. Multiplication is associative, because every product equals the leftmost factor. Left and right distribution follow by the calculations,

$$a(b + c) = a = a + a = ab + ac,$$
$$(a + b)c = a + b = ac + bc.$$

□

Diversion C.14.5. Conversely, if a semiring R has left multiplication, then it is the result of the construction above. So, addition is idempotent, because $a+a = (ab)+(ac) = a(b + c) = a$.

Diversion C.14.6. Some idempotent semigroups are not commutative, and the resulting semirings have non-commutative addition.

Diversion C.14.7. The semiring in which $a + b = ab = a$, both operations being the left operator, is a semiring with both operation (addition and multiplication) non-commutative if the underlying set has at least two elements.

C.14.3 Null addition, any multiplication with a zero

Lemma C.14.3. *Suppose that $S^\times$ is a multiplicative semigroup with an absorbing element 0. Define addition on S as null addition by the rule $a+b = 0$ for all a, b. Then S is a semiring.*

Proof. Multiplication in S are associative, because S is a (multiplicative) semigroup. Addition is associative (since it is a null semigroup, all sums equalling zero). To establish distribution, notice that $a(b + c) = a0 = 0 = (ab) + (ac)$, and similarly for right distribution. □

Diversion C.14.8. Conversely, if a semiring R has null addition, then it arises from the construction above. Write ∞ for additively absorbing element ∞. Then $a\infty = a(b + c) = (ab) + (ac) = \infty$, so ∞ is multiplicatively absorbing. Then we can re-write $0 = \infty$.

C.14.4 Null multiplication, any addition with an idempotent

Lemma C.14.4. *Let S^+ be an additive semigroup with at least one idempotent element o (meaning $o + o = o$). Define multiplication on S as null multiplication by the rule $ab = o$ for all a, b. Then S is a semiring.*

Proof. Addition is associative because S is a semigroup. Multiplication is associative because all products equal o. Distribution holds because $a(b+c) = o = o + o = (ab) + (ac)$, and similarly for right distribution. □

Diversion C.14.9. Conversely, if a semiring R has null multiplication, then it arises from the construction above. Let $o = ab$ for any $a, b \in R$. Then $o+o = (ab)+(ab) = a(b+b) = o$, so o is idempotent.

C.15 Semirings from sets

This section considers some semirings that can be derived from an arbitrary set.

C.15.1 Boolean semiring

Let S be any set. The boolean semiring $R = B(S)$ consists of all subsets of B. Addition is the union. Multiplication is the intersection.

Diversion C.15.1. There is another definition of boolean algebras in which addition is defined differently, as the **symmetric difference** of sets. So $A+B$ means $(A-B)cup(A-B)$.

This Boolean algebra forms a ring, in which subtraction is possible (and the same as addition), with addition forming an abelian group.

This semiring $B(S)$ has a zero and a one, as follows. The zero is the empty set, and the one is S.

Both addition and multiplication in $B(S)$ are idempotent: $s+s = s \cup s = s$ and $ss = s \cap s = s$.

The semiring $B(S)$ can be given an additional structure from a complement operation, a unary operation. This is defined by $a^c = \{s \in S : s \notin a\}$. The complement satisfies $(a^c)^c = a$. It also satisfies $a^c \neq a$.

The complement operation in $B(S)$ essentially swaps the roles of addition and multiplication. So $(a + b)^c = a^c b^c$ and $(ab)^c = a^c + b^c$. This shows that addition in $B(S)$ is distributive over multiplication. In other words, $B(S)$ is bi-distributive.

Diversion C.15.2. Boolean semirings, can also be equipped some extra structure: full infinitary summation and products: arbitrarily many inputs can be added, or multiplied. The order of the inputs does not matter.

If $|S| = 1$, then $B(S)$ has just two elements, 0 and 1. This semiring is fundamental that we call the **basic** or **fundamental** Boolean semiring, or the **bit semiring**, which we write as $\mathbb{B}$. Its operation tables are thus:

$$\begin{array}{c|cc} + & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 1 \end{array}\,, \qquad \begin{array}{c|cc} \times & 0 & 1 \\ \hline 0 & 0 & 0 \\ 1 & 0 & 1 \end{array}\,. \tag{C.15.1}$$

Viewing 0 and 1 is integers, the addition is the maximum (or supremum) operation, while multiplication is the infimum (or minimum) operation.

Diversion C.15.3. The general Boolean semiring $B(S)$ is the actually the Cartesian power of the bit semiring $\mathbb{B}$ with the set S, which we write as $\mathbb{B}^S$. Recall that the Cartesian power is the set all functions $f : S \to \mathbb{B}$, with the usual point-wise addition and multiplication of functions, so $(fg)(x) = (f(x))(g(x))$.

C.15.2 Semiring of relations

Let X be an arbitrary set. We can form a semiring $R = \mathrm{Rel}(X)$ of relations on S. A relation is simply of a subset of $X^2 = \{[x, y] : x, y \in S\}$, consisting of all ordered pairs with element in X (the Cartesian product of X with itself).

Diversion C.15.4. The relation semiring $\mathrm{Rel}(X)$ and the Boolean semiring $B(X^2)$ share the same underlying set, and the same addition operation. The multiplication operations are different.

Addition in $\mathrm{Rel}(X)$ is the union operation:

$$r + s = \{[x, y] : [x, y] \in r, s\}. \tag{C.15.2}$$

Addition, being an instance of the union operation, is associative $r+(s+t) = (r + s) + t$, commutative $(r + s = s + r)$ and idempotent $(r + r = r)$.

Multiplication in $\mathrm{Rel}(X)$ is **(relational) composition**, defined as follows:

$$rs = \{[x, z] : \exists z, [x, z] \in r, [z, y] \in s\}. \tag{C.15.3}$$

Multiplication of relations is associative, but generally non-commutative.

Multiplication is both left and right distributive over additions: $r(s+t) = (rs) + (rt)$ and $(r + s)t = rt + st$, so $\mathrm{Rel}(X)$ is a **semiring**.

Notationally, we can following traditional practices in algebra, placing higher precedence on multiplication, writing $rs + rt$ instead of $(rs) + (rt)$, thus saving on parentheses.

C.15.2.1 Further structure on relations

Two special elements, of the semiring of relations can be defined.

Diversion C.15.5. A special designated elements can also be considered as a nullary operation, which makes for a more consistent description in universal algebra of sets being equipped with multiple operators, some binary (such as multiplication), some unary (such as transpose, see further below), and some nullary (such as identity, defined next).

First there is an element 0, which is the empty set. It satisfies the usual properties of a zero in a ring:

$$0 + r = r + 0 = 0, \tag{C.15.4}$$

$$0r = r0 = 0. \tag{C.15.5}$$

In other words, 0 is both an additive identity element, and a multiplicatively absorbing element. In $B(X^2)$, the empty set is also the semiring 0.

Similarly, one can define an element 1, as follows:

$$1 = \{[x, x] : x \in S\}. \tag{C.15.6}$$

So, 1 is the identity relation, or diagonal relation. Element 1 is a multiplicative identity:

$$1r = r1 = r. \tag{C.15.7}$$

Thinking of $\mathrm{Rel}(X)$ as Boolean matrices, notice that 1 has Boolean ones on its main diagonal, and Boolean zeros off its main diagonal, just like the identity matrix I of linear algebra.

The semiring 1 of $\mathrm{Rel}(X)$ is not the semiring 1 of $B(X^2)$.

Two unary operations on relations can be considered. These enhance the semiring structure, resulting in a more intricate structure.

The first unary operation is the **transpose**, or **reversal** operation, and is defined as follows:

$$r^t = \{[y, x] : [x, y] \in r\} \tag{C.15.8}$$

The transpose operation is an involution:

$$(r^t)^t = r. \tag{C.15.9}$$

So, the transpose is its own inverse, and the transpose is bijective.

Transpose interacts with addition and multiplication as follows:

$$(r+s)^t = s^t + r^t, \tag{C.15.10}$$
$$(rs)^t = s^t r^t. \tag{C.15.11}$$

Because of reversal of the order of r and s on the right hand side of these equations, the transpose operation provides be considered as anti-automorphism.

The second unary operation is the complement, defined:

$$r^c = \{[x,y] : x,y \in X; \quad [x,y] \notin r\}. \tag{C.15.12}$$

The complement is also an involution:

$$(r^c)^c = r, \tag{C.15.13}$$

so it is bijective (on relations).

The complement commutes with the transpose operation in the sense that $(r^c)^t = (r^t)^c$. We can safely write r^{ct} or r^{tc}, for the complement of the transpose.

The complement interacts with the addition operation just the same that the union operation interacts with complement. In particular, the complement can be used to recover the boolean semiring structure $B(X^2)$, because the intersection operation is the complement of the union of complements.

The complement interact with multiplication (composition) in $\mathrm{Rel}(X)$ quite differently than it interacts with multiplication (intersection) in $B(X^2)$. For example, $0^c \neq 1$, in $\mathrm{Rel}(X)$ (whereas $0^c = 1$ in $B(x^2)$).

In $\mathrm{Rel}(X)$, the element 0^c is an absorbing element for addition, (just as it is $B(S^2)$), and is almost an absorbing element for multiplication. (But there can be at most one absorbing element for multiplication, and this is 0. Nonetheless, for many elements r, we will have $0^c r = 0^c = 0^c r$.)

C.15.2.2 Relation types using semiring equations

It is common to define certain types of relations, such as **reflexive**, **transitive**, **symmetric**, and several other. Many of these types of relations can be define using semiring notation for $\mathrm{Rel}(X)$. A relation r is:

- **reflexive** if $1 + r = r$,
- **transitive** if $r + rr = r$,
- **symmetric** if $r^t = r$,
- an **equivalence** if reflexive, transitive and symmetric,
- **anti-symmetric** if $r^c + r^{ct} = 1^c$,
- a **partial ordering** if reflexive, transitive and anti-symmetric
- **total** if $r + r^c = 0^c$,
- a **total ordering** if a partial ordering and total,
- **left-surjective** if $1 + rr^t = rr^t$, (or equivalently, if $r0^c = 0^c$),
- **left-injective** if $1 + r^t r = 1$,
- a **function** (**graph**) if left-surjective and left-injective,
- a **permutation** if $rr^t = 1$, or equivalently, if $rr^t = 1$, or equivalently, if both r and r^t are functions.

To be completed.

Characterize well-orderings.

Apply these semirings equation to more usual applications of special types of relations.

C.15.2.3 Relations as bit-valued matrices

Consider the full matrix semiring $M_S(\mathbb{B})$ whose index set is S, and whose base ring is the bit semiring $\mathbb{B}$, where we place no restrictions on the row or column contents.

Matrix multiplication is well-defined because $\mathbb{B}$ allows infinitary sums. Matrix addition and multiplication obey the semiring axioms, for similar reasons.

The semiring $\mathrm{Rel}(S)$ is isomorphic to the semiring $M_S(\mathbb{B})$. A relation r correspond to the matrix with entry 1 in each position $(x, y) \in r$, and 0 in all other matrix positions.

C.15.2.4 Alternative notation for complement and transpose

When dealing the many variable, using letter notation for complement and transpose might lead to too much clutter. One reasonable alternative notation is to write $\bar{a}$ or a^- for the complement, and d' for the transpose.

C.16 A free semiring with one generator???

This section attempts to begin to describe a practical representation of a semiring S, with non-commutative addition, that is **free** over one generator.

As usual, if g is the generator, being free over g, means that for any other semiring T, and any element $h \in T$, there exists exactly one semiring morphism $m : S \to T$ such that $m(g) = h$.

We try represent each element of S by a positive integer representatives. That is to say, each element of S is an equivalence class of positive integers. As done elsewhere in this report, we indicate an equivalence class by surrounding the positive integer in square brackets.

Many of the equivalence classes for small integers have only one member, but for large positive integers $m \neq n$, it sometimes will occur that $[m] = [n]$. This report does not yet offer a practical algorithm to determine if $[m] = [n]$ when $m \neq n$. So, it this description of S is not yet fully practical.

Addition (of representatives) is defined by the rule:

$$[a] + [b] = \left[(2a-1)2^{\lfloor \log_2 b \rfloor} + b\right]. \tag{C.16.1}$$

This operation is non-commutative and associative, even if the equivalence classes are defined to have one element each. We will arrange that addition is well-defined on equivalence classes.

Diversion C.16.1. The general size of $[c] = [a] + [b]$ can be approximated as $c \approx 2ab$.

Diversion C.16.2. The arithmetic for addition can be translated into a simpler rule on bit strings. Write a and b in standard binary notation. Drop the leading one-valued bits, getting two bit strings α and β (possibly empty). Form the bit string $\alpha 0 \beta$ by concatenation (with a 0-bit inserted). Then convert back to an integer by pre-pending a 1-bit.

Every positive integer has a decomposition into a sum of Mersenne integers, like this:

$$[n] = [2^{n_1} - 1] + \cdots + [2^{n_t} - 1]. \tag{C.16.2}$$

For example, $[45] = [1]+[7]+[3]$. Now we define a multiplication of Mersenne numbers, via the rule:

$$[M][N] = [MN + M + N] \tag{C.16.3}$$

which only applies if $M = M_m = 2^m - 1$ and $N = M_n = 2^n - 1$ for some positive integers m and n. In this case, $[M][N] = [M_m][M_n] = [M_{m+n}] = [2^{m+n} - 1]$.

General multiplication is now conducted by using decomposition together with the distributive law. For example, using left distribution first:

$$\begin{aligned}
[45][2] &= ([1] + [7] + [3])([1] + [1]) \\
&= (([1] + [7] + [3])[1]) + (([1] + [7] + [3])[1]) \\
&= ([1][1] + [7][1] + [3][1]) + ([1][1] + [7][1] + [3][1]) \\
&= ([M_1][M_1] + [M_3][M_1] + [M_2][M_1]) + \dots \\
&= ([M_2] + [M_4] + [M_3]) + ([M_2] + [M_4] + [M_3]) \\
&= \dots \\
&= [454075]
\end{aligned} \tag{C.16.4}$$

But we also have $[45][2] = [220635]$ by using right distribution first, as in:

$$\begin{aligned}
[45][2] &= ([1] + [7] + [3])([1] + [1]) \\
&= ([1]([1] + [1])) + ([7]([1] + [1])) + ([3]([1] + [1])) \\
&= [M_2] + [M_2] + [M_4] + [M_4] + [M_3] + [M_3] \\
&= \dots \\
&= [220635].
\end{aligned} \tag{C.16.5}$$

This example allows us to deduce that $[220635] = [454075]$.

C.16.1 Operation tables

This subsection gives some examples of arithmetic in the semiring. Representations $[n]$ are not necessarily unique.

The tens' table is given by:

+	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
[1]	[2]	[4]	[5]	[8]	[9]	[10]	[11]	[16]	[17]	[18]
[2]	[4]	[8]	[9]	[16]	[17]	[18]	[19]	[32]	[33]	[34]
[3]	[6]	[12]	[13]	[24]	[25]	[26]	[27]	[48]	[49]	[50]
[4]	[8]	[16]	[17]	[32]	[33]	[34]	[35]	[64]	[65]	[66]
[5]	[10]	[20]	[21]	[40]	[41]	[42]	[43]	[80]	[81]	[82]
[6]	[12]	[24]	[25]	[48]	[49]	[50]	[51]	[96]	[97]	[98]
[7]	[14]	[28]	[29]	[56]	[57]	[58]	[59]	[112]	[113]	[114]
[8]	[16]	[32]	[33]	[64]	[65]	[66]	[67]	[128]	[129]	[130]
[9]	[18]	[36]	[37]	[72]	[73]	[74]	[75]	[144]	[145]	[146]
[10]	[20]	[40]	[41]	[80]	[81]	[82]	[83]	[160]	[161]	[162]

A multiplication table was generated as

×	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
[1]	[3]	[13]	[7]	[53]	[27]	[29]	[15]	[213]	[107]	[109]
[2]	[13]	[213]	[59]	[3413]	[875]	[941]	[247]	[54613]	[13739]	[13997]
[3]	[7]	[59]	[15]	[475]	[119]	[123]	[31]	[3803]	[951]	[955]
[4]	[53]	[3413]	[475]	[218453]	[28011]	[30125]	[3959]	[13981013]	[1758635]	[1791661]
[5]	[27]	[859]	[119]	[27355]	[3511]	[3771]	[495]	[874203]	[110007]	[112059]
[6]	[29]	[949]	[123]	[30421]	[3819]	[3949]	[503]	[973653]	[121771]	[122285]
[7]	[15]	[247]	[31]	[3959]	[495]	[503]	[63]	[63351]	[7919]	[7927]
[8]	[213]	[54613]	[3803]	[13981013]	[896363]	[964013]	[63351]	[3579139413]	[225105323]	[229332653]
[9]	[107]	[13659]	[951]	[1747675]	[112055]	[120507]	[7919]	[223696603]	[14069175]	[14333371]
[10]	[109]	[13749]	[955]	[1750741]	[112363]	[120685]	[7927]	[223796053]	[14080939]	[14343597]

But this table is a little misleading because it suggests that $[2][5] \neq [5][2]$, which is false.

Diversion C.16.3. Actually, I think that this semiring has commutative multiplication.

J code for all these computation:

```
plus=:]+(((((*<:)+:)~2x&^)<.)2&^.)
plus"0 table 1+i.10
intify=:(#;.1)@:(0&,)@:}.@:#:
mult=:[:plus/_1+2x^,@(+/)&intify
mult"0 table 1+i.10
```

This code makes no effort to determine if two positive integers are equivalent. Essentially, it simply finds one representation for the output of an operation given representations of the input operations.

C.16.2 Ineffectiveness of representation

The discussion above provides no efficient rule to determine if $[m] = [n]$ for arbitrary integers, though perhaps the reader might devise such a rule. Consequently, the additive and multiplicative semigroups involved in this semiring lack, as yet, the monography property: finding a unique a representation is not yet described (in this report). So these semigroup are not (yet) useful for associative key agreement.

C.16.3 Representation as quasi-polynomials

If we let $M = [1]$, then we see that every element of the semiring can be written in the form $M^a + M^b + \cdots + M^d$ for one or more positive integers $a, b, c, \ldots$, but that these representations are not unique. In other words, these are quasi-polynomials in one variable, where by quasi-polynomial, we mean non-commutative addition.

As an example of some non-uniqueness, it is fairly easy to show that that if $a + d = b + c$ and $b, c \geq 3$, then, it generally holds that:

$$M^a + M^b + M^c + M^d = M^a + M^c + M^b + M^d, \tag{C.16.6}$$

which is a special case of internal commutation (of addition). To see this, let $(e, f, g, h) = \frac{1}{2}(a + b - i, c + d - i, c - d + i, b - a + i)$ for some (perhaps negative) integer i such that all four integer sums with i are even positive integers. (To do: show i exists. I think we can take $i = 2$ or $i = b - a + 2$ or $i = b - a + 3$.) Then expand $(M^e + M^f)(M^g + M^h)$ twice, first by left distribution, and second by right distribution, which gives both sides of the desired equation. For example the first term in both expansions is $M^{e+g} = M^{((a+b-i)+(c-d+i))/2} = M^{(2a+(b+c)-(a+d))/2} = M^a$.

C.16.4 Weighted lexicographic representation

Finite, (non-empty) sequences of positive integers, can be ordered by total weight, and then lexicographically, like this:

$$\begin{gathered} 1 \\ < 11 < 2 \\ < 111 < 12 < 21 < 3 \\ < 1111 < 112 < 121 < 13 < 211 < 22 < 31 < 4 \\ < 11111 < 1112 < \dots \end{gathered}$$

Diversion C.16.4. For brevity, the sequences are shown as concatenation of digits, without any separators, but when greater clarity to handle larger integers is needed, the sequences should be enclosed in the parentheses, with commas separating sequence elements, so 11 above, should be $(1, 1)$.

Enumerating sequences gives a correspondence:

$$\begin{aligned} 1 &\mapsto 1 \\ 11 &\mapsto 2 \\ 2 &\mapsto 3 \\ 111 &\mapsto 4 \\ 12 &\mapsto 5 \\ &\dots \end{aligned}$$

Diversion C.16.5. A simple procedure can convert between these sequences and the binary representation of indices. For example, consider sequence 12, or $(1, 2)$ in full. Subtract one from each entry, giving $(0, 1)$. Replace each entries by an all-ones bit string of that length, giving $(, 1)$ (whose first entry is the empty bit string). Next replace each comma (sequence separator) with a single bit of value 0, giving 01. Finally, prepend a leading one-valued bit, to get 101, which is five.

A positive integer n represented by a sequence $(n_1, \dots, n_t)$ obeys a simple relationship in the semiring:

$$[n] = [1]^{n_1} + [1]^{n_2} + \cdots + [1]^{n_t}. \tag{C.16.7}$$

C.16.5 Polynomial images

Every semiring element $[n] \in Q$ (for a positive integer n) can be represented a sum of powers of $M = [1]$. We can thus define a semiring morphism

$h : Q \to \mathbb{P}[x]$ that sends M to x, where, as usual $\mathbb{P}[x]$ means the semiring of polynomials with positive integer coefficients. Now $\mathbb{P}[x]$ has commutative addition.

It follows that each equivalence class of Q maps to a single polynomial. So, the only question in determining an equivalence is which ordering of the individual powers are equivalent.

C.17 Semirings of functions on a semigroup?

In this section, we build some nearrings and semirings from an arbitrary semigroup S.

Diversion C.17.1. As usual, call S the **base** semigroup when necessary to distinguish S from the additive and multiplicative semigroups of the nearring or semiring under construction.

For notational convenience, we will assume that base semigroup, so write it as S^+.

Diversion C.17.2. Additive notation here is not meant to imply commutativity: we allow S^+ to be non-commutative.

C.17.1 Nearring of semigroup functions

Let $F = S^S$ be the set of functions from S to itself. We will make F into a nearring by defining as addition and multiplication, as follows.

$$\sigma + \tau : s \mapsto \sigma(s) + \tau(s), \tag{C.17.1}$$

$$\sigma\tau : s \mapsto \sigma(\tau(s)), \tag{C.17.2}$$

for any $\sigma, \tau \in F$. In other words,

- addition in F is point-wise addition, making F^+ the Cartesian power semigroup $(S^+)^S$ (see §C.9.2.2).
- multiplication in F is function composition, making $F^\times$ a subsemigroup of the function composition semigroup S^S (see §C.5.5).

Since F^+ and $F^\times$ are semigroups, the associativity of addition and multiplication is established. For F to be a nearring, we also need at least one distributive law.

Lemma C.17.1. *Set $F = S^S$, with addition and multiplication defined above, is a right nearring.*

Proof. Verifying right distribution is rather immediate:

$$\begin{aligned}((\sigma + \tau)\upsilon)(s) &= (\sigma + \tau)(\upsilon(s)) \\ &= \sigma(\upsilon(s)) + \tau(\upsilon(s)) \\ &= (\sigma\upsilon + \tau\upsilon)(s).\end{aligned}$$

□

C.17.2 Semiring of endomorphisms

Let $E = \mathrm{End}(S^+) = \mathrm{Mor}(S^+, S^+)$ be the set of **endomorphisms**, which is the set of functions $\sigma : S \to S$, such that $\sigma(s + t) = \sigma(s) + \sigma(t)$. Clearly, E is a subset of the nearring F from the previous section.

The semiring S^+ is said to be **medial** if $a + b + c + d = a + c + b + d$ for all $a, b, c, d \in S$.

Lemma C.17.2. *If S is medial, then the $E = \mathrm{End}(S)$ is a subsemiring of nearring F.*

Proof. We must first show that E is closed under both addition and multiplication. First we show that $\sigma + \tau \in E$ if $\sigma, \tau \in E$:

$$\begin{aligned}(\sigma + \tau)(s + t) &= \sigma(s + t) + \tau(s + t) \\ &= \sigma(s) + \sigma(t) + \tau(s) + \tau(t) \\ &= \sigma(s) + \sigma(t) + \tau(s) + \tau(t) \\ &= \sigma(s) + \tau(s) + \sigma(t) + \tau(t) \\ &= (\sigma + \tau)(s) + (\sigma + \tau)(t).\end{aligned}$$

This argument required S^+ to be medial.

Closure under multiplication is simpler, and does not require S^+ to be medial. Expressly, $(\sigma\tau)(s + t) = \sigma(\tau(s + t)) = \sigma(\tau(s) + \tau(t)) = \sigma(\tau(s)) + \sigma(\tau(t)) = (\sigma\tau)(s) + (\sigma\tau)(t)$.

Right distributivity and associativity of addition and multiplication are inherited in E from F.

To verify the left distributive law in E:

$$\sigma(\tau + \upsilon) = \sigma\tau + \sigma\upsilon,$$

by evaluating at the function at arbitrary element of S^+:

$$\begin{aligned}(\sigma(\tau+\upsilon))(s) &= \sigma((\tau+\upsilon)(s))\\ &= \sigma(\tau(s)+\upsilon(s))\\ &= \sigma(\tau(s))+\sigma(\upsilon(s))\\ &= (\sigma\tau)(s)+(\sigma\upsilon)(s)\\ &= (\sigma\tau+\sigma\upsilon)(s).\end{aligned}$$

This holds for all s, so the functions are equal. The only step above relying any of σ, τ, υ being the endomorphism was the step $\sigma(\tau(s)+\upsilon(s)) = \sigma(\tau(s))+\sigma(\upsilon(s))$, which uses only the fact that σ is an endomorphism. □

Diversion C.17.3. More generally, replace the condition that S^+ is medial by the condition that it has an medial T^+ subsemigroup. Then, let E_T be the subset of E consisting of functions σ such that whose image $\sigma(S) \supseteq T$.

Then E_T is a semiring.

When S^+ is medial, we can put $T = S$, and get back E as $E = E_S$.

Diversion C.17.4. More generally, suppose that S^+ is an arbitrary additive-notation semigroup, not necessarily medial.

Then E is closed under multiplication. Let $\sum E$ denote the set of sums of all elements of E. Then $\sum E$ is closed under addition, and also multiplication, so it forms the subnearring of F. It is actually the nearring closure $\langle E\rangle$ of E.

The functions in $\langle E\rangle$ might deserve (and already have) a name (the term **summorphism** would somewhat descriptive but quite awkward).

Nearring $\sum E = \langle E\rangle$ is not usually a semiring, because it usually lacks left distributivity.

C.18 Semilattice semirings??

If R is a semiring in which in addition is commutative and idempotent, then R can be called a **semilattice semiring** because R^+ is a semilattice: as noted in §C.3.1. The fact that additive semigroup S^+ is commutative and idempotent means that it defines a semilattice, a partial ordering, defined by:

$$a \le b \iff a = a + b. \tag{C.18.1}$$

It is a semilattice in the sense that $a + b = \inf\{a, b\}$.

The partial ordering $\le$ (the semilattice) is consistent with multiplication in the semiring S in the sense the multiplicative by a constant is monotonic, as shown next.

Lemma C.18.1. *If S a left nearring with S^+ idempotent and commutative, then $b \leq c$ implies $ab \leq ac$ for all $a, b, c \in S$.*

Proof. If $b \leq c$, then $b = b + c$. Multiply both sides on the left by a to get $ab = a(b + c)$. By left distributivity, $a(b + c) = ab + ac$. So, $ab = ab + ac$, which means $ab \leq ac$. □

The converse holds any semilattice $\leq$ on a multiplicative semigroup S, such that left multiplication in monotonic, defines a left nearring, by similar arguments.

Diversion C.18.1. Not every partially ordered semigroup forms a semilattice semiring. If the partial order lacks pairwise infimums (and pairwise supremums), then it is does not define a semilattice.

To be completed.

Diversion C.18.2. The **tropical algebras** are semilattice semiring.

Diversion C.18.3. Multiplication in a semilattice semiring can be non-commutative.

For example, let S be the free semigroup of words over the alphabet $\{0, 1\}$, where multiplication in S is word concatenation. This is non-commutative: $(00)1 \neq 1(00)$.

Define an ordering $\leq$ on S, by embedding S into set of $\mathbb{P}$ the positive integer by prepending a 1 to the word, and then interpreting the prepended string as the binary representation of a positive integer. Equivalently, the ordering can be defined by comparing a word length first, and comparing words lexicographically.

Left and right multiplication in S are each monotonic with respect to this ordering. To see this, notice that multiplication in S is additive over word length, so preserves all comparison based on word length. For tied word lengths, comparison is done by lexicographically, but the common factor does not disrupt the ordering of the shorter words, whether it is a prefix or suffix.

Indeed, this semilattice semiring is totally ordered. So, totally ordered semigroups can be non-commutative.

Yet another description of this semigroup S is to represent each element as $[a]$ for some positive integer, and define multiplication by concatenating binary representation after removal the leading bit, and then replacing the leading bit.

In the J programming language, multiplication in S, under this representation, may be implemented by the verb `m=:(1(#.@,),"1&}.&#:)"0`.

C.18.1 Lattice semirings

A partial ordered set P is a lattice if every pair $a, b \in P$ of elements has a unique greatest lower bound $a + b$ and a unique least upper bound ab.

For example, if P is totally ordered, then it is a lattice. The set of subsets of a set, ordered by inclusion is a lattice.

C.18.2 Total orders

Let T be a total order. Let Z be a subset of T. Define a semiring from (T, Z) as follows. Let the underlying set be T.

Let $ab = \max(a, b)$ for all a, b. Let

$$a + b = \begin{cases} \min(a, b) & \text{if } \max(a, b) \in Z \\ \max(a, b) & \text{if } \max(a, b) \notin Z \end{cases} \tag{C.18.2}$$

Clearly, addition and multiplication are commutative.

Diversion C.18.4. This type of semiring generalizes the notion of adding zeros and infinities to an existing semiring, as described in §**??**.

To be completed.

C.18.3 Incidence algebras?

An **incidence algebra** T is a semiring that combines a semiring R with locally finite partially ordered set P, consisting of functions from nonempty closed intervals of P to R.

To be completed.

C.19 The standard semiring of positive integers

Perhaps the most fundamental and natural semiring is the semiring of positive integers $\mathbb{P} = \{1, 2, 3, \dots\}$ under standard addition and multiplication.

Diversion C.19.1. For practicality, we need a monography for positive integers: and there are several different representation systems.

C.19.1 Positive integers under multiplication

Integer multiplication is associative, and positive integers are closed under multiplication. So the positive integers form a semigroup $\mathbb{P}^{\times}$ under standard integer multiplication.

A familiar part of multiplication table is:

×	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

(C.19.1)

Diversion C.19.2. This semigroup is also:

- **unital**, since it has an element 1 such that $1a = a1 = a$, for all a;

- **commutative**: $ab = ba$ for all a, b;

- **cancellative**: $ab = ac$ implies $b = c$, for all a, b, c.

Diversion C.19.3. A unital semigroup is also called a **monoid**.

Algorithms for multiplication are well-known and standard.

See §**??** for division in this semigroup $\mathbb{P}^{\times}$.

C.19.2 Positive integers under addition

The positive integers are closed under standard integer addition, which is associative. This forms a semigroup $\mathbb{P}^{+}$. As usual, we write the semigroup operation with the symbol $+$. This helps avoid confusion with semigroup $\mathbb{P}^{\times}$.

It has partial table:

+	1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10	11
2	3	4	5	6	7	8	9	10	11	12
3	4	5	6	7	8	9	10	11	12	13
4	5	6	7	8	9	10	11	12	13	14
5	6	7	8	9	10	11	12	13	14	15
6	7	8	9	10	11	12	13	14	15	16
7	8	9	10	11	12	13	14	15	16	17
8	9	10	11	12	13	14	15	16	17	18
9	10	11	12	13	14	15	16	17	18	19
10	11	12	13	14	15	16	17	18	19	20

(C.19.2)

Diversion C.19.4. Semigroup $\mathbb{P}^+$ is commutative and cancellative, but it is not a monoid (because here we have excluded non-positive integer 0, which would have been the identity element of the additive operation).

Diversion C.19.5. A simple way to recover multiplicative notation for $\mathbb{P}^+$, is to write the elements as powers of some formal variable, say x. The underlying set is then $\{x^1, x^2, x^3, \ldots\}$. Multiplication looks like $x^a x^b = x^{a+b}$, which is just addition in the exponents.

This set does not include a multiplicative identity 1 (because $st \neq s$ for all s, t). It also does not include a zero element 0 (because $st \neq s$ for all s, t).

It is also possible to embed $\mathbb{P}^+$ inside $\mathbb{P}^\times$, as a subsemigroup (subsemigroups are discussed more generally in §C.6.1), by replacing formal variable x for some positive integer. For example, if $x = 10$, then we represent $\mathbb{P}^+$ by the set $\{10, 100, 1000, \ldots\}$, the powers of ten (excluding one), under multiplication (or, with $x = 2$, a computer-friendlier set $\{2, 4, 8, 16, \ldots\}$).

See §?? for a discussion of division (re-named as subtraction, of course!) in $\mathbb{P}^+$.

C.19.3 Distributing over integer multiplication

This subsection describes some unusual operations on positive integers that are distributive over standard multiplication.

C.19.3.1 Factorialization of an existing positive integer semigroup

Let $*$ be any associative operation on the positive integers. In this section, we derive from $*$ a second associative operation $*_!$, called its **factorializa-**

tion. The derivation uses prime factorizations and indexing, which we briefly review.

It is well-known that there an infinitely many primes. So, index the primes by increasing size with positive integers, as in $p_1 < p_2 < p_3 < p_4 < \dots$. For example, $p_4 = 7$. Efficiently finding p_i from i, or vice versa, is possible for small enough i.

Every positive integer has a prime factorization, unique up to ordering of the factors:

$$a = \prod_i p_i^{a_i}. \tag{C.19.3}$$

Each a_i is a non-negative integer, and at most finitely many of the a_i are positive. All the indices i in p_i are positive.

Now let $*$ be any associative operation defined on $\mathbb{P}$. We define a new associative operation $*_!$, called the factorialization of $*$, as follows.

Definition C.19.1. *The* ***factorialization*** *of associative operation $*$ is the operator defined by*

$$\left(\prod_i p_i^{a_i}\right) *_! \left(\prod_j p_j^{b_j}\right) = \prod_{i,j} p_{i*j}^{a_i b_i}. \tag{C.19.4}$$

We leave it for the reader to verify that $*_!$ is associative.

In addition to being associative, operator $*_!$ is distributive over standard multiplication, which we can write as $\times \nearrow *_!$. In particular, $*_!$ can be used to form a new semiring over the positive integers, which we may as $\mathbb{P}_{\times}^{*!}$.

Diversion C.19.6. The positive integers have a distributive chain of associative binary operations: $\sqcup \nearrow + \nearrow \times \nearrow *_!$. Perhaps future work will find cryptographic application of the curious fact.

C.19.3.2 Factorialized addition

For example, we can take $* = +$, and get factorialized addition, whose ten by ten table looks like:

$+_!$	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
2	1	3	5	9	7	15	11	27	25	21
3	1	5	7	25	11	35	13	125	49	55
4	1	9	25	81	49	225	121	729	625	441
5	1	7	11	49	13	77	17	343	121	91
6	1	15	35	225	77	525	143	3375	1225	1155
7	1	11	13	121	17	143	19	1331	169	187
8	1	27	125	729	343	3375	1331	19683	15625	9261
9	1	25	49	625	121	1225	169	15625	2401	3025
10	1	21	55	441	91	1155	187	9261	3025	1911

(C.19.5)

Of course, implementing $+_!$ is quite inefficient, since it requires factorization and conversions between i and p_i. Perhaps, it becomes efficient with a a quantum computer.

For division in $\mathbb{P}^{+_!}$, see §5.6.3.

C.19.3.3 Recursive factorialization

A unique operator $\perp$ on the positive integers has the property it is its own factorialization: $\perp_!{=}\perp$. Its ten by ten table looks like:

$\perp$	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
2	1	2	2	4	2	4	2	8	4	4
3	1	2	3	4	3	6	7	8	9	6
4	1	4	4	16	4	16	4	64	16	16
5	1	2	3	4	5	6	7	8	9	10
6	1	4	6	16	6	24	14	64	36	24
7	1	2	7	4	7	14	53	8	49	14
8	1	8	8	64	8	64	8	512	64	64
9	1	4	9	16	9	36	49	64	81	36
10	1	4	6	16	10	24	14	64	36	40

(C.19.6)

Diversion C.19.7. The fact that $2^a \perp 2^c = 2^{ac}$ is reminiscent of Diffie–Hellman key agreement, although that is not at all how one would use $\mathbb{P}^{\perp}$ in associative key agreement.

Diversion C.19.8. An implementation of $\perp$, written in the J programming language is:

```
o=:*/@,@($:&.(>:@(_1&p:))"0/&q:)
```

(C.19.7)

This recursive code is excessively slow due to some J programming bug[1].

As with the previous semigroup $\mathbb{P}^{+!}$, the semigroup $\mathbb{P}^{\perp}$ has a more natural representation in which multiplication does not require prime factorization and indexing.

In the alternative representation, we represent integers by rooted free trees. At the root, there is one branch per each prime (counted with multiplicity) in the factorization of the positive integer. For example, 20 correspond to a tree with a root vertex, three vertices at height 1, and then two more vertices at height 2 and 3, which can be seen formulaicly by

$$20 = (2)(2)(5) = p_1 p_1 p_3 = p_1 p_1 p_{p_2} = p_1 p_1 p_{p_{p_1}} = ppp_{p_p}, \tag{C.19.8}$$

where in the final formula, we omit all the indices of 1, and each p corresponds to a non-root vertex of the tree, with levels be directed downwards (instead of upwards in the usual convention), by levels of subscripting.

In this tree-representation, one can multiply recursively, without referring to any primes or number-theory. Given two trees t and u, recursively multiply their branches (if any) in pairs. Then assemble all the results products as the branches of the new tree.

Diversion C.19.9. When some branches repeat, we use a yet more compact representation of the trees. We record each branch once, and record its multiplicity as a positive integer. The multiplicities correspond to the exponents in the prime-power factorization.

This representation requires caution to make sure that all the branches represent genuinely distinct free trees, which can be non-obvious. In this compact representation, we can sometimes multiply more quickly, by apply recursion only to the distinct branches, and then using standard integer multiplication to find resulting multiplicities. However, we must then take an extra step of identifying because some of the products of the branches may end up the same. For these repeats we should gather the matching branches together by adding up their multiplicities. (In some contexts, we can tolerate ambiguous representation of trees.)

The compact notation can also be written by a system of subscripting. The empty string represents 1. Otherwise, write the multiplicities (as standard integers), but write the index of the prime as subscript, represented recursively. In decimal, end multi-digit integers with a period, to avoid confusion, with the left digits being considered as multiplicities. For example, consider the standard notation integer $23^5 - 2$ (the b value from

[1] Possibly using verbs of too low rank

the highest quality known *abc* triple). In the notation just described, its compact representation can be given as:

$$10._1 1_{1_{11_{1_1}}} \tag{C.19.9}$$

because, in standard notation, $23^5 - 2 = 3^{10}109 = p_2^{10}p_{29}^1 = p_{p_1^1}^{10}p_{p_{10}^1}^1 = p_{p_1^1}^{10}p_{p^1_{p_1^1 p_5^1}}^1 = \ldots$ and so on. But the tree 23^5 has its compact notation in the form

$$5_{2_1} \tag{C.19.10}$$

because $23^5 = p_9^5 = p_{p_2^2}^5 = p^5_{p^2_{p_1^1}}$. When convenient, some notation the single-vertex tree, which corresponds to 1 in $\mathbb{P}^\perp$, can be used, namely 0 which helps in doing some recursions. For example $1 = 1_0$, so $1 \perp 1_1 = 1_0 \perp 1_1 = 1_{0\perp 1} = 1_0 = 1$. So, now the general tree multiplication pattern begins:

$$a_ib_j\ldots c_k \perp d_le_m\ldots f_n = (ad)_{i\perp l}(ae)_{i\perp m}\ldots(cf)_{k\perp n}. \tag{C.19.11}$$

But, if we want a monography (canonical representation), we must identify isomorphic branches, indicated in the subscripts. If all the subscripts were distinct, except the first and last, then we could get a final answer of:

$$a_ib_j\ldots c_k \perp d_le_m\ldots f_n = (ae)_{i\perp m}\ldots(ad+cf)_{k\perp n}. \tag{C.19.12}$$

Now let us compute $(23^5-2)\perp(23^5)$, in the free tree representation, using the compact notation.

$$\begin{aligned}
10._1 1_{1_{11_{1_1}}} \perp 5_{2_1} &= 50._{1\perp 2_1}5_{1_{11_{1_1}}\perp 2_1} \\
&= 50._{2_{0\perp 1}}5_{2_{11_{1_1}\perp 1}} \\
&= 50._{2_0}5_{2_{11_{1_1}\perp 0}} \\
&= 50._2 5_{2_{11}} \\
&= 50._2 5_{2_2}
\end{aligned} \tag{C.19.13}$$

which, as promised, uses standard integer multiplication, and recursion, but does require prime factorization or indexing. The last step involved replacing 2_{11} by 2_2, by gather common isomorphic free trees (i.e. primes in $\mathbb{P}^\perp$). Converting back to positive integers (where conversion require prime indexing, but no factorization), we get answer $5^{50}227^5$.

See §**??**, for division in $\mathbb{P}^\perp$.

C.20 Semirings from semirings

C.20.1 Subsemirings

Let S be a semiring. A subset $T \subseteq S$ is a **subsemiring** if T is closed under addition and multiplication. (Equivalently: T^+ and $T^\times$ are subsemigroups of S^+ and $S^\times$.)

Diversion C.20.1. The intersection of any family of subsemirings is a subsemiring (making the set of subsemirings into a complete (lower) semilattice).

If some property P of semirings has the property that if it holds a semiring S, then it holds for all subsemirings of T of S, then we say that P is an **inheritable** property. For any inheritable property P, a semiring S has a unique smallest subsemiring T with property P. (To be useful, the inheritable property P should exclude very small semirings, such as the empty semiring, otherwise the smallest T may be too small to be useful.)

C.20.1.1 Subsemirings generated by a subset

Let G be any subset of S. There exists a smallest subsemiring $\langle G\rangle$ that contains G. So, $\langle G\rangle = \{g_1 \circ_2 g_2 \cdots \circ_n g_n : n \in \mathbb{P}, g_i \in G, \circ_i \in \{+, \times\}\}$. This is subsemiring generated by G.

The notation $\langle G\rangle_S$ can be used if there is any confusion about which semiring is being used.

The notation $\langle G\rangle_+$ can be used to to indicate the subsemigroup of the additive semigroup S^+ of S, so $\langle G\rangle = \{g_1 + \cdots + g_n : g_i \in G\}$. If G has a single element then with $G = \{g\}$, then we also write $\mathbb{P}g = \langle G\rangle_+ = \{g, g+g, g+g+g, \ldots\{$. Similarly, notations $\langle G\rangle_\times = \{g_1 \ldots g_n : g_i \in G\}$ and $g^{\mathbb{P}} = \langle\{g\}\rangle_\times = \{g, g^2, g^3, \ldots\}$.

C.20.1.2 Subsemiring fixed by an element

Let R be a semiring, let $e \in R$. The subsemigroup of $R^\times$ fixed by e, namely $\{s : es = s = se\}$ is also closed under addition, because $e(s+t) = es + et = s + t$, and so on. So it defines a subsemiring.

If e is idempotent, then this subsemiring is eSe. In the subsemiring eSe, the element e is a multiplicative identity.

C.20.2 Solitary extensions of semirings

Recall from §C.8.1 that a solitary extension semigroup S has just one extra element. Examples include S_0, S_1, and shadow extension.

Similar solitary extension of a semiring R can be defined.

C.20.2.1 Extension by zero

Let R_0 be the semiring R with one extra element 0 added.

Diversion C.20.2. If R already has an element labelled 0, then re-label it as 0_R to distinguish it as the old zero. Then $0_R \in R$ and $0 \not R$.

In the additive semigroup R_0^+, the new 0 is an identity element: $0 + r = r = r + 0$ for all $r \in R_0$. In the multiplicative semigroup $R_0^\times$, the new 0 is an absorbing element: $r0 = 0 = 0r$ for all $r \in R_0$.

Diversion C.20.3. In more detail, the underlying set of R_0 is $R \uplus \{0\}$. Addition R_0 is defined

$$r + s = \begin{cases} r + s & \text{if } r \neq 0 \neq s, \\ s & \text{if } r = 0, \\ r & \text{if } s = 0. \end{cases} \tag{C.20.1}$$

Multiplication in R_0 is defined:

$$rs = \begin{cases} rs & \text{if } r \neq 0 \neq s, \\ 0 & \text{if } r = 0, \\ 0 & \text{if } s = 0. \end{cases} \tag{C.20.2}$$

If R already has an element 0 that is both an additively identity element and an multiplicatively absorbing element, then it is often redundant a new 0. Let $R_{(0)}$ be R if R has such a 0, and otherwise, let it be $R_{(0)} = R_0$. In other words, $R_{(0)}$ extends by zero only if necessary.

C.20.2.2 Extension by infinity

Given a semiring R, let R_∞ be the semiring with underlying set $R \uplus \{\infty\}$, meaning the disjoint union, so that ∞ is some arbitrary element which is not an element of R. Define addition and multiplication as follows.

$$r + s = \begin{cases} r + s & \text{if } \infty \not in\{r, s\}, \\ \infty & \text{if } \infty \in \{r, s\}, \end{cases} \tag{C.20.3}$$

and

$$rs = \begin{cases} rs & \text{if } \infty \notin \{r, s\}, \\ \infty & \text{if } \infty \in \{r, s\}. \end{cases} \tag{C.20.4}$$

In other words, to form R_∞ we extend both the additive semigroup R^+ and the multiplicative semigroup by an absorbing element, using the same element, written ∞.

Diversion C.20.4. Semirings extension by zero and by infinity are not isomorphic, unless they have one element. Equivalently, if Q and R are semirings, then either $Q_0 \ncong R_\infty$, or both Q and R are the empty semiring (with no elements).

To see this, note that the unique additively absorbing element of Q_0 is a multiplicative identity. The unique additively absorbing element of R_∞ is multiplicatively absorbing. If

$Q_0 \cong R_\infty$, then the unique additively absorbing element is both a multiplicative identity element and a multiplicative absorbing element. An absorbing identity element is necessarily the only element. So, if $Q_0 \cong R_\infty$ must consist of a single element. Being a solitary extension of Q and of R, means that both Q and R must have no elements.

Diversion C.20.5. This previous note suggests that the solitary extensions by zero and by infinity can be iterated. Using n extensions, and starting from a non-empty semiring, one can get 2^n distinct extensions by such iterations (one extension for each sequence of zeros and infinities). (Starting from the empty semiring, there are 2^{n-1} extensions, because the choice of first extension does not matter.)

C.20.2.3 Shadow extensions

Recall that a shadow extension of a semigroup means to add a second copy of one element of the semigroup, called a shadow, such that any products involving the new shadow element equal the product with the shadow replaced by the original element it shadows.

Shadow extensions can be defined similarly for semirings.

C.20.3 Semiring of polynomials

Given a semiring R (with commutative addition) we can form the semiring $R[x]$ of univariate polynomials in an indeterminate variables.

Univariate polynomials are a standard and well-studied type of object in algebra. We review the formalities only for completeness.

C.20.3.1 A formalism for univariate polynomials

A polynomial f is a formal sum $f = \sum_{i=0}^{d} f_i x^i$ where $f_i \in R$, and d is a non-negative integer. The element f_i are called the coefficients of f and d is called the formal degree.

If R has an element zero 0, then we consider equivalence classes of formal sums. If $f_d = 0$, then we regard $\sum_{i=0}^{d} f_i x^i = \sum_{i=0}^{d-1} f_i x_i$. We then extend this equivalence by transitivity. The effective degree of f is the largest i such that $f_i \neq 0$, if such an f_i exists; the effective degree -1 if all $f_i = 0$.

Polynomial addition obeys the rule:

$$\left(\sum_{i=0}^{d} f_i x^i\right) + \left(\sum_{j=0}^{e} g_j x^j\right) = \sum_{k=0}^{\max(d,e)} h_k x^k$$

where

$$h_k = \begin{cases} f_k + g_k & \text{if } k \leq \min(d, e), \\ f_k & \text{if } k > e, \\ g_k & \text{if } k > d. \end{cases}$$

It is a simple exercise to show that the addition is well-defined under the equivalence of dropping leading zero coefficients.

If addition in R is commutative, then polynomial multiplication obey the usual convolution rule

$$\left(\sum_{i=0}^{d} f_i x^i\right)\left(\sum_{j=0}^{e} g_j x^j\right) = \sum_{k=0}^{d+e}\left(\sum_{i=\max(0,k-e)}^{\min(k,d)} f_i g_{k-i}\right) x^k.$$

Again, it is a simple exercise to show that multiplication is well-defined under the equivalence of dropping leading zero coefficients.

Diversion C.20.6. If R has an element 1, then it is customary to write x^i for $1x^i$.

Diversion C.20.7. We consider R to be subsemiring of $R[x]$, as follow. Formally, we use a semiring morphism that maps r to the formal sum $rx^0 = \sum_{i=0}^{d} f_i x^i$ where $d = 0$ and $f_0 = r$. This semigroup morphism is injective, even under the equivalency classes obtained by dropping leading zero coefficients. This is called the **standard embedding**.

Diversion C.20.8. The variable x in $R[x]$ is a formal variable. One can also use other variables names, when convenient. For example, one can define $R[t]$ and so on. Of course $R[x]$ and $R[t]$ are isomorphic as semiring, with the isomorphism merely being substitution of the formal variables.

Diversion C.20.9. It may be also possible to define polynomials over a base semiring with non-commutative addition. (To be completed.)

C.20.3.2 Bivariate polynomials

Of course $R[x]$ is itself a semiring. It is common, therefore, to form yet another semiring of polynomials with coefficients in $R[x]$. For example, one can form the semiring $(R[x])[y]$. This becomes a ring of bivariate polynomials, which we abbreviate to $R[x, y]$.

Of course, $R[x, y]$ and $R[x, y]$ are isomorphic as semirings.

C.20.3.3 Multivariate polynomials

Similarly, we can form multi-variate polynomials for arbitrarily many variables. In this case, we often write $R[x_1, \ldots, x_n]$.

If we are sufficiently careful, we can also consider semirings polynomials over an infinite set of variables, although each polynomial will only have only finitely many terms as formal sum. For example, $R[x_1, x_2, \ldots]$ can be described as the union $\bigcup_{n\geq 1} R[x_1, \ldots, x_n]$, with the convention that these semirings are nested (chained) under inclusion by the standard embedding.

C.20.3.4 Non-commutative polynomials

It is also possible to define bivariate (and multivariate) polynomials in which the indeterminate variables do not commute with each other, although they do commute with coefficients from the base ring.

For example, bivariate polynomials in non-commuting indeterminates x and y with coefficients that commute with variables can be represented as finite formal sums of the $\sum_i r_i w_i$ where each $r_i \in R$, and $w_i \in \{1, x, y, x^2, xy, yx, y^2, x^3, x^2y, xyx, xy^2, .$ meaning that w_i is essentially a word in the alphabet $\{x, y\}$.

Diversion C.20.10. A notation $R\langle x, y\rangle$ can indicate this.

We can further consider polynomials in which the indeterminate variables do not commute with coefficients of the base semiring R. In this case, it is simplest to consider R which have a multiplicative identity 1, because they we easily an indeterminate variable x with the polynomial $1x1$. In this case, a univariate polynomial with an indeterminate x would be a finite formal sum of the form $\sum_i r_{i,0} x r_{i,1} x \ldots r_{i,d_i-1} x r_{i,d_i}$, where either $r_{i,j} \in R$, and d_i is a non-negative integer.

However, in this form, we would have a further task of deciding when two formal sums are equivalent. Surely, any sensible polynomial arithmetic would imply that $ax(b+c) + cxd = axb + axd + cxd = axb + (a+c)xd$. So, the task of determining equivalence and canonical forms of formal sums might be non-trivial, or might not be even be well-defined.

Diversion C.20.11. The notation $\langle R, x\rangle$ is suggestive for this type of univariate ring.

Diversion C.20.12. If the base semiring R lacks a multiplicative identity, then the terms of formal sums allow some of the coefficients to be absent.

C.20.4 Semiring of matrices

C.20.4.1 Square matrices

Given a semiring R with internally commutative (medial) addition and a positive integer n, we can form a semiring $S = M_n(R)$ of $n \times n$ square matrices with entries in R.

Specifically, writing $a_{ij} \in R$ for the (i, j) entry of a matrix $a \in M_n(R)$, then addition and multiplication are defined by:

$$\begin{aligned}(a+b)_{ij} &= a_{ij} + b_{ij},\\ (ab)_{ik} &= \sum_{j=1}^{n} a_{ij} b_{jk}.\end{aligned}$$

Addition in $M_n(R)$ is associative since it is associative in R. (Addition in $M_n(R)$ is commutative if it is in R.)

Multiplication in $M_n(R)$ is distributive over addition, if it distributive over addition in R.

Multiplication in $M_n(R)$ is associative because

$$(a(bc))_{im} = \sum_{j=1}^{n}\sum_{k=1}^{n} a_{ij}(b_{jk}c_{km}) = \sum_{k=1}^{n}\sum_{j=1}^{n}(a_{ij}b_{jk})c_{km} = ((ab)c)_{im}.$$

Diversion C.20.13. The proof above of multiplicative associativity of $M_n(R)$ uses associativity of multiplication in R, both left and right distributivity in R, and internal commutativity (mediality) of addition in R.

Diversion C.20.14. The base semiring R can have non-commutative multiplication.

If R has an additive identity 0, then so does $M_n(R)$, the matrix with all entries valued 0.

If R also has a multiplicative identity 1, then so does $M_n(R)$: it is the usual matrix of all 0 entries except for entries of 1 on the diagonal serves as a multiplicative identity for R.

If R has additive inverses (negations), so that R is a ring, then so does $M_n(R)$, and $M_n(R)$ is also a ring.

Existence of all multiplicative inverses of nonzero elements of R are not generally enough to ensure their existence in $M_n(R)$ (unless $n = 1$).

Multiplication in R requires at most n^3 multiplication in R and n^3 additions in R. For some R, more efficient multiplication algorithms may be available.

For division in $M_n(R)$, see §5.16.

C.20.4.2 Rectangular matrices

To be completed.

Addition and multiplication of rectangular matrices is sometimes possible, provide that the operands have matching shapes.

The operand matrices in addition $a + b$ must have identical shapes, so a and b have the same number of rows and the same number of columns; the result $a + b$ also has the same shape. The operand matrices in multiplication ab must agree in shape such that the number of columns of a equals the number of rows of b; the result ab has the same number of rows as a and the same number of columns as b.

One attempt to construct a semiring of rectangular matrices works as follows. Add a formal error element, a non-matrix, written as 0, which is the result of any operation where operands do not have matching shapes as matrices. We call this semiring $M_*(R)$.

Diversion C.20.15. The semiring $M_n(R)$ is a subsemiring of $M_*(R)$ for all positive integers n.

Diversion C.20.16. These semigroups are images of the semigroup closure of §**??**: in which the congruence identifies all copies of a given matrix.

Diversion C.20.17. If $0 \in R$, an all zeros matrix 0_n in $M_n(R)$ is a zero of $M_n(R)$, but it is not a zero of $M_*(R)$ (since $0_n 0 = 0 \neq 0_n$.

To do: division of rectangular matrices uses the same principles as for square matrices (linear algebra).

C.20.4.3 Infinitary matrices

The definitions above assume that a matrix is an array with finitely many entries. But there are ways to allow form a similar semiring of arrays with infinitely many entries.

We consider square matrices, but some of these results generalized to non-square rectangular matrix.

Suppose that I is an arbitrary infinite set, and that R is a semiring. An **infinitary square matrix** m is a function $m : I \times I \to R$. (We may also write $R^{I \times I}$ for the set of matrices.) We generally write $m_{i,j}$ instead of $m(i, j)$.

Addition of square matrices is as usual, point-wise as functions, so

$$(a+b)_{i,j} = a_{i,j} + b_{i,j},$$

which is the straightforward generalization of finite matrix addition. We would like to generalize finite matrix multiplication to infinitary matrices, but we encounter a problem: the sum

$$(ab)_{i,k} = \sum_j a_{i,j} b_{j,k}$$

can have infinitely terms, and therefore is not defined for every semiring R, in addition is only a binary operation (which cannot be iterated infinitely often). There are some ways to resolve this.

Summable matrix entries If semiring R has commutative summation over sets of type I, then matrices with index set I can be defined.

Diversion C.20.18. Perhaps the most important example is the semiring of relations, which can be viewed a matrices with entries in bit semiring $\mathbb{B}$, which admits arbitrary summation.

Diversion C.20.19. For another example, let R be the semiring of positive real numbers, together with $+\infty$. It has arbitrary summation. Therefore, infinitary matrices with positive real entries (including $+\infty$) form a semiring.

Column-finite matrices Suppose that the base semiring R has an element 0. A matrix a has is **column-finite** if, for each j, there at most finitely many value i such that $a_{i,j} \neq 0$. In other words, each column of the matrix has finite support (the **support** being the set of matrix position with a nonzero entry).

Column-finite matrices are closed under matrix addition. Also, the usual matrix multiplication for column-finite matrices is well-defined, and results in another column-finite matrix. So, column-finite matrices thus form a semiring.

Diversion C.20.20. When R is a field, column-finite matrices naturally represent linear operators from an infinite vector space R^I.

Row-finite matrices similarly form a semiring. Matrices that are both row-finite and column-finite also form a semiring.

Well-supported matrices??? Let R be a semiring with a 0. Recall that the **support** of a function $f : S \to R$ is $\{s \in S : f(s) \neq 0\}$. Each column and row of a matrix can be viewed as a rectangular matrix, and thus as a function, so it has a support.

Suppose that index set I is equipped with a total ordering. For example, I could be real numbers.

A matrix $a \in R^{I\times I}$ is **well-supported** if the support of each column is a well-ordered subset of I, and the support of each row is an anti-well-ordered subset of I. Matrix multiplication of well-supported matrices is defined because the support of the function nonzero terms in the sum $j \mapsto a_{i,j}b_{j,k}$ is the intersection of a column and row support. It is thus well-ordering in both directions, and thus finite. Therefore the sum is defined.

C.20.4.4 Matrices with non-commutative addition???

To be completed.

The previous definitions of matrices assume that the base semiring R has commutative addition. Commutative addition ensures the associativity of matrix multiplication, and also the distributivity of matrix multiplication over matrix addition.

Actually, commutative addition is not necessary to achieve these goals. For example, for finite matrices, the slightly weaker property of internal commutativity (that $a+b+c+d = a+c+b+d$ for all $a, b, c, d \in R$) seems to suffice.

More generally, a matrix a has **rank one** if there exist two functions $\alpha, \alpha' : I \to R$ such that $a_{i,j} = \alpha_i \alpha'_j$. Rank one matrices are closed under multiplication, and their multiplication is associative.

The distributive might not necessarily hold, however.

To be completed.

C.20.4.5 Natural matrix representations of a semigroup

The section generalizes part of the representation theory of groups to semigroups: specifically the representation of group elements by permutation matrices.

Recall that we write $\mathbb{B}$ for the Boolean semiring, which has two elements 0 and 1. Element 0 is the additive identity and a multiplicative absorbing element. Element 1 is the multiplicative identity, and an additive idempotent.

(In other words, 0 is false, 1 is true, additive is Boolean OR, and multiplicative is Boolean AND.) Recall that $\mathbb{B}$ is a summation semiring, meaning that we can (convergently) sum any infinitely series $\mathbb{B}$.

Let S be any semigroup, with multiplicative notation. Recall that we write $S_{(1)}$ for the semigroup by adding, only if necessary, a multiplicative identity element 1. If S is already unital (a monoid), the $S = S_{(1)}$.

Form a semiring $R = M_{S_{(1)}}(\mathbb{B})$. Recall that matrix multiplication is well-defined, even for infinite S, because $\mathbb{B}$ is a summation semiring. Define a semigroup morphism, the **left natural representation**, as follows:

$$\lambda : S \to R : s \mapsto [a = sb]_{a,b}, \tag{C.20.5}$$

meaning that $\lambda(s)$ is a matrix whose entry in position $[a, b]$ is

$$\lambda(s)_{a,b} = \begin{cases} 1 & \text{if } a = sb, \\ 0 & \text{if } a \neq sb. \end{cases} \tag{C.20.6}$$

By λ being a semigroup, morphism we claim that

$$\lambda(s)\lambda(t) = \lambda(st). \tag{C.20.7}$$

Proof to be added.

Diversion C.20.21. To be clear, the notation above implicitly means that a morphism λ is from S into $R^\times$.

Actually, this is a morphism even if we replace the indices $S_{(1)}$ by S. Using indices $S_{(1)}$ ensures the that morphism is injective.

Similarly, a **right natural matrix representation** has $\rho(s)_{a,b} = [as = b]$.

Actually, the base semiring $\mathbb{B}$ can be replaced by any semiring other B with 0 is a zero element and 1 any multiplicative idempotent, but we may need to restrict the semiring $M_{S_{(1)}}(B)$ to column-finite or row-finite matrices. To be completed.

We can also generalize slightly, allowing multi-valued representations such that nonzero matrix entries belong to a multiplicative subset.

C.20.4.6 Kronecker matrix operations

Kronecker multiplication and addition of integers is also.

To do: (division is easy).

C.20.5 Semigroup semiring

Let R be semiring and G be a semigroup. The **semigroup semiring** $S = R[G]$ is well-known construction, which we review below

Diversion C.20.22. We assume that R has commutative addition (but not necessarily commutative multiplication).

The most general case of $R[G]$ is rather technical, so we handle some simpler cases separately.

C.20.5.1 Finite semigroup semiring

If G is finite, then we can let the underlying set of S be R^G: functions from G to R.

Define addition in S as point-wise addition in R and multiplication as convolution, so

$$(s+t)(g) = s(g) + t(g), \tag{C.20.8}$$

$$(st)(g) = \sum_{fh=g} s(f)t(h). \tag{C.20.9}$$

$$\tag{C.20.10}$$

We consider elements of $s \in S = R[G]$ as finite formal sums $\sum_{g\in G} s(g)[g]$.

C.20.5.2 Semigroup semiring with zero

Suppose that base semiring R has an element 0, which is its additive identity.

For $f \in R^G$, let the **support** of f be the set $S(f) = \{g \in G | f(g) \neq 0\}$.

Let $R[G] = \{f \in R^G : |S(f)| < \infty\}$. In words, $R[G]$ consists of all functions from G to R with finite support.

We define addition as function-wise and note that a sum of two finite support functions has a finite support. We define multiplication in $R[G]$ by convolution, limited to sums over nonzero evaluations. By finite support, these sums are finite. They are therefore defined.

As above, we think of elements of $R[G]$ as finite formal sums.

Diversion C.20.23. If G is finite, then this definition matches the previous exactly, because $R[G] = R^G$. So, these definitions are consistent wherever they overlap.

C.20.5.3 General semigroup semirings

The previous definitions do not cover the situation in which G is infinite and R lacks a zero. With care, a definition of the semigroup semigroup $R[G]$ is possible here, and generalizes the previous two definitions.

Elements of $R[G]$ consist of equivalence classes of charts. A chart is pair of the form (F, f), where F is a finite subset of G and $f \in R^F$ is a function from F to R. Charts (E, e) and (F, f) are equivalent if e and f agree on $E \cap F$ and f is 0-valued on $F - E$ and e is 0-valued on $E - F$, where 0 is the additive identity of R, if it exists. Of course, if R has no 0, then this formalism implicitly requires $E = F$, and therefore $e = f$, so each equivalence class has only one chart.

Diversion C.20.24. To recover the previous definition in which R has a zero 0, represent a function f with finite support $S(f)$ by a chart $(S(f), f)$

Addition of charts is obtained by $(E, e) + (F, f) = (E \cup F, h)$, where

$$h(g) = \begin{cases} e(g) + f(g) & \text{if } g \in E \cap F \\ e(g) & \text{if } g \in E - F \\ f(g) & \text{if } g \in F - E \end{cases} \tag{C.20.11}$$

It is a simple exercise to show the addition of charts respects the equivalence of charts.

Multiplication of charts $(E, e)(F, f) = (D, d)$ is done as follows.

$$\begin{aligned} D &= \{gh : g \in E, h \in F\} \\ d(k) &= \sum_{\substack{g \in E \\ h \in F \\ gh = k}} e(h) f(h) \end{aligned}$$

It is another exercise to show that multiplication of charts respects the equivalence of charts.

We again write elements in $s \in R[G]$ as formal sums, with chart (F, f) being written as

$$\sum_{g \in F} f(g)[g] \tag{C.20.12}$$

With this notation, addition and multiplication act agreeably with the formal sums.

C.20.5.4 Semigroup semirings with non-commutative addition???

To be verified. Probably incorrect.

Let S be a semigroup, and R a semiring, possibly with non-commutative addition.

We try to form a **small semigroup semiring** $R[S]$ as follows.

First, form the free semiring $F_{R\times S}$ which generated by the set $R \times S$. For consistency with previous notation (that for semigroup semirings), write the pairs in $R \times S$ as $r[s]$, with $r \in R$ and $s \in S$.

Diversion C.20.25. The semiring $F_{R\times S}$ has non-commutative addition, by definition.

The semiring $F_{R\times S}$ treats both S and R as sets, entirely ignoring their binary operations.

Next, define a semiring congruence $\sim$ on $F_{R\times S}$ generated by relation of the form:

$$r_1[s] + r_2[s] = (r_1 + r_2)[s], \tag{C.20.13}$$

$$(r_1[s_1])(r_2[s_2]) = (r_1 r_2)[s_1 s_2]. \tag{C.20.14}$$

We then let $R[S] = F_{R\times S}/\sim$. Every element in $R[S]$ can be expressed as an (ordered) sum of elements of the form $r[s]$.

To be completed.

The **large semigroup semiring** $R[S]_+$ starts from the free semiring $F_{R\uplus S}$, with elements in the copy of S written as $[s]$, and the copy of R written as (r). The the congruence is generated by relations:

$$(r_1) + (r_2) = (r_1 + r_2), \tag{C.20.15}$$

$$(r_1)(r_2) = (r_1 r_2), \tag{C.20.16}$$

$$[s_1][s_2] = [s_1 s_2] \tag{C.20.17}$$

Because elements of R operate just like their copy in $R[S]_+$, we can usually drop the parentheses in the notation.

Every element in $R[S]_+$ can be expressed as an (ordered) sum of ordered products, where each ordered product alternates with factors from (copies of) R and S.

To be completed.

C.20.5.5 Recovering the semirings of polynomials and matrices

Let $\mathbb{N}^+ = \{0, 1, 2, \dots\}$ be the semigroup of non-negative integers under standard addition. The semigroup semiring $R[\mathbb{N}^+]$ is isomorphic to the ring of univariate polynomials $R[x]$, mapping $r[n]$ to rx^n.

Let n be a positive integer. Let M_n be the semiring with underlying set $\{0\} \cup \{1, 2, \dots, n\}^2$ with multiplication rule

$$(i, j)(k, m) = \begin{cases} 0 & \text{if } j \neq k, \\ (i, m) & \text{if } j = k. \end{cases}$$

The semigroup semiring $R[M_n]$ is very similar to the matrix semiring $M_n(R)$.

To be completed.

C.20.5.6 Yet another formalism

We can view $R[G]$ as the smallest semiring generated by formal pairs $r[g]$ with $r \in G$ and $g \in G$, which also multiply as $r[g]s[h] = rs[gh]$, and add as $r[g] + s[g] = (r + s)[g]$.

Diversion C.20.26. This suggest yet another way to view $R[G]$. Take $R^\times \times G$, the direct product of semigroups. Form the semigroup semiring $S = \mathbb{P}^+[R^* \times G]$, using the constructions above. Then define a congruence $\sim$ on S such that $[r + s, g] = [r, g] + [s, g]$.

C.20.5.7 Various other considerations

Note that $(R[G])[H]$ is isomorphic to $R[G \times H]$.

Note that if G is a monoid, then it is customary to identify r with $r[1]$, and to view $R[G]$ as an extension of R.

Multiplication of elements in $R[G]$, when represented in the sparse form by (C.20.12) depends on the number of terms.

Division in $R[G]$ can vary considerably in difficulty. If a and b are relatively sparse compare to the size of G, then ab will also be sparse (its support will be smaller than G). In this case a can be recovered fairly efficiently.

C.20.5.8 Non-commutative variants

We can also form a semiring $R\langle G\rangle$ by not allowing the elements of R to commute with those of G.

To be completed.

C.20.5.9 Summation semigroup semirings

To be completed.

If R is a summation ring, meaning that R^+ allows some kind of infinite sums, then it may make sense to allow infinite formal sums, forming a larger set $R[[G]]$.

In particular, if $R = \mathbb{B}$ is the single-bit boolean semiring, then a semiring $\mathbb{B}[[G]]$ with arbitrary formal sums can be formed. More precisely, we can define $\mathbb{B}[[G]]$ to the semiring whose underlying set is the power set $\mathbb{B}^G$ of semigroup G, the set of all subsets of G. Addition in $\mathbb{B}[[G]]$ is the union operation. Multiplication is the multiplication of sets.

A subset of h of G is an element of $\mathbb{B}[[G]]$. Subset h is a subsemigroup if $h^2 \subseteq h$. Using semiring operations, this can be written $h = h + h^2$.

If g is arbitrary element of $\mathbb{B}[[G]]$, the subsemigroup $\langle g \rangle$ of G that it generates can be expressed using arithmetic in $\mathbb{B}[[G]]$ by the formula:

$$\langle g \rangle = g + g^2 + g^3 + \dots \tag{C.20.18}$$

where the summation is allowed in $\mathbb{B}[[G]]$ because (arbitrarily) infinite summation is allowed in $\mathbb{B}$.

C.20.5.10 Twinning a key agreement scheme

The next discussion should probably be placed elsewhere in the report.

Cash, Kiltz and Shoup [CKS08] (CKS) defined the twin Diffie–Hellman non-interactive key exchange (NIKE). As already discussed, in §A.3, any NIKE implies an asynchronous key agreement scheme. Any asynchronous key agreement scheme implies an associative key agreement, and hence a semigroup.

This section describes a generalization of CKS of **twinning** Diffie–Hellman key agreement to a wider class of key agreement schemes. One of the security improvements provided by this generalized twinning procedure, can stated in terms of finding a semigroup with a decision-aided wedge problem of an established difficulty (equal to the difficulty of the wedge problem in some established semigroup).

The twinning construction Given an associative key agreement scheme K associated with a semigroup S, the **twinned** variant K' is an associative key agreement whose associated key agreement scheme is $\mathbb{Z}[S]$, the multiplicative semigroup of the **semigroup algebra** of S over the ring of integers $\mathbb{Z}$.

Diversion C.20.27. Recall that in the standard notion of a semigroup algebra, such as $\mathbb{Z}[S]$, the elements formal linear combinations $\sum_{i=1}^{n} a_i[s_i]$, where the coefficients a_i belong to the ring, in this case $\mathbb{Z}$, and generators $[s_i]$ corresponds to elements $s_i \in S$. (Formally, the free module (or, abelian group for ring $\mathbb{Z}$) over the set S.) Each has a unique form such that $n \geq 0$, $a_i \neq 0$ and all s_i are distinct, which can be obtained from the general form by summing the coefficients of the terms sharing the same semigroup element. (Writing the copies of elements of S in $\mathbb{Z}[S]$ inside square brackets is somewhat optional but avoids confusion when elements of S are themselves written as integers.) Multiplication is given

$$\left(\sum_{i=1}^{m} a_i[s_i]\right)\left(\sum_{j=1}^{n} b_j[t_j]\right) = \sum_{i=1}^{m}\sum_{j=1}^{n} a_i b_j [s_i t_j].$$

The twinned associative key agreement K' also specifies how the base point b' and secrets a' and c' are to be selected. The deliveries d' and e' and keys f' and g' are computed according the multiplication of K'.

Diversion C.20.28. We use the usual standard variable letters for key agreement K', except that mark them with prime tick, to distinguish them from the similar variables in K.

In K', the base point is $b' = [b]$, where b is the base point of S. In K', Alice chooses secret as follows:

$$a' = [a_1] + [a_2], \tag{C.20.19}$$

where a_1 and a_2 are independent random variables in S, each variable being identical in distribution to Alice's secret in associative key agreement scheme K.

Diversion C.20.29. The fact that Alice's secret a in K has been copied twice into a' may justify why this can be called **twinning**.

Similarly, Charlie's secret c' in K' is chosen as $c' = [c_1] + [c_2]$, where c_1 and c_2 are independent copies of Charlie's random variables c for his secret in K.

The rest of a key agreement session in K' follows from the definition of semigroups $\mathbb{Z}[S]$. Expressly, Alice's delivery is

$$d' = a'b' = ([a_1] + [a_2])[b] = [a_1 b] + [a_2 b] = [d_1] + [d_2],$$

where d_1 and d_2 would be Alice's deliveries if she had participated into two sessions of K. Charlie's delivery $e' = [e_1] + [e_2]$ is compute similarly. Alice's key is then

$$f' = a'e' = [a_1 e1] + [a_1 e_2] + [a_2 e_1] + [a_2 e_2]. \tag{C.20.20}$$

Monography in $\mathbb{Z}[S]$ is possible if: S has monography, and also the monograms of S can be sorted. Expressly, first collect coefficients of common terms, sum the coefficients, compute monograms for all semigroup element in the sum. Sort the sum. (In twinned key agreement K, all sums have at most 4 terms, generally much less than in a general element of $\mathbb{Z}[S]$, so the cost of monography in K' is only a little more than four times the cost in K.)

Security benefit of twinning? Cash, Kiltz, and Shoup introduced twinning in order to improve security: in the case of Diffie–Hellman key agreement, they prove that decision-aided wedge problem for the twinned key agreement scheme is equivalent to the wedge problem for the base key agreement scheme.

This section sketches a possible generalization of their proof.

The generalization aims to cover a wide class of base semigroups, but not all. First, we require that the base semigroup S is the multiplicative semigroup of a semiring.

Diversion C.20.30. Recall that a semiring is a pair semigroups, one additive and one multiplicative, sharing the same underlying, and obeying an additional axiom(s) connecting the two semigroups: the distributive axioms: $a(b+c) = ab+ac$ (left) and $(a+b)c = ac+bc$ (right).

Let S' be the multiplicative semigroup of the semigroup algebra $\mathbb{Z}[S]$.

Suppose that V' is an algorithm that solves the aided wedge problem in S', at least for when all inputs match those expected twinned key agreement scheme K'.

Diversion C.20.31. For simplicity of notation, we consider special elements like $d = [d_1]+[d_2]$, but these are technically indistinguishable from $[d_2]+[d_1]$. To avoid such parsing difficulties, one can consider elements like $d = 2[d_1]+3[d_2]$, where the coefficients uniquely determine the decomposition (assuming monography in S). This only adds clutter to the notation, so this trick is omitted from the exposition.

We aim to construct an algorithm V that solves the wedge problem in S, using algorithm V' as a subroutine. In other words, we aim to the wedge in S to the decision-aided wedge in $S' = \mathbb{Z}[S]$.

The job of V is to compute $d \wedge_b e$. To this end, V lets

$$\begin{aligned} d_1 &= d \\ d_2 &= a_2 b + td \end{aligned}$$

for random $a_2, t \in S$. We call t and a_2 the **trapdoor** values.

Next, V calls its subroutine V'. Specifically, V asks V' to compute the wedge

$$([d_1] + [d_2]) \wedge_{[b]} [e].$$

Because V' solves the decision-aided wedge problem, it expects a decision-oracle. Therefore, V must answer the oracle queries for V'.

If V is able to answer the oracle queries from V' correctly, then V' will return an answer $[f_1] + [f_2]$ to the wedge problem $([d_1] + [d_2]) \wedge_{[b]} [e] =$.

Now V can answer f_1, this being the correct value for $d \wedge_b e$.

It only remains to show that V can correctly answer the oracle queries from V'. Recall that an oracle query from V' is a request for the boolean value of the equation:

$$\left[([d'_1] + [d'_2]) \wedge_{[b]} [e'] = [f'_1] + [f'_2]\right].$$

Now, V uses its trapdoor values to answer (hopefully correctly) the query with boolean value:

$$[(a_2 e' + t f'_1) = f'_2]. \tag{C.20.21}$$

It is straightforward to verify one direction of implication between these boolean values of the oracle query responses. The other direction may require some cancellation or non-lossiness in one or both of the operations of the semiring S.

To be verified and completed.

C.20.6 Semiring of additive subsemigroups?

Let S be a semiring, with additive semigroup written S^+ (as usual). Let $A = \mathrm{Add}(S)$ be the set of subsemigroups of S^+.

We try to make A into a semiring by defining operations addition and multiplication as follows:

$$\begin{aligned} a + b &= \langle a \cup b \rangle_+, \\ ab &= \langle \{st : s \in a, t \in b\} \rangle_+, \end{aligned}$$

where (recall) $\langle G \rangle_+$ means the smallest subsemigroup of additive semigroup S^+, such that the subsemigroup contains G.

Lemma C.20.1. *The construction $A = \mathrm{Add}(S)$ is a semiring.*

Proof. Both operations use additive enclosure operator $\langle \cdot \rangle_+$ as their final step so they both result in subsemigroups of S^+.

Additive associativity holds because $a+(b+c) = \langle a \cup b \cup c \rangle_+ = (a+b)+c$. (To elaborate, elements in $b+c$ have the form $t_1+t_2+\cdots+t_n$ where $t_i \in b \cup c$, while elements in $a+(b+c)$ have form $s_1+s_2+\cdots+s_m$, where $s_i \in a \cup (b+c)$. Each term s_i in can be replace by a sum like $t_1 + \cdots + t_n$, expanding the whole into a sum all of whose terms belong to $a \cup b \cup c$.)

Multiplicative associativity holds because $a(bc) = \langle \{stu : s \in a, t \in b, u \in c\} \rangle_+ = (ab)c$. (To elaborate, elements in elements in bc have the form $t_1u_1 + \cdots + t_nu_n$ where $s_i \in b$ and $u_i \in c$. Elements of $a(bc)$ have the form $s_1v_1 + \cdots + s_mv_m$ where $s_i \in a$ and $v_i \in bc$. Each factor v_i can be replaced by factor of the form $t_1u_1+\cdots+t_nu_n$, so that the term s_iv_i becomes $s_i(t_1u_1 + \cdots + t_nu_n) = s_it_1u_n + \cdots + s_it_nu_n$.)

Left distributivity holds because

$$\begin{aligned} a(b+c) &= \langle \{st : s \in a, t \in b+c\} \rangle_+ \\ &= \langle \{s(t_1 + \ldots t_n) : s \in a, t_i \in b \cup c\} \rangle_+ \\ &= \langle \{st_1 + \cdots + st_n : s \in a, t_i \in b \cup c\} \rangle_+ \\ &= \langle \{st : s \in a, t \in b \cup c\} \rangle_+ \\ &= \langle \{st : s \in a, t \in b\} \cup \{st : s \in a, t \in c\} \rangle_+ \\ &= \langle \{st : s \in a, t \in b\} \rangle_+ + \langle \{st : s \in a, t \in c\} \rangle_+ \\ &= ab + ac, \end{aligned}$$

and right distributivity holds similarly. □

The empty set belongs to A, and is an additive identity and a multiplicative zero, so we write 0 for the empty set.

Element $a \in A$ is **finitely generated** (or just **finite** if clear from context) if $a = \langle G \rangle_+$ for a finite set G. The finitely generated elements of A form a subsemiring of A.

Finitely generated elements of A have a potentially epigraphy (representation), the finite generating set. However, finding a monography, a unique representation might be more difficult. So, practicality of this semiring is not a given.

Diversion C.20.32. Another formalism for $A = \mathrm{Add}(S)$ is to use to the two semirings $\mathbb{B}[[S^\times]]$ and $\mathbb{B}[[S^+]]$. An operation (adding or multiplying) in A is done by first applying the operation in $\mathbb{B}[[S^\times]]$, and second applying the function $\langle \cdot \rangle_+ : g \mapsto g + g^2 + g^3 + \dots$ in $\mathbb{B}[[S^+]]$.

To be confirmed: semiring $\mathrm{Add}(S)$ can be seen as congruence semiring, with morphism $\langle \cdot \rangle_+ : \mathbb{B}[[S^\times]] \to \mathrm{Add}(S)$. The lemma above can be viewed as claiming that this function is a semiring morphism.

The finitely generated elements of $\mathrm{Add}(S)$ are then just the images under this morphism of the usual semigroup semigroup $\mathbb{B}[S^\times]$ consisting only of formal finite sums. The epigraphy of using finite generating sets to represent these elements is simply to use the elements in $\mathbb{B}[S^\times]$ in the pre-image of the morphism. Making this into a monography requires determining a unique pre-image.

If $e \subseteq S$ is a subsemiring of S, then $e \in A$ and $e^2 \subseteq e$. Conversely, if $e \in A$ and $e^2 \subseteq e$, then e is a subsemiring of S.

If $e^2 = e$, then e can be called a **surjective subsemiring**. If e is a subsemiring with a multiplicative identity (such as the multiplicative identity of S, or some other element), then $e^2 = e$. (Question: can $e^2 = e$ with e lacking a multiplicative identity?)

Recall that the subset of A of elements fixed by e under multiplication forms a subsemiring. If e is surjective subsemiring, then the set of these elements is eAe. These elements of A can be called the **relative ideals** of e in A. The semiring of these relative ideals has a zero 0 and a multiplicative identity e.

If e is fixed (so clear from context) write $e = 1$, when looking at the semiring $E = eAe$.

It is not uncommon in algebra to consider **class groups**, with a construction similar to the above. Let S be a field, let e be a subsemiring (usually whose field of fractions is S). Form semiring $E = eAe$, and then ask if all elements are finitely generated, ask if all invertible, and so on. Often looks at the subsemigroup P of $E^\times$ consisting of **principal ideals**, those that $p = \langle \{g\} \rangle_+$, those generated a single element set. (Principals ideals are closed under multiplication, but not addition.) Then one can form the quotient semigroup $E^\times/P$.

Practicality (especially monography) of such constructions is generally not easy.

C.20.7 Hahn series

To be completed.

Suppose that G is a strictly ordered semigroup, and that R is semiring.

We extend the $R[G]$, as follows. Instead of only functions with finite support, we instead allows charts with well-ordered support.

To be completed.

C.20.8 Semiring of resultants (???)

Diversion C.20.33. This section is highly speculative, it is likely to be wrong.

It is supported only by lots of guesswork on part, with a only few machine-aided calculations.

Let R be a unital commutative ring. We try to define a semiring $S = \mathrm{Res}_{x,y}(R)$, which we dub the **resultant semiring** (or **curve semiring**).

The elements S are equivalence classes of **bimonic** polynomials: polynomials $b(x, y) \in R[x, y]$ such that $b(x, y)$ or $-b(x, y)$ monic[2] in variable x, and similarly for y. For example, $y - x$ is bimonic because $-(y - x) = x - y$ is monic in x, and $(y - x)$ is monic in y.

The equivalence classes are $\{b(x, y), -b(x, y)\}$, which we write as $[b(x, y)]$.

Addition and multiplication in S are given by the laws:

$$[a(x, y)] + [b(x, y)] = [a(x, y)b(x, y)]\,, \tag{C.20.22}$$

$$[a(x, y)][b(x, y)] = [\mathrm{Res}_t\, a(x, t)b(t, y)]\,, \tag{C.20.23}$$

where Res_t indicates forming the **resultant** with respect to variable t.

I do not have a proof that this forms a semiring.

C.20.8.1 A definition for the resultant

We now review one possible definition of the resultant as it is used in the multiplication definition for S.

[2] A polynomial is monic in a variable x if its highest degree term in x has coefficient 1.

Given any ring of bivariate polynomials $R[u, v]$ (now in arbitrary variables u and v, and any ring R, not necessarily the ring R this section started with), define the **Bezout differential** as a function:

$$\Delta_{u,v} : R[u, v] \to R[u, v] : g(u, v) \mapsto \frac{g(u, v) - g(v, u)}{u - v}.$$

To see that $\Delta_{u,v}$ is well-defined, notice is R-linear and sends

$$\Delta_{u,v} : u^i v^j \mapsto \pm \left(u^{\max(i,j)-1} v^{\min(i,j)} + u^{\max(i,j)-2} v^{\min(i,j)+1} + \cdots + u^{\min(i,j)} v^{\max(i,j)-1}\right),$$

which is a bivariate polynomial.

Working in the same ring $R[u, v]$, for any positive integers i, j, define R-linear **truncation** operators:

$$\tau_{u,i,v,j} : R[u, v] \to R[u, v] : u^m v^n \mapsto \begin{cases} u^m v^n & \text{if } m < i, n < j, \\ 0 & \text{if } (m, n) \notin [0, i-1] \times [0, j-1]. \end{cases}$$

Hence, $\tau_{u,i,v,j} g(u, v) = g(u, v)|_{x^i=0, y^j=0}$. The operator $\tau_{u,i,v,j}$ can considered be projection from $R[u, v]$ down to a subset with maximum degree in u of $i - 1$ and v-degree of at most $j - 1$.

Let M be a square matrix with entries $R[u, v]$. Define a **(Bezout) truncation** matrix operator by

$$(T_{u,v} M)_{i,j} = \tau_{u,i,v,j} M_{i,j},$$

where we index rows and columns with positive integers (so, starting from 1 not 0.)

For any non-negative matrix, let J_m be the $m \times m$ square matrix with all entries valued at one. If d is a scalar, then $J_m d$ is the $m \times m$ square matrix with all entries valued d.

Given $[a], [b] \in S$, define the **composition degree** as

$$m(a, b) = \max(\deg_y(a(x, y)), \deg_x(b(x, y)))$$

When a, b are clear from context, we abbreviate $m(a, b)$ to m.

A formula for the product in S is:

$$[a][b] = \left[\frac{1}{(uv)^{\binom{m+1}{2}}} \det T_{u,v} J_m \Delta_{u,v} a(x, u) b(v, y)\right].$$

(If $m = 0$, then we use the convention that the determinant of a 0×0 matrix is 1.)

My hope is that this expression is a useful step towards determining the associativity of the multiplication in S.

C.20.8.2 Morphism to the ring of relations

Consider the semiring T of relations on R.

Then it seems that there is a semiring morphism:

$$h : S \to T : [a(x, y)] \mapsto \{(x, y)|a(x, y) = 0\}.$$

In some sense, the set of planar affine algebraic curves on R defines a subset of the relations on R. The claimed morphism above shows that such algebraic curve relations form a subsemiring $h(S)$ of the semiring T of relations.

C.20.8.3 Examples in the resultant semiring

To be completed.

Let

$$\begin{aligned} 0 &= [1] \\ 1 &= [y - x] \\ n &= [y + x] \\ t &= [y - x^2] \end{aligned}$$

It is generally true that $[y - f(x)][y - g(x)] = [y - g(f(x))]$.

C.21 Semigroups (and semirings) from categories

C.21.1 Category algebra

Given any category C and any semiring R, one can form the category algebra $R[C]$, which is a semiring as follows.

Elements of $R[C]$ are formal sums of zero or more terms. Each term has the form $r[f]$ where f is a morphism between two objects in the category.

Terms with a common f add, so that $r[f] + s[f] = (r + s)[f]$. Otherwise they do not add. In other words, $R[C]$ is a free R-module generated by the morphisms of C.

Terms multiply as $r[f]s[g] = (rs)[fg]$ if the morphism fg is defined. If morphism fg is not defined in the category, then the terms multiply to 0 (the empty sum).

If the C is a finite category with one morphism between objects then $R[C]$ is effectively a matrix semiring.

If C is the category obtained from the relation partially ordered set, then $R[C]$ is called an incidence algebra. An incidence algebra is a subalgebra of a matrix algebra. This leads one to think that the matrix division algorithms can be used for division in an incidence algebra.

If C is a finite concrete category, with a finite number of objects, each of which is a structure with an underlying finite set, and the morphisms are faithfully represented by functions between the underlying sets, then the category algebra can be embedded into a matrix as follows. A block matrix form is used. The blocks are indexed by pairs of objects, the morphism set. Each block is a rectangular (or square) matrix, indexed by elements of the underlying of the two objects indexing the block. Each block matrix is a sum of term matrices of the same shape, with one term matrix per term of the category algebra sum. A term matrix has all entries of 0 except where the morphism of the term maps one object to another, where the term matrix entry is the semiring value of the term coefficient.

C.21.2 Product and co-product semigroups

Many important categories have an operation on objects called the product. If the products of objects exists, then the product is associative on the isomorphism classes of the objects. So, in categories where products exist, the set of isomorphism classes under the product forms a semigroup. This semigroup is commutative.

For example, in sets, the category product is the Cartesian product. In groups and rings, the category product is the direct product (similar to the Cartesian product).

Similarly, some categories also have a co-product. When defined it is also an associative operation, forming a commutative semigroup on the isomorphism classes.

For example, in sets, the co-product is the disjoint union. In groups, the co-product is the free product. In rings, the co-product is the tensor product.

Sometimes, the product is distributive over the co-product. For example, this holds on the category of sets. In this case, the isomorphism classes form a semiring (see below). In the category of sets, the isomorphism classes are called cardinalities, and represent the sizes of the sets.

In yet other categories, the co-product is distributive over the product. For example in rings and modules. In this case, the isomorphism also a form a semiring.

To be completed.

C.21.3 Category completions

Given a category C, one can form a semigroup S as follows. The set S is the set of morphisms in C, together with one additional object, distinct from all C-morphisms, which we will call 0.

Multiplication in S is defined to be morphism composition, except that 0 behaves like a zero should, so $0s = s0 = 0$ for all s.

In the other direction, any semigroup can be used to create a category, by turning semigroup elements into morphisms. There can be many ways to do this. For example, we can form a category with one object. Take semigroup S. Recall that we can embed S into a monoid M, such as $M = S_1$ (the extension by a new unit) or the $M = S_{(1)}$ (the conditional addition by a unit). Now define a category C with one object O, and morphisms $\mathrm{Mor}(O, O) = M$, where morphism composition is defined as the multiplication in M.

C.22 Semirings from rings

Rings provide the most interesting examples of semirings.

The theory of rings is very extensive. We only list a few examples that might be interesting.

C.22.1 Weyl algebra

The Weyl algebra is non-commutative $\mathbb{Z}\langle x, d\rangle/(dx - xd - 1)$. In a sense, d acts like the operator of differentiating a function by x, sending $f(x)$ to $f'(x)$; and x acts like the operator of multiplying a function by x, sending $f(x)$ to $xf(x)$.

The Weyl algebra might be interesting in that it has no finite matrix representations. Therefore, matrix division might not be useful to divide in the Weyl algebra.

Furthermore, the Weyl algebra is non-commutative, and lacks a pre-divider. Consider a ring of square matrices with entries in the Weyl algebra.

Division in this ring might be difficult: or, at least, I have no clue how to divide in that ring. Perhaps additive decomposition is easy enough, however.

C.22.2 Integer-valued polynomials

Already briefly discussed. These are integer combinations of binomial functions. They are interesting for being a non-Noetherian ring. As such, they lack some of the finiteness properties of Noetherian ring without being too complicated.

This algebra is a sub-algebra of a Noetherian polynomial algebra so division is no harder in it. But ideal arithmetic might perhaps be more difficult.

C.22.3 Semiring of (fractional) ideals

To be completed.

Let R be a ring. Let $I(R)$ be the set of ideals of R. Ideals can be added and multiplied, and ideal multiplication is distributive over ideal addition. Therefore, $S = I(R)$ is a semiring.

If R is a polynomial ring, then Grobner bases can be used to find unique representation for each ideal in $I(R)$. For other rings, it might be difficult to find unique representations: in which case they would be difficult to use for associative key agreement.

If R is a polynomial ring, then division can be done using the ideal quotient operation. Effective algorithms are known for computing ideal quotients [CLO91, §4.4].

If R is an integral domain, then the set $I(R)$ can be extended a larger set $F(R)$ of fractional ideals. Let K be the field of fractions of R. Note that K is an R-module. The set of R-modules can be added and multiplied to form a semiring. The fractional ideals form a subsemiring of the R-submodules of K. It is those submodules F such that there is an element $x \in K$, with $kF \subseteq R$.

The multiplicative semigroup $F(R)^*$ has a subsemigroup P of principal ideals Rx for $x \in K$. Typically, one can use P to define a congruence $\sim$ on $F(R)^*$ and get a class semigroup $C(R)$.

A class semigroup is a group if R is a Dedekind domain.

Elliptic curve groups are actually class semigroups for a suitably chosen ring R.

C.23 Semirings from calculus

Let R be the set of continuous real-valued functions define unit square $[0, 1] \times [0, 1]$. Define a product by:

$$(f * g)(x, y) = \int_0^1 f(x, t)g(t, y)dt. \tag{C.23.1}$$

Define addition in the usual way. This sometimes known as the Fredholme product.

The Fredholme problem is find f given g and $f * g$. This is an example of an integral equation.

Note Fourier analysis can be applied to convert the Fredholme product into an product of infinite dimensional matrices.

This semiring is R is infinite, and in fact, uncountably infinite. For a cryptographic application, we would need to restrict to a finite or at least countably subset of R.

Appendix D

Trial search review: some basics

This appendix reviews trial search as it applies to key agreement, as mentioned in §3.2.8.

Consequently, trial search is thus applicable to semigroup division.

Trial search is even more general, being applicable, for example, for general password or key search (given the output of a known deterministic function evaluated at a secret).

Some of the terminology used in this appendix may not be the standard terminology.

D.1 Trial search strategies

This section describes **trial search** in general terms, allowing for considerable variation.

A typical user of trial search is **Betsy**. When Bugsy is trying to implement a watcher to attack a key agreement scheme, he might try to use trial search. In this case, Bugsy outsources the implementation of trial search to Betsy.

D.1.1 Inputs to trial search

The primary inputs to trial search are:

- A function h.

- A random variable a.

The function h and random variable a is assumed to efficient. So, h can be implemented efficiently, and a can be sampled efficiently. Generally, the random variable can be sampled independently. In some cases, the trial search may also know a lot about the probability distribution of a, which is more than the mere ability to sample a. The random variable is to take values in the domain of h. So, $h(a)$ is another random variable.

When attacking key agreement, trial search could take h defined as $h(a) = k_1(a, b)$, where b is a known base element. When attacking probabilistic key agreement, trial search could take a as the same random variable used by Alice to generate her value of a. In practice, Alice likely uses known software and hardware to generate a. The trial searcher can can obtain a copy of Alice's software and hardware, and may therefore be able to replicate Alice's random variable identically.

D.1.2 Target of trial search

Trial search has a target:

- A value d, or its equivalent.

The value d is promised to be an independent sample of the random variable $d = h(a)$.

Trial search, by definition, uses the target d (or δ) in only one way: to test the correctness of trials, trying to find a value a' such that $h(a') = d$.

An **equivalent** to d here means a function δ, such that $\delta(d') = 1$ if $d' = d$, and $\delta(d') = 0$ if $d' \neq d$. The function δ might not enough to determine d. For simplicity, we will assume that the target d is available.

When attacking key agreement, the target could be set to Alice's delivery d. (Trial search can also be applied to Charlie's delivery e, with a suitable change to the trial search inputs.)

D.1.3 Iterations of trial search

Trial search iterates through some number n of trials.

- Each trial generates a value a' in the domain of the function h.

Trial search allows total freedom the selection the value a' to test in each trial, so long as the as the selection depends only on the primary inputs h and a. If the selections of a', depend on the target, then it is not trial search, but some other attack.

Different methods to select trial values a' are just different variants (or strategies) for trial search.

Trial search may have some resources, such as a source of randomness (other than just random variable a), to generate trial values a'.

Formally, we may view trial search as a function:

$$[h, a] \mapsto [a'_1, \ldots, a'_n]. \tag{D.1.1}$$

The search list $[a'_1, \ldots, a'_n]$ is obtained from the primary inputs.

D.1.4 Success of trial search

Trial search **succeeds** as soon $h(a') = d$, for one of the trial values a'. In this case, we say that a' is a **discovery**. So, trial search succeeds once it finds a discovery.

Diversion D.1.1. Trial search is a type of **second pre-image attack** against h: finding a' such that $a' = k_1(a)$.

The converse is not true. Not all second pre-image attacks are trial searches. More sophisticated second pre-image attacks are given the value $d = h(a)$, and find a discovery a' such that $h(a') = d$, using some process that inverts the function h.

So, trial search is second pre-image attack in which the image is withheld from the attacker.

Diversion D.1.2. The fairy tale Rumpelstiltskin perhaps illustrates trial search. A princess must guess an imp's name using trial search in three days. The imp promises to tell the princess whether or not her guesses are correct.

The princess finds trial search to be ineffective, so she uses a more sophisticated pre-image attack, trying to get more information from the imp than the imp intended. Luckily for the princess, the imp foolishly re-uses his own secret name in another protocol, a weak protocol that leaks his name. The princess uses a zero-day exploit to capture his name from this weak protocol.

The imp had expected trial search to fail, with fairly good reason. The fairly tale thus has a lesson that more sophisticated pre-image attacks can be much more effective than trial search.

D.1.5 Impact on asynchronous key agreement

If Bugsy has recorded Charlie's delivery e, and Betsy has provided him a potential discovery a' against Alice's delivery $d = h(a') = k_1(a', b)$, then Bugsy can compute a key f', which will equal (Alice and) Charlie's key g, as follows:

$$\begin{aligned} f' &= k_3(a', e) \\ &= k_3(a', k_2(b, c)) \\ &= k_4(k_1(a', b), c) \\ &= k_4(d', c) \\ &= k_4(d, c) \\ &= g \end{aligned} \tag{D.1.2}$$

Diversion D.1.3. This attack does not require Bugsy to observe Alice's delivery d.

Instead, the attack uses the event that $f' = g = f$ as an equivalent to d. Bugsy computes f' as above, and then tests whether it is consistent with the way Alice or Charlie used their key.

If Alice or Charlie uses $f = g$ to encrypt or authenticate a message, then Bugsy can test each f' by decrypting the ciphertext, or verifying the authentication. Bugsy can run such tests on off-line.

If Bugsy observes d, then testing if $h(a') = k_1(a', b) = d$ might be easier than trying f' as decryption or verification key.

D.1.6 Trial multiplication for division

Trial search can apply to try to implement (weak) division in a (multiplicative) semigroup S, meaning to compute an operator / such that $((ab)/b)b = ab$ for all $a, b \in S$. For some versions of the trial search, the divider will be faulty, and will fail.

To compute d/b from d and b, do the following. Begin a trial search with the primary inputs h and a as follows. Let h be the function defined $h(e) = eb$, for the given fixed $b \in S$. Let a be some S-valued random variable a. If S is finite or countable, assume that a has nonzero probability on every S.

Trial search then seeks value a' such that $h(a') = d$, where d is the target. As soon as a discovery is made, meaning $h(a') = a'b = d$, then output a' as the value of d/b.

If no discovery is made in the n trials, then output some arbitrary value, such as d itself, for the value of d/b. If n is chosen to be infinite, then trial search might never halt.

If $d = ab$ and trial search makes a discovery, then $ab = d = h(a') = a'b$. So $((ab)/b))b = (d/b)b = a'b = ab$, as required for a (weak) divider.

Because the trial search success involves a multiplication, then it makes sense to relabel trial search in this setting as **trial multiplication**.

Diversion D.1.4. The term **trial multiplication** as a division algorithm is large consistent with the term **trial division** as a factorization algorithm. Indeed, a trial division algorithm can be regarded as a kind of trial search.

D.2 Metrics for trial search

Trial search can use a large variety of methods to choose trial values a'. In this section, we define some metrics for comparing variant strategies of trial search.

Diversion D.2.1. We prefer to use standardized units of **bits**, by taking base-two logarithms of the number of trials, and base-two logarithms of the inverse probability of success.

D.2.1 Runtime and workload

If the trial searcher Betsy has the resources to make at most $n = 2^w$ trials: then we say that Betsy has **workload** of w bits. For convenience, we may use looser terminology for certain ranges of w.

- If Betsy's workload is 0, we say that Betsy is attempting **instant search** (or **surmisal**).
- If Betsy's workload is 4, way say that Betsy is attempting **limited search**.
- If Betsy's workload is around 20, we say that Betsy is attempting **efficient search**
- If Betsy's workload is around 64, then we say that Betsy is attempting **feasible search**.
- If Betsy's workload is around 128, then we say that Betsy is attempting **extensive search**.

Diversion D.2.2. In some cases, the function h itself, or some other method that Betsy might use to test trials might be costly. For example, each trial-test requires an iterated hash function.

In this case, we may want to add an appropriate adjustment to the workload (and re-name the previous workload to **guessload**).

Diversion D.2.3. In addition to low workload, instant and limited search might have extra benefits for Betsy.

Each trial requires a test, and in some case, the test requires an on-line interaction with one of the parties under attack. These parties might sensibly choose to limit the number of tests.

D.2.2 Success rate and entrophy

Betsy's **success rate** is the probability of Betsy of making a discovery.

Diversion D.2.4. When calculating probabilities (including those used in averages) in the success of trial search, we define the probabilities over the internal choices made by Betsy and also the external choices made in sampling of the target d.

Consequently, the probabilities (and averages) refer to the outcome game before the target and before the trials have been selected.

It is often convenient to translate success rate into the same units as used for workload: bits. So, if Betsy has success rate 2^{-h}, then we say that Betsy has **entrophy**[1] of h bits.

Diversion D.2.5. The term **entrophy** is non-standard, but has the mnemonic connections to entropy and success rate (trophy).

D.2.3 Eternal search

In **eternal search**, Betsy keeps searching until she makes a discovery. In this case, we can rate Betsy according to the average number of trials until a discovery is made. Again, we take the base two logarithm to get a workload in bits.

This metric might sound like a sensible way to handle multiple different workloads simultaneously. This viewpoint is misleading.

The average workload can include rare events where Betsy's makes a totally infeasible amount of trials. This average can severely underrate Betsy. For example, there may be a rare event of probability 2^{-128}, in which Betsy

[1] Not entropy, but related, as discussed later.

uses 2^{2048} trials. Averaging means that this event contributes $2^{2048-128} = 2^{1920}$ to the average number of trials, of a workload of at least 1920 bits.

The number 1920 bits seems very secure. But what if the rest of time, with probability $1 - 2^{-128}$, Betsy only needs 2^{10} trials? Her average workload is about 1930, but Betsy's basically has efficient trial search with success rate $1 - 2^{-128}$.

Diversion D.2.6. One might infer a general principle that it is unwise to average the adversary's costs in cryptography.

Diversion D.2.7. Perhaps, some alternative to conventional average (the arithmetic mean) will resolve this, and will cost eternal search more sensibly.

Diversion D.2.8. Betsy might not be fully aware of her workload (her resources). For example, she may start her search, and think that she is conducting eternal search.

Workload does not refer Betsy's state-of-mind. Workload refers to her resources and capabilities. Even if Betsy thinks she is doing eternal search, she likely only has the budget to feasible search, and not being able to afford the energy to run longer searches.

So, if Betsy sets her algorithm to run forever, as though it were eternal search, but realistically only has resources for doing a 2^{50} trials, then we can consider her workload to be 50 bits.

D.3 Imitative search

In **imitative search**, Betsy uses trial search where each trial a' is chosen:

- randomly,
- independently (of other trials),
- with a probability distribution identical to the given random variable a in the primary input to the trial search.

Diversion D.3.1. In key agreement, a trial search attack could include Alice's probability distribution for her variable a as one of the inputs (supplied to Betsy). In imitative search, Betsy merely uses Alice's distribution as an oracle to generate trials.

One can argue that this formalism is realistic, since Alice's algorithms and processes for selecting her value a should not be assumed to be secret. For example, Betsy might obtain a copy of Alice's computing device. So, generally, it is not unreasonable to assume that that Betsy can deduce and replicate the probability distribution of a.

Nonetheless, it may (arguably) be difficult for Betsy generate trials independently in the vast quantities of high-workload trial search. In this case, imitative search would no longer be realistic.

Arguably, Alice might choose a strong password, and incorporate a strong password into the selection of a. In this case, Betsy would not have the password, but would have to some random variable that models a typical strong user password, and then incorporate password random variable into the system that Alice uses.

Imitative search is so simple that we can consider how it fares when combined with eternal search.

Lemma D.3.1. *If Betsy uses imitative eternal search, then Betsy's average number of trials (averaged over the choices of both Alice and Betsy) equals the number of target values that have nonzero probability $d = h(a)$, in other words, the Hartley entropy $H_0(d)$.*

Proof. Let $p(d)$ be the probability that $d = h(a)$. Let Betsy's trials be $a_1, a_2, \ldots,$ and write $d_i = k_1(a_i)$. Since Betsy imitatively chooses each a_i with exactly the same probability probability as a, the probability that $d_i = d$ is $p(d)$, and the probability that $d_i \neq d$ is $(1 - p(d))$.

The event that Betsy takes t trials to make a discovery is the event that $d_1, \ldots, d_{t-1} \neq d$ and $d_t = d$. The probability of taking t trials is thus

$$(1 - p(d))^{t-1} p(d).$$

The average number of trials that Betsy needs

$$\sum_{t=1}^{\infty} t(1 - p(d))^{t-1} p(d)$$

This infinite sum equals $\frac{1}{p(d)}$, when $p(d) \neq 0$.

We also seek the average over the choice of target d. This means we should sum over all d, such that $p(d) \neq 0$, weighted by the probability. So the average number of trials is:

$$\sum_{d:p(d)\neq 0} p(d) \frac{1}{p(d)} = \sum_{d:p(d)\neq 0} 1$$

which is the number of deliveries with nonzero probability. □

Diversion D.3.2. Per earlier discussion, average number of trials eternal search sometimes overrates security. Nonetheless, the lemma can serve as a caution to Alice on the necessity of adequate Hartley entropy.

Diversion D.3.3. It should be clear within the proof that the number of trials needed in eternal search depends on the target d. Some targets could be more likely than others, and eternal search will require fewer trials for these more likely targets.

The average number of trials refers to average over all targets. It is not conditioned on the targets. It rates the effectiveness of imitative eternal search before target selection.

Another special case of interest is instant search.

Lemma D.3.2. *If Betsy uses imitative instant search (workload of 0 bits), then Betsy's entrophy (logarithm of success rate) is the Renyi entropy of order two (of the probability distribution of Alice's deliveries).*

Proof. Label the possible target be $d_1, d_2, \ldots$, and let their probabilities be $p_1, p_2, \ldots$ (respectively).

Betsy chooses a' with the same probability distribution random variable a, and gets a candidate delivery $d' = k_1(a')$, with the same probability distribution as the target d. The probability of that $d' = d_i$, is p_i.

Betsy chooses her trials a' independently of the value a that was used to generate the target. The probability that both $d = d_i$ and $d' = d_i$ therefore is p_i^2, because independent probabilities multiply . Summing over all i, gives a probability

$$\sum_{i=1}^{\infty} p_i^2,$$

that Betsy $d = d'$. Betsy's entrophy is the negative of the base two logarithm of this probability,

$$H_2(p) = -\log_2 \sum_{i=1}^{\infty} p_i^2,$$

which is also known as the Renyi entropy of order 2. □

The more general result is:

Lemma D.3.3. *If Betsy uses imitative search workload of w bits, then Betsy's entrophy is:*

$$-\log_2 \sum_{i=1}^{\infty} p_i \left(1 - (1 - p_i)^{\lfloor 2^w \rfloor}\right).$$

Proof. Betsy takes at most t trials where $t = \lfloor 2^w \rfloor$.

With notation as in the previous proofs, the probability that every one of t of Betsy's trials fails, meaning $d' \neq d_i$, is $(1 - p_i)^t$. □

Diversion D.3.4. The entrophy values in the two lemmas before to this last lemma he are related in Renyi entropies. Presumably, the more general third lemma can be stated as relation between entrophy and Renyi entropy.

Imitative search, though generally simple, is not an optimal search.

For example, it makes no effort to avoid repeated trial values a'. It makes no effort to avoid repeated values of $d' = h(a')$. It uses random variable a merely as an oracle to be sampled, without examining whether some values are more likely than others.

D.4 Optimal search

In **optimal search**, Betsy uses a pre-search phase that studies both h and the random variable a, to determine the following information, or its equivalent.

The target d is a sample another random variable $d = h(a)$. Each target value of d has a probability, which we write as $p(d)$.

There exists a sequence distinct target values $[d_1, d_2, \ldots]$ such that the probabilities $p_i = p(d_i)$ obey:

$$p_1 \geq p_2 \geq p_3 \geq \ldots \tag{D.4.1}$$

$$\sum_{i\geq} p_i = 1. \tag{D.4.2}$$

(Actually, this assume that random variable $d = h(a)$ is a discrete random variable, not a continuous random variable.)

Betsy also determines a sequence of values $[a_1, a_2, \ldots]$ such that $d_i = h(a_i)$. Betsy's search consists of the trial values $a_1, a_2, \ldots,$ up to however many attempts her workload permits, or forever if Betsy will try eternal search.

We call this strategy **optimal search** because it maximizes Betsy's success rate for a given number of trials, and it also minimize the average number of trials in an eternal search.

In referring to optimal search, we presume that that Betsy has no cost in the pre-search phase. In the real world, the pre-search phase might have a huge cost. For example, if a is a nearly uniform random variable with high entropy, and h is a pre-image resistant function, then even finding a_1 could be very costly. In this case, optimal search seems not to be any better than imitative search.

Nonetheless, if h and a make optimal search infeasible, not counting presearch phase, then other types of trial search would be infeasible too.

When Betsy uses the optimal search strategy, her entropy is determined the delivery's probability distribution. It is a parameter sometimes called[2] **working entropy**, defined as

$$W_w(p) = -\log_2 \sum_{i=1}^{\lfloor 2^w \rfloor} p_i \tag{D.4.3}$$

for a **workload** of w bits. (Reminder: $p_i \geq p_{i+1}$.)

Working entropy at a workload of 0 bits is just the **min-entropy** $W_0(p) = H_\infty(p) = -\log_2 p_1$. Min-entropy is also the Renyi entropy of order infinity.

Usually setting the min-entropy to be 128 bits is more than enough for security. However, min-entropy as low a 50 bits might also be acceptable if the working entropy at higher workloads is sufficiently high. For simplicity, however, it is desired to have a min-entropy matching the overall target security level.

In key agreement, Alice can sometimes work out the exact probability distributions of her deliveries. When Alice can find these probability distributions, she may then be able to work out the exact working entropy, and thereby assess security against all types of trial search.

In other settings, Alice may only be able to estimate the probability distribution, in which case she can estimate the working entropy.

D.5 Reduced search: trading workload for entropy

To be completed.

If Betsy has a trial search method with workload w and entropy h, then Betsy also has a trial search method with workload $w - r < w$, simply by reducing the number of trials, which we call **reduced search**. The benefit (for Betsy) of reduced search is less workload. The cost (for Betsy) of reduced for Betsy) is a potentially lower success rate, or equivalently a higher entropy h'.

[2] In the previous work of the author, *Formally Assessing Cryptographic Entropy.*

Two natural ways to reduce the number of trials are as follows. In the **quit-early** reduction method, Betsy uses the same sequence of trials $a_1, a_2, \ldots$, but simply stops earlier, using trials $a_1, \ldots, a_{2^{w-r}}$ instead of $a_1, \ldots, a_{2^w}$. In the **randomized** reduction method, Betsy chooses a uniformly random set of integers $1 \leq t_1, \ldots, t_{2^{w-r}} \leq 2^w$, and then makes trials $a_{t_1}, \ldots, a_{t_{2^{w-r}}}$.

In imitative search, the quit-early and randomized reduction methods have the same effect, because the trials in imitative are already independent of each other.

We say that Betsy's trial search is **sorted** if Betsy chooses trials $a_1, a_2, a_3, \ldots$, which have non-increasing order of individual trial discovery. If Betsy has a sorted trial search with workload w bit and entropy h bits, and the corresponding reduced search is arranged to have workload $w - r$ bits, then the reduced search has entropy h' for

$$h \leq h' \leq h + r.$$

So, assuming a sorted trial search, a workload-to-entropy tradeoff is possible. Also, the use of bits to measure both workload and entropy seems to be justifiable because of these bounds and tradeoffs.

If Betsy's search is not sorted, the randomized reduction method can still be useful.

For example, if Betsy has a sorted trial search with workload of 64 bits and entropy of 64 bits, then Betsy's also has trial search, obtained by reducing the the initial strategy, with workload 0 bits, and entropy between 64 and 128 bits.

Because of this tradeoff, it may seem to prudent to rate the resistance to trial search as the minimum, over all trial search methods, of $w + h$, where w is workload and h is entropy.

However, taking the sum $w + h$ as a security metric fails to take into account the declining costs of computation, and thus workload. Of course, as computation becomes cheaper, key agreement users Alice and Charlie can afford more computation if that improves their security. But the security of past deliveries is only hindered by increases in computing power. A possible resolution of this defect is to incorporate the age a of delivery, normalized in bits (each bit in age representing a doubling of the number of trials computable at a given cost), into the security metric, giving $w + h - a$.

Diversion D.5.1. It sometimes make sense to add workload to working entropy, getting a quantity $w + W_w(p)$. A more precise function is $\log_2 \lfloor 2^w \rfloor + W_w(p)$, which is a non-decreasing function of w. Therefore, it is always at least $W_0(p) = H_\infty(p)$, the min-entropy. For many p, this function's graph has the appearance of a hockey stick, flat initially, and then rising up (with a slope approaching 1).

For small w, a low value of $w + W_w(p)$, such as 64 bits, may be tolerable, but as w increases, the workload becomes large, and the security lifetime or age needs to be accounted because of the effects of falling costs of computation. So, generally, one wants $w + W_w(p)$ to start at least 64 bits, but then rise quickly (as possible) to 128 bits. (Since $W_w(p)$ is non-increasing, $w + W_w(p)$ can only rise as fast as $w + W_0(p)$, so it can reach 128 bits (starting from 64 bits) only by workload $w \geq 64$.).

Perhaps 128 bits is a reasonable maximum feasible workload, even accounting for declining costs of computation. Under that prediction, we can ignore values of $w + W_w(p)$ for $w \geq 128$.

Diversion D.5.2. It would be illuminating to generalize the lower bound (from Lemma D.6.1 on average delivery bit length to working entropy. For example, suppose that average bit length of a delivery had to be at least $W_w(p) + w - 2$ (such as bound would be ideal in terms of its simplicity).

For concreteness, suppose that a probability distribution p of deliveries had $W_0(p) = W_{64} = 64$. This might be considered a secure probability distribution: with tolerably low probability of rapid search, and very low success rate and workload for feasible search. If the working entropy bound $W_w(p) + w - 2$ were in effect, then we could deduce that average bit length delivery would be at least 126 bits.

Diversion D.5.3. Many real-world key agreement schemes, such as Diffie–Hellman, need considerably larger deliveries than those required to resist (exhaustive) trial search attack. For example, in elliptic curve Diffie–Hellman key agreement, the delivery bit lengths are about twice the minimum needed to resist to exhaustive search.

The reason for the larger delivery size is that there exist attacks faster than exhaustive search. In the case of elliptic curve Diffie–Hellman, the Pollard rho attack is possible, whose run-time hinges on the birthday surprise effect, with the number of steps taking the square root of the search space.

D.6 Average bit length of deliveries

In addition to looking at the cost of trial search for the adversary Betsy, we can also look at the cost for Alice and Charlie to resist trial search attacks, such as optimal search.

By the results of previous sections, Betsy's entropy (logarithm of success rate) is closely related to the entropy of Alice's deliveries.

If Alice's delivery has η bits of Shannon entropy, well-known theorems due to Shannon provide a lower bound of η on the average bit-length of a prefix-free encoding of the deliveries. It is also well-known that Shannon entropy is at least min-entropy, and at least the Renyi (collision) entropy. Therefore, for Alice to resist some of the trial search attacks, the average prefix-free encoded delivery bit length must be at least be the entrophy of the attack she wishes to resist.

The following result is therefore redundant relative to such theorems. In this result, Alice takes the naive view of bit strings have a given length, instead of the more robust prefix-free encoding. (Put another way, Alice may contemplate the deliveries before the application of a prefix-free encoding. For example, file system users are able to distinguish two file contents even if is a prefix of the other: the file system resolves by using an end-of-file marker, and so on.) Also, Alice may opt to focus on min-entropy, rather than Renyi or Shannon entropy, knowing that sufficient min-entropy is likely to resist to resist optimal search (even with tradeoffs between workload and entrophy).

Lemma D.6.1 (Redundant special case of Shannon's theorem)**.** *If Alice's delivery has a min-entropy of h bits, then the average bit length of Alice's delivery must be at least $h - 2$ bits.*

Proof. Let $d_1, d_2, d_3, \ldots$ be the possible deliveries of Alice, and let p_i be the probability of delivery d_i. Then $\sum_i p_i = 1$. Arrange the indices such that p_i is non-increasing: so $p_i \geq p_{i+1}$ for all i.

The average bit length of a delivery is at least the average bit length of optimally-compressed delivery. The optimally compressed delivery $d_i = k_2(a_i)$ can be achieved by representing d_i with the non-leading bits of the standard binary representation of the integer i. (So, for example: the most common delivery is represented by the empty string; the fifth most common delivery is represented by the bit string 01.) The optimally compressed delivery d_i thus has bit length $\lfloor \log_2(i) \rfloor$.

The average bit length of an optimally compressed delivery is

$$\delta = \sum_{i=1}^{\infty} p_i \lfloor \log_2(i) \rfloor .$$

Let

$$\Delta(i) = \sum_{j=1}^{i} \lfloor \log_2 j \rfloor .$$

Then

$$\Delta(i) = (i+1)\lfloor \log_2 i \rfloor - 2^{1+\lfloor \log_2 i \rfloor} + 2,$$

which can be shown by induction, or other techniques. Let

$$q_i = \frac{p_i - p_{i+1}}{p_1}.$$

Then

$$\begin{aligned} q_i &\geq 0, \\ \sum_{i=1}^{\infty} q_i &= 1, \\ \sum_{i=1}^{\infty} i q_i &= 1/p_1, \\ \sum_{i=1}^{\infty} q_i \Delta(i) &= \delta/p_1, \end{aligned}$$

by various telescoping arguments.

If we extend the domain of Δ to real numbers by making it a piece-wise linear function, then Δ is a convex function, obeying the formula above for reals above 1. By convexity of Δ, Jensen's inequality can be applied to get

$$\delta/p_1 = \sum_{i=1}^{\infty} q_i \Delta(i) \geq \Delta\left(\sum_{i=1}^{\infty}(q_i)i\right) = \Delta(1/p_1).$$

Substituting $p_1 = 2^{-h}$ and $\lfloor \log_2 1/p_1 \rfloor = \lfloor h \rfloor$ into the formula for Δ gives:

$$\delta \geq (1 + 2^{-h})\lfloor h \rfloor - 2^{-h+1+\lfloor h \rfloor} + 2^{1-h} \geq \lfloor h \rfloor - 2^{1-(h-\lfloor h \rfloor)} \geq h - 2,$$

which is what we sought to prove. □

Diversion D.6.1. The lower bound $\delta \geq h - 2$ in the lemma is sometimes quite tight. For example, let m be an integer, and let $p_i = 1/(2^m - 1)$ for $1 \leq i \leq 2^m - 1$, so that p represents a uniform distribution over $2^m - 1$ values. With $m = 10$, we have $h \approx 9.99859$ and $\delta \approx 8.00978$.

Diversion D.6.2. The lower bound $\delta \geq h - 2$ in the lemma is sometimes quite loose. For example, if $p_1 = 0.99$ and $p_i = 2^{-128}/100$ for $2 \leq i \leq 1 + 2^{128}$, then average bit-length is $\delta \approx 128$ but min-entropy $h \approx 0.0144$.

Appendix E

Previous work

The general ideas in this report are so simple, that much previous has been done on them.

Due to the great generality of the semigroups, many of the specific details are to be found in previous work scattered among a wide variety of sources, only a few of which have informed this report. The hope is eventually trace back through the literature to credit all the appropriate discoverers.

E.1 A brief history of key agreement

Key agreement was introduced by Diffie, Hellman and Merkle when they introduced public-key cryptography. Key agreement is a goal for two parties, who initially have no common cryptographic key, to agree on a key, by only exchanging public information (and never actually meeting in secret).

Diversion E.1.1. Terms with broader meaning than **key agreement** include:

- **key distribution**,
- **key exchange**,
- **key establishment**, and
- **key encapsulation**.

All four alternatives include schemes that are not key agreement (in the sense used in this report).

Diffie–Hellman key agreement, introduced soon after by Diffie and Hellman, is a specific **scheme**[1] for key agreement. The public values exchanged are computed using the mathematical operation of modular exponentiation, and further modular exponentiation is used to compute the key.

Diversion E.1.2. This report uses notes like this to discuss topics that can be seen as peripheral to the essential topics of this report: and can often be treated as optional.

Subsequently, it was realized that Diffie–Hellman key agreement very naturally generalizes to use any mathematical **group**. Recall that a group includes a set equipped with an associative operation, written as multiplication by default. Diffie–Hellman key agreement use group exponentiation, which means repeated group multiplications. Not all groups are secure if used in Diffie–Hellman key agreement. Generally, security requires that a computational problem, the **discrete logarithm** problem (essentially the opposite of exponentiation) is difficult in the group.

Koblitz and Miller realized that **elliptic curves** (from algebraic geometry), seem to provide groups with a difficult discrete logarithm problem, yielding **elliptic curve cryptography**, often abbreviated to **ECC**. The most relevant ECC scheme for this report is elliptic curve Diffie–Hellman key agreement (ECDH).

By 2018, elliptic curve Diffie–Hellman key agreement has become widely used on the Internet: in the Transport Layer Security (TLS) protocol, which is used, for example, to protect communication between web browsers and secure web sites. The feature of **forward secrecy**, possessed by key agreement, is one of the main reasons that Diffie–Hellman has replaced previously used public-key algorithms, such RSA public-key encryption (which lacks forward secrecy).

Meanwhile, other types of key agreement have been considered. For example, **super-singular-isogeny key exchange** (SIKE, also known as SIDH), uses maps between elliptic curve groups, but does not exponentiation the way DH key agreement does.

Alternative key agreement schemes are considered because of their potential to resist quantum computers, other types of attacks on Diffie–Hellman key agreement.

[1] Emphasis, as in **scheme**, introduces terms used with specialized meanings (usually narrowing of their meanings outside of this report): with formal definitions for the most important terms.

E.2 On associative key agreement

Some very similar previous work in cryptography is Ding–Lin–Xie key agreement [DXL12], which briefly mentions associativity being useful to achieve key agreement, very much in the manner of associative key agreement. Specifically, they mention associativity of matrix multiplication. They thus arguably give the impression that associative key agreement is generally not feasible, because their only example is matrices, where matrix division defeats the security.

Ding, Lin and Xie recognize the insecurity of using matrices in associative key agreement, but instead seeking more secure semigroups (as in this report), they introduce errors and error correction to arrive a probabilistic key agreement scheme. Alice and Charlie agree with high probability. This error correction approach formally excludes an associative axiom that holds for all elements of the semigroup. So, their work does not give an exact semigroup.

Also, Ding, Lin and Xie, do make the connection between Diffie–Hellman key agreement and associativity.

I would not be surprised, if even earlier work describe the general idea of associative key agreement. The rest of this section consists of further updates, in the order that I learn of them.

Rabi and Sherman [RS93] describe using an associative binary operation for a key agreement protocol, which is essentially this report calls multiplicative key agreement.

Berenstein and Chernyak [BC04][2] describe a key establishment scheme based on a commuting double action, and then give left and right semigroup multiplication as one example of a commuting double action. In some sense, they have described associative key agreement, but it is by way of an intermediate notion. They do not seem to emphasize the relevance of division, and the wide variability of semigroups available.

E.3 On semigroups in crypto

I found several papers on cryptography that apply semigroups to cryptography, but not through the idea of an associative key agreement scheme (the topic of this report).

[2] I thank US patent examiner Y. Bayou for alerting me to [BC04].

These other applications of semigroups included the following.

1. Generalizing the group of public keys in Diffie–Hellman (such as an elliptic curve group, or the multiplicative group of a finite field), by allowing the group to become a semigroup. The security then depends on the discrete logarithm problem in the semigroup. I believe that there is a result of LaFlamme and others [**?**] showing the Shor's algorithm can be used to solve the discrete logarithm in finite semigroups. In infinite semigroups, I think somebody has observed that the discrete logarithm can be often be deduced by the data size.

2. Generalizing the group of the private keys in Diffie–Hellman (under multiplication) to semigroup with an action on a set, in the sense that private key acts on public keys in Diffie–Hellman key agreement. One example is the so-called hard homogeneous spaces (introduced by Couveignes???).

3. Generalizing the groups used in other algebraic cryptographic schemes, such as those based on braid groups. The hard problems involved in these schemes are not division, not discrete logarithm, but yet other problems, such as the conjugacy problem. (There may be some overlap with cryptography using semigroup actions).

E.4 On division algorithms

Many of the general ideas for division algorithms are common knowledge in the discipline of mathematics. This report has tried to re-construct some detail about division algorithms from this common knowledge.

For more specific details of division algorithms, I referred to Knuth [Knu98] for information on some division algorithms. I referred to Conway [Con76] and Lenstra [Len97, Len77], for some division algorithms involving combinatorial games. I referred to Cox, Little, O'Shea [CLO91] for some division algorithms related to polynomial rings and their ideals.

E.5 On semigroups themselves

The theory of semigroups extends the classical and fundamental theory of groups, rings, and fields (see Lang [Lan93], for example).

The wider theory of semigroups was unfamiliar to me before 2017. To learn some of the basics of this wider theory, I referred extensively to Wikipedia (especially wikipedia.org/wiki/semigroup). For greater details, I referred to some books (borrowed from the U. of Waterloo library), including: Clifford and Preston [CP64a, CP64b]; Liapin [Lia63]; Howie [How76]; Hall, Jones and Meakin [HJM91]; Petrich [Pet84]; and Okninski [Okn91]. I also referred to online works: the Online Encyclopedia of Integer Sequence oeis.org; and the thesis of Distler [Dis10]. Large parts of report were written independently of the references (in fact, after I returned these books), so any flaws in this report were almost certainly added by me (inadvertently).

Bibliography

[BC04] A. Berenstein and L. Chernyak. *Geometric key establishment.* In *Canadian Mathematical Society Conference*, pp. 1–19. 2004. 1.4, 2.11, E.2, 2

[Con76] J. H. Conway. *On Numbers and Games.* Academic Press, 2nd edn., 1976. E.4

[Cou06] J.-M. Couveignes. *Hard homogeneous spaces.* ePrint 2006/291, International Association for Cryptologic Research, ia.cr/2006/291, 2006. C.2

[CKS08] D. Cash, E. Kiltz and V. Shoup. *The twin Diffie–Hellman problem and applications.* ePrint 2008/067, International Association for Cryptologic Research, ia.cr/2008/067, 2008. 2.23.1, A.3, C.20.5.10

[CLO91] D. Cox, J. Little and D. O'Shea. *Ideals, Varieties, and Algorithms.* Undergraduate Texts in Mathematics. Springer, 2nd edn., 1991. C.22.3, E.4

[CP64a] A. H. Clifford and G. B. Preston. *Algebraic theory of semigroups*, No. 7 in Mathematical surveys, vol. 1. American Mathematical Society, 2nd edn., 1964. E.5

[CP64b] ———. *Algebraic theory of semigroups*, No. 7 in Mathematical surveys, vol. 2. American Mathematical Society, 1964. E.5

[Dis10] A. Distler. *Classification and enumeration of finite semigroups.* Ph.D. thesis, University of St. Andrews, https://research-repository.st-andrews.ac.uk/, 2010. B.3.35, C.1, E.5

[DXL12] J. Ding, X. Xie and X. Lin. *A simple provably secure key exchange scheme based on the learning with errors problem.* ePrint 2012/688, International Association for Cryptologic Research, `ia.cr/2012/688`, 2012. E.2

[Feo17] L. D. Feo. *Mathematics of isogeny based cryptography.* Tech. rep., U. de Versailles, `https://arxiv.org/pdf/1711.04062.pdf`, 2017. 1

[How76] J. M. Howie. *Introduction to semigroup theory.* No. 7 in London Mathematical Society monographs. Academic Press, 1976. E.5

[Hut81] H. C. Hutchins. *Examples of Commutative Rings.* Polygonal Publishing House, 1981. B.12

[HJM91] T. E. Hall, P. R. Jones and J. C. Meakin (eds.). *Monash Conference on Semigroup Theory: in honour of G.B. Preston.* World Scientific, 1991. E.5

[HW15] P. Hrubeš and A. Wigderson. *Non-commutative arithmetic circuits with division.* Theory of Computing, **11**(14):357–393, 2015. doi:10.4086/toc.2015.v011a014. 5.16, 5.16.1

[Knu98] D. E. Knuth. *The Art of Computer Programming*, vol. 2: Seminumerical Algorithms. Addison–Wesley, 3rd edn., 1998. 5.14.1, 5.15.1, 5.15.2, E.4

[Kob98] N. Koblitz. *Algebraic Aspects of Cryptography.* No. 3 in Algorithms adn Computation in Mathematics. Springer, 1998. B.8.30

[Lan93] S. Lang. *Algebra.* Addison–Wesley, 3rd edn., 1993. E.5

[Len77] H. W. Lenstra, Jr. *On the algebraic closure of two.* In *Proceedings of Koninklijke Nederlandse Akademia*, A, vol. 80, pp. 379–396. 1977. E.4

[Len97] ———. *Nim multiplication.* Seminaire de Theorie des Nombres de Bordeaux, pp. 1–23, 1997. E.4

[Lia63] E. S. Liapin. *Semigroups.* No. 3 in Translations of mathematical monographs. American Mathemtical Society, Translated by A. A. Brown and others, 1963. E.5

[Okn91] J. Okninski. *Semigroup algebras*. No. 138 in Monographs and textbooks in pure and applied mathematics. M . Dekker, 1991. E.5

[Pet84] M. Petrich. *Inverse semigroups*. Pure and applied mathematics. Wiley, 1984. B.7.10, B.7.14, E.5

[RS93] M. Rabi and A. T. Sherman. *Associative one-way functions: A new paradigm for secret-key agreement and digital signatures*. Tech. Rep. CS-TR-3183/UMIACS-TR-93-124, University of Maryland, 1993. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.118.6837&rep=rep 1.4, 2.11, E.2