A Subexponential Quantum Algorithm for the Semidirect Discrete Logarithm Problem

Christopher Battarbee[1], Delaram Kahrobaei[1,2,3,4], Ludovic Perret[5], and Siamak F. Shahandashti[1]

[1] Department of Computer Science, University of York, UK
[2] Departments of Computer Science and Mathematics, Queens College, City University of New York, USA
[3] Initiative for the Theoretical Sciences, Graduate Center, City University of New York, USA
[4] Department of Computer Science and Engineering, Tandon School of Engineering, New York University, USA
[5] Sorbonne University, CNRS, LIP6, PolSys, Paris, France

Abstract. *Group-based* cryptography is a relatively unexplored family in post-quantum cryptography, and the so-called Semidirect Discrete Logarithm Problem (SDLP) is one of its most central problems. However, the complexity of SDLP and its relationship to more well-known hardness problems, particularly with respect to its security against quantum adversaries, has not been well understood and was a significant open problem for researchers in this area. In this paper we give the first dedicated security analysis of SDLP. In particular, we provide a connection between SDLP and group actions, a context in which quantum subexponential algorithms are known to apply. We are therefore able to construct a subexponential quantum algorithm for solving SDLP, thereby classifying the complexity of SDLP and its relation to known computational problems.

Introduction

The goal of Post-Quantum Cryptography (PQC) is to design cryptosystems which are secure against classical and quantum adversaries. A topic of fundamental research for decades, the status of PQC drastically changed with the NIST PQC standardization process [10].

In July 2022, after five years and three rounds of selection, NIST selected a first set of PQC standards for Key-Encapsulation Mechanism (KEM) and Digital Signature Scheme (DSS) protocols, based on lattices and hash functions. The standardization process is still ongoing with a fourth round for KEM and a new NIST call for post-quantum DSS in 2023. Recent attacks [5, 32, 1] against round-3 multivariate signature schemes, Rainbow [5] and G*e*MSS [7], as well as the cryptanalysis of round-4 isogeny based KEM SIKE [8, 24], emphasise the need to continue the cryptanalysis effort in PQC as well as to increase the diversity in the potential post-quantum hard problems.

A relatively unexplored family of such problems come from *group-based* cryptography (see [20]). In particular we are interested in the so-called Semidirect Discrete Logarithm Problem (`SDLP`), which initially appears in the 2013 work [16] of Habeeb et al. Roughly speaking, we generalise the standard notion of group exponentiation by employing products of the form $\phi^{x-1}(g) \cdot \ldots \cdot \phi(g) \cdot g$, where g is an element of a (semi)group, ϕ is an endomorphism and $x \in \mathbb{N}$ is a positive integer. Our task in `SDLP` is to recover the integer x given the pair g, ϕ and the value $\phi^{x-1}(g) \cdot \ldots \cdot \phi(g) \cdot g$. It turns out that products of this form have enough structure to be cryptographically useful, in a sense we will expand upon later - in particular, protocols based on `SDLP` are plausibly post-quantum, since there is no known reduction of `SDLP` to a Hidden Subgroup Problem.

By far the most studied such protocol is known as Semidirect Product Key Exchange (`SPDKE`), originally proposed in [16] (note that this is the same work in which `SDLP` first appears). It is a Diffie–Hellman-like key exchange protocol in which products of the form $\phi^{x-1}(g) \cdot \ldots \cdot \phi(g) \cdot g$ are exchanged between two parties in such a way as to allow both parties to recover the same shared key. Clearly, the security of `SPDKE` and the difficulty of `SDLP` are heavily related - in particular, an adversary able to solve `SDLP` is also able to break `SPDKE`.

There is therefore motivation to analyse the difficulty of `SDLP`. However, the complexity of `SDLP` and its relationship to more well-known hardness problems, particularly with respect to its security against quantum adversaries, has not been well understood and was a significant open problem for researchers in this area. In this paper we provide the first dedicated analysis of `SDLP`, obtaining two key contributions. First, we demonstrate that a subset of all possible products of the form $\phi^{x-1}(g) \cdot \ldots \cdot \phi(g) \cdot g$ is a set upon which a finite abelian group acts; in other words, that `SPDKE` is, modulo some context-specific technicality, a variant of the group action-based key exchange schemes originally proposed by Couveignes [14]. In particular, solving `SDLP` can be translated into a problem with respect to a group action. This surprising connection provides a sharper classification of `SPDKE` than was previously known, and allows us to derive our second contribution, an application of known tools that gives a quantum algorithm for solving `SDLP`. The algorithm runs in subexponential time $2^{\mathcal{O}(\sqrt{\log p})}$, where p is the security parameter.

Related Work

Examples of concrete proposals for `SPDKE` can be found in [16, 15, 21, 28, 29]; respective cryptanalyses can be found in [27, 19, 6, 25, 3, 4]. This body of work proceeds more or less chronologically, in that proposed platforms are a response to cryptanalysis addressing a weakness in an earlier version. As a brief summary: the platform proposed in the first version of the scheme [16] is the multiplicative semigroup of a matrix algebra formed of 3×3 matrices with entries in a group ring. It was pointed out in [27] that the unused addition operation of the matrix algebra was actually a vulnerability allowing for complete shared key recovery, and that moreover any group with a sufficiently low-dimensional representation as a matrix algebra would be vulnerable to a similar kind of attack. In response, in [21] certain classes of p-groups are proposed, since they admit only extremely high-dimensional representations. Another method of counteracting this linear vulnerability is to mix operations: in [28] the matrix algebra $M_3(\mathbb{Z}_p)$ is employed for some prime p in such a way that both matrix multiplication and matrix addition are called upon. It turns out, however, that this mixing of operations opens the scheme up to a different type of attack [6, 3] based on the so-called 'telescoping equality'. It appears at time of writing that one can get around this vulnerability (as argued in [4]) by removing some structure; in particular, using matrices over a semiring rather than a ring. An earlier attempt at utilising a semiring is made in [15], though this particular case turned out to admit full shared key recovery due to a partial order allowing for more efficient search algorithms. A more detailed survey of the back-and-forth on this topic can be found in [2].

Despite the relationship between `SPDKE` and `SDLP`, none of the works discussed above provide an analysis of `SDLP`. Indeed, the general direction of research in this area has been either to achieve shared key recovery by exploiting some underlying linearity of a platform (semi)group, or to find examples of (semi)groups with sufficiently lax structure to render these attacks less powerful. In particular, none of the cryptanalyses in this area solve `SDLP`.

Ideas much closer to the spirit of our work appear in papers that, at first glance, appear unrelated to `SPDKE` and `SDLP`. Our results are achieved in part by careful synthesis of the techniques in the two papers [11, 12]: since the set of all products of the form $\phi^{x-1}(g) \cdot \ldots \cdot \phi(g) \cdot g$ admits some similarity to that of a monogenic semigroup, we can adapt some ideas from a quantum algorithm in [12] that solves the Semigroup Discrete Logarithm Problem. However, in our setting we are lacking some key structure that allows the direct application of [12]. The full algorithm is constructed by adapting ideas in [11], allowing us to show the important quantum algorithms of Kuperberg [23] and Regev [30] can be used to solve `SDLP`.

Finally, we note that the connection to group actions alluded to above allows the application of a recent landmark result of Montgomery and Zhandry [26]. The implications of this result on our work are discussed in Section 3.3.

Organisation of the Paper and Main Results

The construction of the algorithm claimed in the title is, from a high-level perspective, achieved by two reduction proofs followed by an application of known algorithms. With this in mind, we make the following contributions.

Section 1. We start with preliminaries in which the necessary background is reviewed. In particular we give a brief discussion of the relevant algebraic objects, a short note on quantum computation, a full description of `SPDKE` (including the motivation for choosing a particular framework to operate in, which will give us a security parameter), and a kind of glossary of computational problems.

Section 2. It will be immediately convenient to write $s(g, \phi, x)$ for $\phi^{x-1}(g) \cdot ... \cdot g$, not only for clarity of notation but to aid the crucial shift in perspective offered in this paper. Armed with this notation our first task is to study the set of possible exponents $\{s(g, \phi, i) : i \in \mathbb{N}\}$. In Theorem 1 we deduce, borrowing from some standard ideas in semigroup theory, that this set is finite and has the form $\{g, ..., s(g, \phi, n), ..., s(g, \phi, n + r - 1)\}$ where the integers n, r are a function of the choice of (g, ϕ). The main result of this section is Theorem 4: an abelian group acts freely and transitively on the set $\{s(g, \phi, n), ..., s(g, \phi, n + r - 1)\}$. This set is called the cycle of g, ϕ and is denoted by the calligraphic letter $\mathcal{C}$. In particular, we show the following:

Theorem. *Fix $(g, \phi) \in G \times End(G)$ and let n, r be the index and period corresponding to g, ϕ. Moreover, let $\mathcal{C}$ be the corresponding cycle of size r. The abelian group $\mathbb{N}_r$ acts freely and transitively on $\mathcal{C}$.*

where $\mathbb{N}_r$ is a cyclic group defined in such a way as to ensure that the function s is not required to take negative integer arguments.

Section 3. If the set of exponents with respect to g, ϕ consisted entirely of the cycle we would immediately have a reduction to the Group Action Discrete Logarithm Problem (`GADLP`), also referred to as *Group Action DLog* in [26], or the *Parallelisation Problem* in [14]. Roughly speaking, since we know that an abelian group acts on the cycle, in this case the exponent x to be recovered is precisely the group element acting on g to give $s(g, \phi, x)$. This is not, however, generally the case, and to proceed it will be necessary to extract the pair n, r from the base values g, ϕ. In Theorem 5 we show that one can achieve this in efficient quantum time by using canonical quantum period-finding methods - moreover, for certain classes of semigroup this algorithm succeeds with constant probability as the security parameter grows. We can therefore deduce in Theorem 6 that one can solve `SDLP` efficiently given access to a `GADLP` oracle; or, if $s(g, \phi, x)$ is not in the cycle of g, ϕ, by invoking a classical procedure exploiting the knowledge of n, r. Indeed, we show the following:

Theorem. *Let $\{G_p\}_p$ be an easy family of semigroups, and fix p. Algorithm 2 solves `SDLP` with respect to a pair $(g, \phi) \in G_p \times End(G_p)$ given access to a `GADLP` oracle for the group action $(\mathbb{N}_{r_{g,\phi}}, \mathcal{C}_{g,\phi}, \circledast)$. The algorithm runs in time $\mathcal{O}(N(p)^2 \log N(p) + (\log N(p))^3)$, makes at most a single query to the `GADLP` oracle, and succeeds with probability $\Omega(1/(n(p) + r(p)))$.*

Clearly, many of the requisite notions in this statement have not yet been defined. Roughly speaking, an easy family of semigroups is a family of semigroups parameterised by p such that each of the functions taking p as an argument grows polynomially in p.

Section 4. In order to give a full description and complexity analysis of the algorithm it remains to examine the state of the art for solving `GADLP`. It is reasonably well-known (see [31], [11]) that `GADLP` reduces to the Abelian Hidden Shift Problem if the action is free and transitive, though we provide a context-specific reduction in Theorem 7. There are two popular choices to solve this problem: an algorithm due to Kuperberg [23] and another due to Regev [30], each of which has trade-offs with respect to time and space complexity. Finally, the full algorithm is given in Theorem 10, though we are essentially assembling the components we have developed throughout the rest of the paper. This main result is the following:

Theorem. *Let* $\{G_p\}_p$ *be an easy family of semigroups, and fix* p*. For any pair* $(g, \phi) \in G_p \times End(G_p)$*, there is a quantum algorithm solving* `SDLP` *with respect to* (g, ϕ) *with time and query complexity* $2^{\mathcal{O}(\sqrt{\log p})}$.

which proves the claim of a quantum subexponential algorithm for `SDLP` given in the title.

1 Preliminaries

1.1 Notation

One need only be familiar with the standard $\mathcal{O}()$ and $\Omega()$ notations. Various bespoke notations are introduced throughout the course of the paper, but these will be defined in due course.

1.2 Background Mathematics

We recall a number of group-theoretic notions used throughout this paper.
Recall that a group for which one is not guaranteed to have inverse with respect to the group operation is a *semigroup*. Writing the operation multipicatively, the semigroups G we are interested in all have an element 1 such that $1 \cdot g = g = g \cdot 1$ for each $g \in G$ - such a semigroup is also called a *monoid*. The constructions that follow are defined for both groups and semigroups - to reflect this scenario we will use the catch-all notation (semi)groups.
We will deal with both abelian groups and non-abelian (semi)groups in this paper; that is, for a non-abelian (semi)group G one cannot expect that $g \cdot h = h \cdot g$ for all $g, h \in G$. For the sake of clarity we will write abelian groups additively. In this case, the operation $g + h$ commutes, and we require an inverse for every element. Note also that the identity is written as 0 in this case.
Consider a function from G to itself, say ϕ. If ϕ preserves multiplication - that is, $\phi(g \cdot h) = \phi(g) \cdot \phi(h)$ for each $g, h \in G$ - we call it an *endomorphism*. Certainly we can compose these functions according to the usual notion, and indeed it is standard that the set of all endomorphisms under function composition defines a semigroup. Since we allow for (and in some cases require) that the endomorphisms are not invertible, we have a semigroup rather than a full group. In particular, every finite semigroup G immediately induces an *endomorphism semigroup*, denoted $End(G)$.

An important, and in this context frequently invoked source of semigroups come from *matrix algebras*. The set of square matrices of fixed size with entries in some ring R forms an R-module, since we can add matrices together and scale each entry of a matrix by some $r \in R$. The necessary distributivity properties are inherited from the properties of R. However, unlike in a usual R-module, we can also *multiply* elements just by defining multiplication to be the usual notion of matrix multiplication. The resulting matrix algebra is denoted $M_n(R)$, where $n \in \mathbb{N}$ is the fixed size of matrix, and R is the underlying ring. Indeed, consider a matrix algebra under only the multiplication operation. It is again clear that this object is a semigroup; we would have a full group if every matrix was invertible, but of course this is not true. The all-zero matrix, for example, has no multiplicative inverse. A matrix algebra considered only under its multiplication, therefore, is a useful source for concrete examples of semigroups.
It will be useful for us to build a new (semi)group from an existing (semi)group. One way of doing this is via a structure called the *holomorph*. Let G be a (semi)group and $End(G)$ its endomorphism semigroup. The holomorph is the set $G \times End(G)$ equipped with multiplication

$$(g, \phi) \cdot (g', \phi') = (\phi'(g) \cdot g', \phi \circ \phi')$$

where $\circ$ refers to function composition. In fact, the holomorph is itself a special case of the semidirect product, hence the terminology.

Group Actions. A key idea for us will be that of a group action, and in particular a commutative group action. Roughly speaking such an object allows one to map elements of a set to each other in a cryptographically useful fashion, but in a less structured manner than in more classical settings. More formally:

Definition 1 (Commutative Group Action). *Let G be a finite abelian group and X be a finite set. Consider a function from $G \times X \to X$, written by convention as $g \star x$, with the following properties:*

1. $1 \star x = x$
2. $(g + h) \star x = g \star (h \star x)$

The tuple $(G, X, \star)$ is a commutative group action. If only the identity fixes an arbitrary element of X the action is free, and if for any $x, y \in X$ there is a g such that $g \star x = y$ the action is transitive.

The group action defined in this paper is commutative, so we will sometimes just write 'group action' to mean a commutative group action. It will remain for us, however, to prove that this action is free and transitive. If the action is indeed free and transitive, it follows that for any $x \in X$, all $y \in X$ are such that there exists a unique $g \in G$ with $g \star x = y$. Borrowing notation from Couveignes [14], it will sometimes be convenient for us to write $\delta(y, x)$ to denote this value.

1.3 Quantum Computation

In order to present our quantum algorithm for `SDLP` (Section 3), the reader needs only be familiar with standard quantum tools, presented for example in [17]. We give a brief summary of the required notions below.

Recall that n qubits can be represented by the complex vector space H_{2^n}, where the basis states are exactly the n-fold tensor products of basis states of H_2. An ordered system of m qubits is called a *quantum register of length* n, and the basis states are sometimes written $\{|i\rangle : 0 \leq i < 2^m\}$ by identifying i with its binary representation.

What happens when we observe these multiple-qubit states? In a two-qubit example, consider the state $\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$. Observing the whole system gives (with respect to binary representation) $0, 1, 2$ or 3 with respective probabilities $\alpha, \beta, \gamma, \delta$. We may, however, choose to observe only one of the qubits, whence a crucial idea arises from the notion of *entanglement*. We say a state $\mathbf{z} \in H_4$ is *entangled* if it cannot be written as the tensor product of two H_2 states; one such state is $1/\sqrt{2}(|00\rangle + |11\rangle)$. In this case, observing the second qubit gives 0 or 1 with probability 1/2 - but if 1 is observed, the first qubit cannot give an observation of 0. In particular, an observation of one qubit in an entangled state affects the state of the other.

Of course, multiple qubits can be entangled. An important generalisation of the ideas above is the following: suppose integers $\{0, ..., M-1\}$ can be represented by a quantum register of length l for some $l, M \in \mathbb{N}$, and moreover that a function f on $\{0, ..., M-1\}$ is such that $\{f(0), ..., f(M-1)\}$ can be represented similarly. A state of the form

$$\frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |j\rangle |f(j)\rangle$$

is such that when observation of the second register gives some $Y \in \{f(0), ..., f(M-1)\}$, the first register is left in superposition

$$\frac{1}{\sqrt{L}} \sum_{j:f(j)=Y} |j\rangle$$

where L is the number of $j \in \{0, ..., M-1\}$ such that $f(j) = Y$. The factor $1/\sqrt{L}$ is to normalise the probabilities, ensuring the state is a unit vector.

Finally, we describe how states of the above form can be efficiently prepared. Starting with the state $|0\rangle$, a computational basis state in an l-qubit register, we can apply a special type of unitary matrix called a *Hadamard gate* which gives the uniform superposition $1/\sqrt{M}\sum_j |j\rangle$. The tensor product is distributive, so one has

$$\left(\frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |j\rangle\right) \otimes |0\rangle = \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |j\rangle \otimes |0\rangle$$

Suppose some function f from $\{0, ..., m-1\}$ is such that $f(\{0, ..., m-1\})$ can be represented by l qubits. The following transformation can be efficiently carried out:

$$\frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |j\rangle |0\rangle \to \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |j\rangle |f(j)\rangle$$

so one can efficiently prepare states of the required form.

1.4 Semidirect Product Key Exchange

We here define in full `SPDKE`. One verifies by induction that holomorph exponentiation takes the form

$$(g, \phi)^x = (\phi^{x-1}(g) \cdot \ldots \phi(g) \cdot g, \phi^x)$$

where ϕ^x denoted the endomorphism ϕ composed with itself x times. Note that this operation involves multiplying (semi)group elements, endomorphisms, and applying an endomorphism to a semigroup element. If all these operations are efficient, the holomorph exponentiation is efficient since one can apply standard square-and-multiply techniques.
The central idea of `SPDKE` is to use products of these form as a generalisation of Diffie–Helman Key-Exchange. Suppose N is the number of all possible distinct holomorph exponents - there are finitely many - then the protocol works as follows:

1. Suppose Alice and Bob agree on a public (semi)group G and hence the integer N, as well as a group element g and endomorphism of G, say ϕ.
2. Alice picks a secret integer x uniformly at random from $\{1, ..., N\}$, and calculates the holomorph exponent $(g, \phi)^x = (A, \phi^x)$. She sends **only** A to Bob.
3. Bob similarly calculates (B, ϕ^y) corresponding to a random, private integer y, and sends only B to Alice.
4. With her private automorphism ϕ^x Alice can now calculate her key as the group element $K_A = \phi^x(B) \cdot A$; Bob similarly calculates his key $K_B = \phi^y(A) \cdot B$.

We have

$$\begin{aligned}
\phi^x(B) \cdot A &= \phi^x(\phi^{y-1}(g) \cdot \ldots \cdot g) \cdot (\phi^{x-1}(g) \cdot \ldots \cdot g) \\
&= (\phi^{x+y-1} \cdot \ldots \cdot \phi^x(g)) \cdot (\phi^{x-1}(g) \cdot \ldots \cdot g) \\
&= (\phi^{x+y-1} \cdot \ldots \cdot \phi^y(g)) \cdot (\phi^{y-1}(g) \cdot \ldots \cdot g) \\
&= \phi^y(A) \cdot B
\end{aligned}$$

so $K := K_A = K_B$. Note that $A \cdot B \neq K$ as a consequence of our insistence that the group operation is non-commutative.
Writing these products in full will quickly become rather cumbersome. We therefore introduce some non-standard notation, which is useful both for convenience of exposition and the required shift in perspective we will introduce in this paper.

Definition 2. *Let G be a finite, non-commutative (semi)group, $g \in G$, and $\phi \in End(G)$. We define the following function:*

$$s : G \times End(G) \times \mathbb{N} \to G$$
$$(g, \phi, x) \mapsto \phi^{x-1}(g) \cdot \ldots \cdot \phi(g) \cdot g$$

Notice that when g, ϕ are fixed - as in the case of the key exchange - the function s is really only taking integer arguments, analogously to the standard notion of group exponentiation. Indeed, a passive adversary observing a round of SPDKE has access to the values $s(g, \phi, x)$ and $s(g, \phi, y)$ - in order to recover the shared key $s(g, \phi, x+y)$ one strategy they might adopt is to recover the private integers x, y from $s(g, \phi, x), s(g, \phi, y)$ to allow calculation of $s(g, \phi, x + y)$. In short, the security of SPDKE is clearly in some sense related to the Semidirect Discrete Logarithm Problem alluded to in the introduction. We shall have much more to say about this later on.

1.5 Efficiency Considerations

The works discussed in the introduction, as well as the contents of the more comprehensive survey [2], highlight that every extant proposal of a platform for SPDKE suggests for use some variety of matrix algebra. In particular, insofar as parameters are recommended, the convention is to fix a matrix size - usually 3 - and adjust the size of an underlying ring in order to increase security. One very good reason for this is that each scheme requires a user to repeatedly perform matrix multiplication, the complexity of which scales exponentially with the size of the matrix. Table 1 gives examples of platforms over 3×3 matrices, the size of the platform, and the variable that can be considered the security parameter[6].

Table 1. Growth of Proposed Platforms

Proposed Platform	Size of Platform	Security Parameter
$M_3(G[R])$	$\|R\|^{9\|G\|}$	$\|R\|$
Certain classes of p-group	Polynomial in prime p	Prime p
$M_3(\mathbb{Z}_p)$	p^9	p

We point this out in order to justify the following assumptions. Having defined SPDKE above relative to some semigroup G and its endomorphism semigroup $End(G)$, we will assume that each such semigroup is one of a family of semigroups $\{G_p\}_p$, where the family $\{G_p\}_p$ is parameterised by a security parameter p. Note that this immediately induces a family of endomorphism semigroups $\{End(G_p)\}_p$, so we can talk about pairs (g, ϕ) from the set $G_p \times End(G_p)$ for

[6] Note here that $|R|$ is chosen as the parameter for reasons of efficiency of representation.

each p. Each semigroup in this family has size polynomial in p, and estimates of security and complexity will be given in terms of p. Moreover, since the size of the matrices in question are assumed to be fixed, each matrix multiplication can be carried out in a number of operations bounded above by some constant independently of p.

We also observe that the endomorphisms suggested for use with SPDKE typically involve multiplication by one or more auxiliary matrices, which by the above requires a number of operations bounded above by some constant independently of p. In particular, for a family of semigroups $\{G_p\}_p$ we may assume for a particular (semi)group G_p if $(g,\phi) \in G_p \times End(G_p)$ the group element $\phi(g)$ has the form $A \cdot g \cdot B$, where $A, B \in G$ are fixed (one easily verifies that this indeed defines an endomorphism). We can therefore provide some insight on the efficiency of calculating $s(g,\phi,x)$ for some $x \in \mathbb{N}$ - we calculate the element $(g,\phi)^x$ in the holomorph, which by the standard square-and-multiply technique requires about $\mathcal{O}(\log x)$ holomorph multiplications. Since each holomorph multiplication is by assumption a matter of computing a fixed number of matrix multiplications, we get that holomorph multiplication can be carried out in a number of operations bounded above by some constant independently of p. We conclude that calculating $s(g,\phi,x)$ takes $\mathcal{O}(\log x)$ operations.

1.6 Computational Problems

We have already alluded to some of the hard problems to be found in this paper. Here we give full definitions of all of them to serve as a kind of 'glossary' section.

Definition 3 (Semidirect Discrete Logarithm Problem). *Given a public (semi)group G, its public endomorphism semigroup $End(G)$ and a public pair $(g,\phi) \in G \times End(G)$, let N be the size of the set $\{s(g,\phi,i) : i \in \mathbb{N}\}$. Choose x from $\{1,...,N\}$ uniformly at random, calculate $s(g,\phi,x)$ and create the pair $((g,\phi), s(g,\phi,x))$. The Semidirect Discrete Logarithm Problem (*SDLP*) with respect to (g,ϕ) is to recover the integer x given the pair (g,ϕ) and $s(g,\phi,x)$.*

Definition 4 (Semidirect Computational Diffie–Hellman). *Given a public (semi)group G, its public endomorphism semigroup $End(G)$ and a public pair $(g,\phi) \in G \times End(G)$, let N be the size of the set $\{s(g,\phi,i) : i \in \mathbb{N}\}$. Choose x, y from $\{1,...,N\}$ uniformly at random, compute the values $(s(g,\phi,x), s(g,\phi,y), s(g,\phi,x+y))$, and create the tuple $((g,\phi), s(g,\phi,x), s(g,\phi,y))$. The Semidirect Computational Diffie–Hellman problem (*SCDH*) with respect to (g,ϕ) is to recover the value $s(g,\phi,x+y)$.*

The cognoscenti will notice that this version of a Computational Diffie–Hellman-type problem is defined slightly differently than equivalent variants found in this area. We choose this particular weaker form to illustrate the connection to SPDKE. We assume that these problems are globally difficult across the set of pairs $G \times End(G)$ and their associated group actions - that is, there should not be a choice of pair (g,ϕ) such that any of the above problems is significantly easier for this pair than an arbitrary choice of pair. Let us codify this assumption in a definition.

Definition 5. *Suppose* $\{G_p\}_p$ *is a family of (semi)groups as described in Section 1.5. The family of (semi)groups is easy if the following also holds. For a fixed* p *let* $(g, \phi), (g', \phi')$ *be two pairs in* $G_p \times End(G_p)$*, and let* $f(p), f'(p)$ *(resp.* $g(p), g'(p)$*) be the number of operations required to solve* SDLP *(resp.* SCDH*) for* (g, ϕ) *and* (g', ϕ') *respectively. We require that* $f(p) = \mathcal{O}(f'(p))$ *(resp.* $g(p) = \mathcal{O}(g'(p))$*).*

We now give the computational problems that we seek to reduce to. These versions of the problems are taken from [26], but can be found as the *vectorisation* and *parallelisation* problems, respectively, in [14].

Definition 6 (Group Action Discrete Logarithm). *Given a public commutative group action* $(G, X, \star)$*, sample* $g \in G$ *and* $x \in X$ *uniformly at random, compute* $y = g \star x$ *and create the pair* (x, y)*. The Group Action Discrete Logarithm Problem (*GADLP*) with respect to* x *is to recover* g *given the pair* (x, y)*.*

Definition 7 (Group Action Computational Diffie–Hellman). *Given a public commutative group action* $(G, X, \star)$*, sample* $g, h \in G$ *and* $x \in X$ *uniformly at random. Compute* $y = g * x$ *and* $z = h * x$*, and create the tuple* (x, z, y)*. The Group Action Computational Diffie–Hellman problem (*GACDH*) is to recover* $(g + h) \star y$ *given the tuple* (x, y, z)*.*

These versions of GADLP and GACDH are slightly weaker than those found in [26], which allow for x to be chosen according to *any* distribution. For the purposes of our reduction, the uniform distribution will do. Note also that in our reductions, SDLP and SCDH are defined relative to some fixed pair (g, ϕ), which will induce some instances of GADLP, GACDH in which the value $x \in X$ is *not* sampled uniformly at random and is instead fixed. We get around this by observing that for each $x \in X$, fixing x defines a GADLP (resp. GACDH) problem with respect to this x. Clearly, a GADLP (resp. GACDH) oracle for the group action $(G, X, \star)$ can solve the GADLP (resp. GACDH) problem with respect to x for each $x \in X$. The distinction is sufficiently subtle for us to suppress this detail for the remainder of the paper.
Finally we give a seemingly unrelated problem requiring a small amount of introduction. Let $f, g : A \to S$ be injective functions, where S is a set and A is a finite abelian group. We say that f, g *hide* some $s \in A$ if one has $g(a) = f(a + s)$ for each $a \in A$.

Definition 8 (Abelian Hidden Shift Problem). *Given a public abelian group* A *and a set* S*, suppose two injective functions* f, g *hide some* $s \in A$*. The Abelian Hidden Shift Problem (*AHSP*) is to recover the group element* s*.*

2 Structure of the Exponents

All of the algorithms in this paper rely on the construction of a certain group action - recall that such an object consists of a group, a set, and a function (Section 1.2, Definition 1). As a general outline to our strategy, we first define

and deduce properties of a particular set, from which the appropriate group and function will follow.
With this in mind, we make the following definition. For now we will dispense with our notion of a parameterised family of (semi)groups, since the results presented in this section apply to any fixed (semi)group. In fact, for compactness of exposition, for the remainder of this section by G we mean an arbitrary finite (semi)group, and by $End(G)$ we mean its associated endomorphism semigroup.

Definition 9. *For a pair $(g, \phi) \in G \times End(G)$, define*

$$\mathcal{X}_{(g,\phi)} := \{s(g, \phi, i) : i \in \mathbb{N}\}$$

We will often write $\mathcal{X}_{(g,\phi)}$ as $\mathcal{X}$ when clear from context. Certainly this object is neither a group nor a semigroup - numerous counterexamples can be found whereby multiplication of elements in this set are not contained in the set - but we can make some progress by borrowing from the standard theory of monogenic semigroups; presented, for example, in [18]. Since $\mathcal{X} \subset G$, $\mathcal{X}$ is finite - the set $\{x \in \mathbb{N} : \exists y \quad s(g, \phi, x) = s(g, \phi, y)\}$ must therefore be non-empty, else it is in bijection with the natural numbers. We may therefore choose its smallest element, say n. By definition of n the set $\{x \in \mathbb{N} : s(g, \phi, n) = s(g, \phi, n + x)\}$ must also be non-empty, so we may again pick its smallest element and call it r.
The structure of $\mathcal{X}$ is further restricted by the following result:

Lemma 1. *Let $(g, \phi) \in G \times End(G)$ and $x, y \in \mathbb{N}$, then*

$$\phi^x\left(s(g, \phi, y)\right) \cdot s(g, \phi, x) = s(g, \phi, x + y)$$

Proof. Note that $s(g, \phi, x+y) = \phi^{x+y-1}(g) \cdot \ldots \cdot g$. Since ϕ preserves multiplication, applying ϕ^x to $s(g, \phi, y)$ adds x to the exponent of each term. Multiplication on the right by $s(g, \phi, x)$ then completes the remaining terms of $s(g, \phi, x + y)$. □

Remark 1. One can entirely symmetrically swap the roles of x and y in the above argument, which gives two ways of calculating $s(g, \phi, x+y)$. In essence, therefore, this result gives us a slightly more elegant proof of the correctness of `SPDKE`.

This method of inducing addition in the integer argument of s is sufficiently important that we will invoke a definition for it.

Definition 10. *Let $(g, \phi) \in G \times End(G)$ and define a function $f : \mathbb{N} \times \mathcal{X} \to \mathcal{X}$ by*

$$f(i, s(g, \phi, j)) = \phi^i(s(g, \phi, j)) \cdot s(g, \phi, i)$$

where $f(i, s(g, \phi, j))$ may also be written as $i * s(g, \phi, j)$.

Remark 2. If G is of the type discussed in Section 1.5, the value $i * s(g, \phi, j)$ can be computed in time $\mathcal{O}(\log i)$. This is because computing $\phi^i(s(g, \phi, j))$ requires calculating some fixed number of G-elements to the power i and multiplying, which can be done with $\mathcal{O}(\log i)$ operations by square and multiply; and, as we have seen, computing $s(g, \phi, i)$ requires $\mathcal{O}(\log i)$ operations.

Thus far we have established that corresponding to any fixed pair $(g,\phi) \in G \times End(G)$ is a set $\mathcal{X}_{g,\phi} = \mathcal{X}$ and a pair of integers n, r. By Lemma 1 we know that $i * s(g,\phi,j) = s(g,\phi,i+j)$ for any $i, j \in \mathbb{N}$, so by definition of n, r we have

$$\begin{aligned} s(g,\phi,n+2r) &= r * s(g,\phi,n+r) \\ &= r * s(g,\phi,n) \\ &= s(g,\phi,n+r) = s(g,\phi,n) \end{aligned}$$

We conclude, by extending this argument in the obvious way, that $s(g,\phi,n+qr) = s(g,\phi,n)$ for each $q \in \mathbb{N}$. In fact, we have the following:

Lemma 2. *Fix $(g,\phi) \in G \times End(G)$ and let n, r be the corresponding integer pair as above. One has that*

$$s(g,\phi,n+x+qr) = s(g,\phi,n+x)$$

for all $x, q \in \mathbb{N}$.

We will frequently invoke Lemma 2. Indeed, we immediately get that the set $\mathcal{X}$ cannot contain values other than $\{g,...,s(g,\phi,n),...,s(g,\phi,n+r-1)\}$. If any of the values in $\{g,...,s(g,\phi,n-1)$ are equal we contradict the minimality of n, and if any of the values in $\{s(g,\phi,n),...,s(g,\phi,n+r-1)\}$ are equal we contradict the minimality of r. We have shown the following:

Theorem 1. *Fix $(g,\phi) \in G \times End(G)$. The set $\mathcal{X} = \{s(g,\phi,i) : i \in \mathbb{N}\}$ has size $n+r-1$ for integers n, r dependent on g, ϕ. In particular*

$$\mathcal{X} = \{g,...,s(g,\phi,n),...,s(g,\phi,n+r-1)\}.$$

We refer to the set $\{g,...,s(g,\phi,n-1)\}$ as the *tail*, written $\mathcal{T}_{g,\phi}$, of $\mathcal{X}_{g,\phi}$; and the set $\{s(g,\phi,n),...,s(g,\phi,n+r-1)\}$ as the *cycle*, written $\mathcal{C}_{g,\phi}$, of $\mathcal{X}_{g,\phi}$. The values $n_{g,\phi}$ and $r_{g,\phi}$ are called the *index* and *period* of the pair (g,ϕ). Note that the subscript notation is sometimes ommitted when clear from context.

One can see that unique natural numbers correspond to each element in the tail, but infinitely many correspond to each element in the cycle. In fact, each element of the cycle corresponds to a unique residue class modulo r, shifted by the index n. This is a rather intuitive fact, but owing to its usefulness we will record it formally. In the following we assume the function $\bmod$ returns the canonical positive residue.

Theorem 2. *Fix $(g,\phi) \in G \times End(G)$ and let $x, y \in \mathbb{N}$. We have*

$$s(g,\phi,n+x) = s(g,\phi,n+y)$$

if and only if $x \bmod r = y \bmod r$.

Proof. In the reverse direction, setting $x' = x \bmod r$ and $y' = y \bmod r$, we have by Lemma 2 that $s(g,\phi,n+x) = s(g,\phi,n+x')$ and $s(g,\phi,n+y) = s(g,\phi,n+y')$.

By assumption $x' = y'$, and $0 \leq x', y' < r$. The claim follows since we know values in the range $\{s(g, \phi, n), ..., s(g, \phi, n + r - 1)\}$ are distinct by Theorem 1.
On the other hand, suppose $s(g, \phi, n + y) = s(g, \phi, n + x)$ but $x \not\equiv y \mod r$. Without loss of generality we can write $y = x' + u + qr$ for some $q \in \mathbb{N}, 0 < u < r$ and $x' = x \mod r$. By Lemma 2, since $s(g, \phi, n + y) = s(g, \phi, n + x)$ we must have

$$s(g, \phi, n + x') = s(g, \phi, n + x' + u)$$

where $s(g, \phi, n + x) = s(g, \phi, n + x')$ also by Lemma 2. There are now three cases to consider; we claim each of them gives a contradiction.
First, suppose $x' + u = r$, then $s(g, \phi, n + x') = s(g, \phi, n)$. Since $x' < r$ we contradict minimality of r. The case $x' + u < r$ gives a similar contradiction.
Finally, if $x' + u > r$, without loss of generality we can write $x' + u = r + v$ for some positive integer v, so we have $s(M, \phi, n + x') = s(M, \phi, n + v)$. Since $x' \neq v$ (else we contradict $u < r$), and both values are strictly less than r, we have a contradiction, since distinct integers of this form give distinct evaluations of s.
□

We are almost ready to define our group action; first, however, we must specify the group acting on the cycle.

2.1 Positive Residues

The previous result implies that we are in some sense interested in the action of residue classes, but in order to use the usual notion of integers $\mod r$, denoted here by $\mathbb{Z}_r$, we would need to be comfortable letting the function s take negative integer inputs. In fact, a well-behaved notion of the output of s on negative integers can be constructed provided one is willing to let g and ϕ be invertible. Fortunately, we need not restrict ourselves to this case, and instead consider the following object:

Definition 11. *We write $\mathbb{N}_r = \{[i]_r : 0 \leq i < r\}$, where $[i]_r = \{k \in \mathbb{N} : k \equiv i \mod r\}$. We define the operation, written additively, by $[i]_r + [j]_r = [i + j]_r$.*

The reader happy to take this group at face value may skip to the next section. Nevertheless, a potential *cri de cœur* is that, unlike its cousin $\mathbb{Z}_r$, this object is not compatible with notion of set addition. It turns out that this is not cause for concern.

Theorem 3. *The object defined in Definition 11 is a finite abelian group.*

Proof. Consider the semigroup of integers $\{0, 1, 2, ...\}$ with respect to addition. Define, for some positive integer r, a relation by $i \sim j$ if and only if $i \equiv j \mod r$. It is standard that $\sim$ is an equivalence relation; indeed, it is a *congruence* relation, in that $a \sim c$ and $b \sim d$ implies that $a + b \sim c + d$. The equivalence classes under this relation are exactly the elements of $\mathbb{N}_r$; again, we can write them $[i]_r$. By, for example, [18, Theorem 1.5.2], the commutative operation $[i]_r + [j]_r = [i + j]_r$ on these equivalence classes is well-defined; and the set of equivalence classes with respect to this operation is a semigroup. It remains to check that the semigroup is a group, which is immediate since $[0]_r$ is clearly the identity element, and $[r]_r = [0]_r$ implies that each $[i]_r$ has inverse $[r-i]_r$. □

2.2 A Group Action

We conclude the section by constructing the action of $\mathbb{N}_r$ on the cycle

$$\{s(g,\phi,n),...,s(g,\phi,n+r-1)\}.$$

Theorem 4. *Fix $(g,\phi)\in G\times End(G)$ and let n,r be the index and period corresponding to g,ϕ. Moreover, let $\mathcal{C}$ be the corresponding cycle of size r. The abelian group $\mathbb{N}_r$ acts freely and transitively on $\mathcal{C}$.*

Proof. Define the action $\psi:\mathbb{N}_r\times\mathcal{C}\to\mathcal{C}$ by

$$\psi([j]_r,s(g,\phi,n+i)) = \phi^{[j]_r}(s(g,\phi,n+i))\cdot s(g,\phi,[j]_r)$$

This object is, of course, a set: indeed, it is equal to

$$\{\phi^{j+kr}(s(g,\phi,n+i))\cdot s(g,\phi,j+kr): k\in\mathbb{N}_0\}$$

which, by Proposition 1, is exactly the set $\{s(g,\phi,n+i+j+kr): k\in\mathbb{N}\}$. By Theorem 2, every element of this set is equal to $s(g,\phi,n+i+j)$, so we will harmlessly abuse notation by writing

$$\psi([j]_r,s(g,\phi,n+i)) = s(g,\phi,n+i+j)$$

Moreover, the output[7] of ψ is indeed in $\mathcal{C}$, since $s(g,\phi,n+i+j) = s(g,\phi,n+(i+j) \mod r)$, and $0\le(i+j) \mod r<r$. In general, when $y>r$ we are free to dispense with the slightly more cumbersome modular notation and write $s(g,\phi,n+y)$ instead of $s(g,\phi,n+(y \mod r))$.

Let us check an action is indeed defined. Certainly $[0]_r$ fixes every element of $\mathcal{C}$. Let $[j]_r,[k]_r\in\mathbb{N}_r$; then, writing $\psi([j]_r,s(g,\phi,n+i))$ as $[j]_r\circledast s(g,\phi,n+i)$ in the conventional fashion, we have

$$\begin{aligned}
[k]_r\circledast([j]_r\circledast s(g,\phi,n+i)) &= [k]_r\circledast s(g,\phi,n+i+j)\\
&= s(g,\phi,n+i+j+k)\\
&= [j+k]_r\circledast s(g,\phi,n+i)\\
&= ([k]_r+[j]_r)\circledast s(g,\phi,n+i)
\end{aligned}$$

where any protests that the sum exceeds r are countered by the well-definedness of modular addition.

[7] Or, more accurately, the content of the singleton set that ψ outputs is an element of $\mathcal{C}$.

Now let us see that the action is free and transitive. If $s(g,\phi,n+i)$ is fixed by $[j]_r$ then Theorem 2 gives that $i+j \equiv i \mod r$, so $[j]_r = [0]_r$. Thus the action is free. Fix $s(g,\phi,n+i)$ and $s(g,\phi,n+j)$; then $[r-i+j]_r \in \mathbb{N}_r$ is such that

$$[r-i+j]_r \circledast s(g,\phi,n+i) = s(g,\phi,n+i+r-i+j) = s(g,\phi,n+j)$$

as required for transitivity. □

We summarise the above by noting that for each $(g,\phi) \in G \times End(G)$ we have shown the existence of a free, transitive, commutative group action $(\mathbb{N}_r, \mathcal{C}, \circledast)$, where r and $\mathcal{C}$ depend on the choice of pair (g,ϕ).

3 Group Action Discrete Logarithms

Now that we have established the group action, we recall the Group Action Discrete Logarithm Problem (`GADLP`) from the introduction. Roughly speaking, for a free transitive group action $(G, X, \star)$, and x, y sampled uniformly at random from the set X, we are tasked with recovering the unique G-element g such that $g \star x = y$. In this section we will show that one can construct a quantum reduction from `SDLP` to `GADLP`.

More precisely, we target the type of structure discussed in Section 1.5; that, is a set of finite semigroups $\{G_p\}_p$ indexed by some parameter p, such that the size of each G_p is polynomial in p. We know that multiplication in each G_p requires a number of operations bounded above by some constant independent of p. Indeed, for each p, any fixed pair $(g,\phi) \in G_p \times End(G_p)$ is such that one can calculate both $s(g,\phi,i)$ and $i * s(g,\phi,j)$ in time $\mathcal{O}(\log i)$ for any $i \in \mathbb{N}$. We will make frequent reference to these facts.

With all this in mind let $\{G_p\}_p$ be such a family of semigroups - recall that we refer to such a family as *easy*. In the previous section we have shown that for a fixed p, to each pair $(g,\phi) \in G_p \times End(G_p)$ is associated a pair $(n_{g,\phi}, r_{g,\phi})$ and a set $\mathcal{C}_{g,\phi}$. In this section we seek to show there is an efficient quantum algorithm to solve `SDLP` with respect to (g,ϕ) provided one has access to a `GADLP` oracle for the group action $(\mathbb{N}_{r_{g,\phi}}, \mathcal{C}_{g,\phi}, \circledast)$.

Such a reduction is more or less immediate when $n = 1$, because in this case the set of exponents is exactly the cycle. This is not, however, generally the case: in order to proceed we will need a method of extracting the index and period n, r. Doing so is the ‘quantum part’ of the reduction - we note that assuming access to a quantum computer is, for our purposes, justified since the best-known algorithms for `GADLP` are quantum anyway.

3.1 Non-empty Tails

The following algorithm borrows heavily from ideas in [12], which itself is a special case of [13]: some complexity estimates, meanwhile, are from [17]. Before presenting the algorithm it will be useful for the purposes of complexity estimation to give some worst-case indicators, defined as follows:

Definition 12. *Let $\{G_p\}_{p \in P}$ be a family of finite semigroups parameterised by some set P. Define the following functions on P:*

$$n(p) = \max_{(g,\phi) \in G_p \times End(G_p)} |\mathcal{T}_{g,\phi}|$$

$$r(p) = \max_{(g,\phi) \in G_p \times End(G_p)} |\mathcal{C}_{g,\phi}|$$

$$N(p) = \max_{(g,\phi) \in G_p \times End(G_p)} |\mathcal{T}_{g,\phi} + \mathcal{C}_{g,\phi}|$$

The function $N(p)$ gives a bound on the size of $\mathcal{X}_{g,\phi}$ for any $(g,\phi) \in G_p \times End(G_p)$. Since a crude such bound is the size of (semi)group G_p, which is assumed polynomial in p, we have that $N(p)$ is at worst polynomial in p - a situation we denote as $N(p) = \mathcal{O}(poly(p))$. Of course, this applies to the functions $n(p)$ and $r(p)$ as well.

We are now ready to give the method (Algorithm 1) of extracting the index and period.

Algorithm 1 Index and Period Recovery

Input: Pair $(g,\phi) \in G_p \times End(G_p)$
Output: Pair $(n_{g,\phi}, r_{g,\phi})$

$R_0 \leftarrow |0\rangle\,|0\rangle$ where each $|0\rangle$ is a computational basis vector in an ℓ-qubit register
Apply a Hadamard gate to R_0 to get the superposition $R_1 = \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |j\rangle\,|0\rangle$
while $0 \le j < M$ **do**
    $S_j \leftarrow s(g,\phi,j)$
end while
Store each S_j in second quantum register to get $R_2 = \sum_{j=0}^{M-1} |j\rangle\,|s(g,\phi,j)\rangle$
Observe second register leaving collapsed first register R_3
$R_4 \leftarrow$ QFT applied to R_3
$R_5 \leftarrow$ observe R_4
$r \leftarrow$ classical post-processing applied to R_5
if $r * s(g,\phi,M) \neq s(g,\phi,M)$ **then**
    Algorithm failed
else
    $B \leftarrow$ empty ordered list
    while $0 \le j \le M$ **do**
        $b_j \leftarrow \gamma(S_j)$
        $B \leftarrow B \cup b_j$
    end while
    $n \leftarrow$ position of first one in B
    $(n_{g,\phi}, r_{g,\phi}) \leftarrow (n, r)$
end if
return $(n_{g,\phi}, r_{g,\phi})$

Theorem 5. *Let $\{G_p\}_p$ be an easy family of semigroups, and fix p. For any pair $(g,\phi) \in G_p \times End(G_p)$, Algorithm 1 recovers the index $n_{g,\phi}$ and period $r_{g,\phi}$ in time $\mathcal{O}(N(p)\log N(p) + (\log N(p))^3)$ with probability $\Omega(1/(r(p)+n(p)))$.*

Proof. Fix a pair $(g,\phi) \in G_p \times End(G_p)$ and let n, r be its index and period. We start by selecting some large $M = 2^\ell \in \mathbb{N}$. This M needs to be sufficiently large to 'capture' the periodic behaviour of the function s in a sense that will become clear below. It turns out (see [13, Algorithm 5]) that it suffices to take $M = \mathcal{O}(N(p)^2)$. Clearly, M has a natural representation by a quantum register of length ℓ. Since $M > N(p) > n + r$, there is an injective function from $\mathcal{X}$ into $\{0,1\}^\ell$, so we can store $\mathcal{X}$ in a length ℓ register as well.
The register R_1 is obtained by applying the Hadamard gate to $R_0 = |0\rangle|0\rangle$ which can be done in time $\mathcal{O}(\log M)$. Then, R_2 is computed from the values $\{g, ..., s(g,\phi,M-1)\}$. Assuming that for a fixed semigroup G the number of operations for a single multiplication is bounded by some constant, each function value can be prepared in time $\mathcal{O}(\log M)$ (where M is the largest exponent to be calculated) for a total time of $\mathcal{O}(M\log M)$ for Step 6.
In Step 7, we measure the second register. With probability n/M doing so will cause us to observe an element of the tail; that is, some $s(g,\phi,i)$ such that $i < n$. In this case, by the laws of partial observation, the first register is left in a superposition of integers corresponding to this value - but by definition there is only one of these, so the first register consists of a single computational basis state and the algorithm has failed. On the other hand, with probability $(M-n)/M$ measuring the second register gives an element of $\mathcal{C}$. Since $M > n+r$, this probability is better than $n/(n+r)$. After measuring, the superposition of corresponding integers in the first register is the following:

$$\frac{1}{\sqrt{s_r}}\sum_{j=0}^{s_r-1}|x_0 + jr\rangle$$

To see this, note that the function s is periodic of period r, and by Theorem 1 each $s(g,\phi,i)$ such that $i \geq n$ can only assume one of the distinct values $s(g,\phi,n), ..., s(g,\phi,n+r-1)$. In particular, the integers in $\{1,...,M\}$ that give a specific value of the cycle under s are of the form $x_0 + jr$ for some $x_0 \in \{n, ..., n+r-1\}$. The largest such integer, by definition, is $x_0 + s_r r$, where s_r is just the largest integer such that $x_0 + s_r r < M$. Note that the superposition is normalised by this factor so that the sum of the squares of the amplitudes is 1. We now have exactly the same kind of state found in [13, Algorithm 5][8], so we may proceed exactly according to the remaining steps in this algorithm. In Step 8 we apply a Quantum Fourier Transform (QFT) over $\mathbb{Z}_M$ to the state, which can be done in time $\mathcal{O}((\log M)^2)$: it is shown in [13] that with probability at least $4/\pi^2$, measuring the resulting state leaves one with information upon which classical post-processing methods of complexity $\mathcal{O}((\log M)^3)$ can be performed. In particular, once we have the type of state R_5, in Step 10 we can recover the period r with probability at least $4/\pi^2$, or $\Omega(1)$.

[8] This type of state also occurs in Shor's factoring algorithm.

It remains to recover the index n, which can be done classically using the same idea as [12, Lemma 1]. Consider an indicator function $\gamma : \{g, ..., s(g, \phi, M)\} \to \{0, 1\}$ such that $\gamma(s(g, \phi, i)) = 1$ if and only if $r * s(g, \phi, i) = s(g, \phi, i)$ and 0 otherwise; then γ is an indicator function for membership in the cycle. Consider the list

$$\{\gamma(g), ..., \gamma(s(g, \phi, n)), ..., \gamma(s(g, \phi, M))\}$$

computed in Step 15. Note that computing γ requires the computation of $r * s(g, \phi, i)$ a total of M times. Crudely bounding the cost of computing $s(g, \phi, r)$ by $\mathcal{O}(\log M)$, since we have already computed the list $\{g, ..., s(g, \phi, M)\}$ in Step 3, we get the same list under γ in time $\mathcal{O}(M \log M)$. By definition, this is an ordered list of $n - 1$ zeroes and $M - (n - 1)$ ones. In particular, in Step 19, we can apply binary search of complexity $\mathcal{O}(\log M)$ to find the position of the first one, and therefore the value of the index n.

In summary, we have an algorithm that, given a state of the form in Step 4, recovers the index and period with constant probability in time $\mathcal{O}((\log M)^3)$. Arriving at such a state can be done in time $\mathcal{O}(M \log M)$ with probability better than $r/(n + r)$, which by definition is itself better than $1/(r(p) + n(p))$. We conclude that the algorithm succeeds with probability $\Omega(1/(r(p) + n(p)))$ in time $\mathcal{O}((M \log M) + (\log M)^3)$, which is $\mathcal{O}((N(p)^2 \log N(p)) + (\log N(p))^3)$ by assumption. □

The Probability of Observing a Useful State. We here provide further insight into how the quantity $r(p)/(n(p) + r(p))$ is bounded.

Recall in Theorem 1 that the existence of n, r essentially follows from the pigeonhole principle. In particular, there is at time of writing no *a priori* information about their size or relationship to each other. We therefore propose a hypothetical easy family of semigroups $\{G_p\}_p$ for which the algorithm in Theorem 5 is 'large', in the sense that is bounded from below by a constant that does not depend on p. Consider the functions defined in Definition 12. We say a family of semigroups $\{G_p\}_p$ has *small tails* if there is a constant $C > 0$ such that $n(p) < Cr(p)$ for all p. Since by definition one has that for any (g, ϕ) the associated period r is such that $r \geq 1$, a family of semigroups with small tails is such that for any index-period pair n, r associated to a pair (g, ϕ), one has

$$\frac{r}{n + r} \geq \frac{1}{(C + 1)r(p)} > \frac{1}{C + 1}$$

so we have a large success probability in the sense described above.

The assumption that $\{G_p\}_p$ has small tails is somewhat problematic. In order to keep the bound M optimal we should like it to be as tight as possible to $n+r$; if the vast majority of the elements in $\{g, ..., s(g, \phi, M)\}$ are in the tail even as p grows we expect our algorithms to be less effective, as they will succeed with diminishing probability. There is therefore motivation to search for semigroup families that do not have small tails - in particular, families such that the index of any pair g, ϕ is expected to be exponential in the period of that pair - which we leave for future work.

3.2 From SDLP to GADLP

Let us assemble the components developed so far in this section into a reduction of `SDLP` to `GADLP`. Similarly to Theorem 5, the complexity bounds are given in terms of a bound N on the size of $n + r$.

Theorem 6. *Let $\{G_p\}_p$ be an easy family of semigroups, and fix p. Algorithm 2 solves* `SDLP` *with respect to a pair $(g,\phi) \in G_p \times End(G_p)$ given access to a* `GADLP` *oracle for the group action $(\mathbb{N}_{r_{g,\phi}}, \mathcal{C}_{g,\phi}, \circledast)$. The algorithm runs in time $\mathcal{O}(N(p)^2 \log N(p) + (\log N(p))^3)$, makes at most a single query to the* `GADLP` *oracle, and succeeds with probability $\Omega(1/(n(p)+r(p)))$.*

Algorithm 2 Solving `SDLP` with `GADLP` oracle

Input: $(g,\phi), s(g,\phi,x)$
Output: x

Proof. 1: $(n,r) \leftarrow$ Algorithm 1 applied to (g,ϕ)
2: **if** $r * s(g,\phi,x) = s(g,\phi,x)$ **then**
3: $\quad d \leftarrow s(g,\phi,n)$
4: $\quad x' \leftarrow$ `GADLP` oracle applied to $d, s(g,\phi,x)$
5: $\quad x \leftarrow n + x'$
6: **else**
7: $\quad B \leftarrow$ ordered empty list
8: $\quad$ **while** $0 \le j \le n$ **do**
9: $\quad\quad b_j \leftarrow \gamma(j * s(g,\phi,x))$
10: $\quad\quad B \leftarrow B \cup b_j$
11: $\quad$ **end while**
12: $\quad y \leftarrow$ position of first one in B
13: $\quad x \leftarrow n - y$
14: **end if**
15: **return** x

Consider an instance of `SDLP` whereby we are given the pair (g,ϕ) and the value $s(g,\phi,x)$, for some x sampled uniformly at random from the set $\{1,..,n_{g,\phi}+r_{g,\phi}\}$. We show that Algorithm 2 recovers x.

We start in Step 1 by applying Algorithm 1 to the pair (g,ϕ), recovering the pair n,r. Now, $s(g,\phi,x)$ might be in tail or in the cycle - but with our knowledge of r we can check in Step 2 which is true by verifying whether $r * s(g,\phi,x) = s(g,\phi,x)$. We can do this in time $\mathcal{O}(\log r(p)) \le \mathcal{O}(\log N(p))$.

There are now two cases to consider. First, suppose that the check in Step 2 is passed, then $s(g,\phi,x)$ is in the cycle, and we may proceed as follows. Compute $s(g,\phi,n)$ in time $\mathcal{O}(\log n(p)) \le \mathcal{O}(\log N(p))$ (Step 3) and query the `GADLP` oracle on input $s(g,\phi,n), s(g,\phi,x)$ (Step 4) to recover the $\mathbb{N}_r$ element $[y]_r$. Without loss of generality the smallest representative of this class, say x', is such that $n + x' = x$, so we recover x in Step 5.

Now suppose that $s(g, \phi, x)$ is in the tail. We may now apply a classical method to recover x similar to [12, Theorem 1]. Via repeated application of the $*$ operation, each of which takes time $\mathcal{O}(n(p)) \leq \mathcal{O}(N(p))$, compute the ordered list

$$\{s(g, \phi, x), s(g, \phi, x+1), ..., s(g, \phi, x+n)\}$$

Borrowing the notation of Algorithm 1, we have that computing the ordered list

$$\{\gamma(s(g, \phi, x), \dots, s(g, \phi, n+x)\}$$

can be done in time $\mathcal{O}(r(p)\log n(p)) \leq \mathcal{O}(N(p)\log N(p))$. The computation of these objects is carried out in Steps 9 and 10.
Certainly there is at least one 1 in this list since $n + x > n$. In Step 12, therefore, we can perform binary search on the list to find the position of the first 1 in time $\mathcal{O}(\log N(p))$. This value, say y, is such that $x + y = n$, so we recover x in Step 13. In either case, the complexity of recovering x is dominated by the complexity of Algorithm 1, so we conclude that Algorithm 2 runs with the same complexity. ◻

In summary, we have an efficient quantum reduction from SDLP to GADLP: an efficient quantum procedure extracts the period r, and from there a classical procedure gives the index n. In order to recover x, it remains to either carry out an efficient classical procedure, or recover x with a single query to a GADLP oracle. Moreover, since the success probability is precisely that of Algorithm 1 multiplied by that of the GADLP oracle, suppose $\{G_p\}_p$ has short tails. It follows that if the GADLP oracle succeeds with probability bounded below by some constant independent of p, Algorithm 2 succeeds with probability $\Omega(1)$.

3.3 Equivalence of SCDH and SDLP

Let us briefly recall the classical setting for the Discrete Logarithm Problem. It is well-known (see, for example, [22]) that the relationship between the Discrete Logarithm Problem and the Computational Diffie–Hellman Problem is not well understood. In particular, one can certainly solve the Computational Diffie–Hellman Problem provided one can solve the Discrete Logarithm Problem, but the converse is not known.
The situation in the group action setting is rather clearer. In recent work [26], Montgomery and Zhandry demonstrate full quantum equivalence of GADLP and GACDH. This remarkable result can be extended by our work in this section to show quantum equivalence of SCDH and SDLP. Setting the scene, let G_p be a finite semigroup taken from an easy family of semigroups $\{G_p\}_p$, and $End(G_p)$ its endomorphism semigroup. Fixing a pair $(g, \phi) \in G_p \times End(G_p)$, we have seen that we obtain a group action $(\mathbb{N}_{r_{g,\phi}}, \mathcal{C}_{g,\phi}, \circledast)$. Consider the computational problems SDLP, SCDH with respect to (g, ϕ) and the problems GADLP, GACDH with respect to $(\mathbb{N}_{r_{g,\phi}}, \mathcal{C}_{g,\phi}, \circledast)$. We have the following chain of reductions: in Theorem 6 we show that SDLP reduces to GADLP, and the work in [26] demonstrates that GADLP reduces to GACDH. Since SCDH clearly reduces to SDLP, in order to show quantum

equivalence of SCDH and SDLP it suffices to demonstrate that one can solve GACDH for $(\mathbb{N}_r, \mathcal{C}, \circledast)$ with access to an SCDH oracle with respect to (g, ϕ). In fact, one can show this rather easily:

Lemma 3. *Let $\{G\}_p$ be an easy family of semigroups, and fix p. For a pair $(g, \phi) \in G_p \times End(G_p)$, there is an algorithm to solve* GACDH *with respect to $(\mathbb{N}_{r_{g,\phi}}, \mathcal{C}_{g,\phi}, \circledast)$ given access to an* SCDH *oracle with respect to the pair (g, ϕ).*

Proof. In the following we will dispense with subscripts indicating that the relevant values correspond to the pair (g, ϕ). Consider an instance of GADLP where we are given the tuple $(x, [i]_r \circledast x, [j]_r \circledast x)$, where $x \in \mathcal{C}$ and $[i]_r, [j]_r \in \mathbb{N}_r$ are sampled uniformly at random. Recall that our task is to recover the value $[i + j]_r \circledast x$.
Note that since $x \in \mathcal{C}$ there is some $k \in \{0, ..., r-1\}$ such that $s(g, \phi, n+k)$. In particular, $x \in G$, so we may rename x as g'. Moreover, we have $x = s(g', \phi, i), y = s(g', \phi, j)$ where i, j are chosen arbitrarily as the smallest representatives of $[i]_r, [j]_r$ respectively (note that any representative will do). We may therefore query an SCDH oracle for (g', ϕ) on the tuple (x, y, z) to obtain $s(g', \phi, i+j) = [i + j]_r \circledast x$, so we are done. □

We summarise the state of affairs with the following diagram, where a directed arrow between two nodes indicates the problem in the former node can be solved by an oracle for the problem in the latter. Moreover, a solid arrow indicates a reduction demonstrated in this paper, whereas a dashed arrow denotes a reduction that was already known.

$$\begin{tikzcd}
\boxed{\texttt{SDLP}} \arrow[d, "\text{Theorem 6}"'] & & \boxed{\texttt{SCDH}} \arrow[ll, dashed, "\text{clear}"'] \\
\boxed{\texttt{GADLP}} \arrow[rr, leftrightarrow, dashed, "\text{[26, Theorem 9]}"'] & & \boxed{\texttt{GACDH}} \arrow[u, "\text{Lemma 3}"']
\end{tikzcd}$$

Fig. 1. Landscape of known reductions.

4 Quantum Algorithms for GADLP

Now that we have shown SDLP can be efficiently solved with access to an appropriate GADLP oracle it remains to examine the state of the art for GADLP. It is here that the Abelian Hidden Shift Problem (Definition 8) comes in to play. Roughly speaking, we are given two injective functions f, g from a group A to a set S that differ by a constant 'shift' value, and our task is to recover the shift value.

It is reasonably well-known (see [31, 11]) that GADLP reduces to AHSP. In this section, we provide a context-specific proof of this fact, before discussing the best known algorithms for AHSP.

4.1 Group Actions to Hidden Shift

The following result is found more or less verbatim in, for example, [11]. We here give a context-specific reduction, for completeness.

Theorem 7. *Let $\{G_p\}_p$ be an easy family of semigroups and fix p. For some pair $(g, \phi) \in G_p \times End(G_p)$ let $(\mathbb{N}_r, \mathcal{C}, \circledast)$ be the associated group action defined in Theorem 4. One can efficiently solve* GADLP *in $(\mathbb{N}_r, \mathcal{C}, \circledast)$ given access to an* AHSP *oracle with respect to $\mathbb{N}_r, \mathcal{C}$.*

Proof. Suppose we are given an instance of GADLP in $(\mathbb{N}_r, \mathcal{C}, \circledast)$; that is, we are given a pair $(s(g, \phi, n+i), s(g, \phi, n+j)) \in \mathcal{C}$ for some $i, j \in \{1, ..., r\}$ and tasked with finding the unique $[k]_r \in \mathbb{N}_r$ such that $[k]_r \circledast s(g, \phi, i) = s(g, \phi, j)$. Our strategy is to construct injective functions $f_A, f_B : \mathbb{N}_r \to \mathcal{C}$ that hide $[k]_r$, and use the AHSP oracle to recover this value.
Set $f_A, f_B : \mathbb{N}_r \to \mathcal{C}$ as $f_A([x]_r) = [x]_r * s(g, \phi, n+i)$ and $f_B([x]_r) = [x]_r * s(g, \phi, n+j)$. Then

$$
\begin{aligned}
f_B([x]_r) &= [x]_r * s(g, \phi, n+j) \\
&= [x]_r * ([k]_r * s(g, \phi, n+i)) \\
&= ([x]_r + [k]_r) * s(g, \phi, n+i) \\
&= f_A([x]_r + [k]_r)
\end{aligned}
$$

In other words, f_A, f_B hide $[k]_r$. To complete the setup of an instance of AHSP we require the functions to be injective, which follows from the action being free and transitive. ◻

Note that we have in this case left out complexity estimates. This is because in order to give a full description of the functions f_A, f_B we need to compute the group $\mathbb{N}_r$, which can be done efficiently with knowledge of r. However, since we have already described a method of recovering r, we will discuss the complexity in the full SDLP algorithm at the end of this section.

4.2 Hidden Shift Algorithms

We have finally arrived at the problem for which there are known quantum algorithms. The fastest known is of subexponential complexity, and is presented in [23, Proposition 6.1] as a special case of the Dihedral Hidden Subgroup Problem.

Theorem 8 (Kuperberg's Algorithm). *There is a quantum algorithm that solves* AHSP *with respect to $\mathbb{N}_r, \mathcal{C}$ with time and query complexity $2^{\mathcal{O}(\sqrt{\log r})}$.*

Kuperberg's algorithm also requires quantum space $2^{\mathcal{O}(\log r)}$. For a slower but less space-expensive algorithm, we can also use a generalised version of an algorithm due to Regev [30]. The generalised version appears in [11, Theorem 5.2].

Theorem 9 (Regev's Algorithm). *There is a quantum algorithm that solves* `AHSP` *with respect to* $\mathbb{N}_r, \mathcal{C}$ *with time and query complexity*

$$e^{\sqrt{2}+o(1)\sqrt{\ln r \ln\ln r}}$$

and space complexity $\mathcal{O}(poly(\log r))$.

We note that both Kuperberg's and Regev's algorithms succeed with constant probability.

4.3 Solving `SDLP`

We finish the section by stitching all the components together into an algorithm that solves `SDLP`. For brevity of exposition we include only complexity estimates for using Kuperberg's algorithm - but finding the bounds in the case of Regev's algorithm is very similar. Recalling the discussion in Section 3.1, the algorithm will succeed with constant probability provided the family of semigroups considered has small tails.

Theorem 10. *Let* $\{G_p\}_p$ *be an easy family of semigroups, and fix* p*. For any pair* $(g,\phi) \in G_p \times End(G_p)$*, there is a quantum algorithm solving* `SDLP` *with respect to* (g,ϕ) *with time and query complexity* $2^{\mathcal{O}(\sqrt{\log p})}$.

Let $(g,\phi) \in G_p \times End(G_p)$ and suppose we are given the value $s(g,\phi,x)$ for some x sampled uniformly from the set $\{1,...,N\}$, where N is the size of $\mathcal{X}_{g,\phi}$. The following algorithm recovers x:

1. Run the algorithm in Theorem 5 on the pair (g,ϕ). With positive probability we recover the index and period of (g,ϕ), the pair (n,r); if the family of semigroups has small tails we recover such a pair with better than constant probability.
2. By Theorem 6, either we are done efficiently, or it remains to solve an instance of `GADLP` with respect to the group action $(\mathbb{N}_r, \mathcal{C}, \circledast)$.
3. By Theorem 7, once we have computed the group action $(\mathbb{N}_r, \mathcal{C}, \circledast)$ it remains to solve an instance of `AHSP` with respect to $\mathbb{N}_r, \mathcal{C}$. This can be done with access to the index and period n, r.
4. Solve `AHSP` using Kuperberg's algorithm or Regev's algorithm.

We have discussed the complexity for each step in the relevant theorems, except for the complexity of computing the group action in step 3. We note that the complexity of carrying this out, as well as the complexity of the prior steps, is completely dominated by that of applying one of the `AHSP` algorithms. In particular this complexity if Kuperberg's algorithm is used is

$$2^{\mathcal{O}(\sqrt{\log r})} < 2^{\mathcal{O}(\sqrt{\log r(p)})}$$

Since $r(p) = \mathcal{O}(poly(p))$ we conclude that the argument runs with time and query complexity $2^{\mathcal{O}(\sqrt{\log p})}$. Since Step 1 succeeds with constant probability if $\{G_p\}_p$ has small tails, the algorithm succeeds with constant probability in this case.

5 Conclusion

We have provided the first dedicated analysis of `SDLP`, showing a reduction to a well-studied problem. Perhaps the most surprising aspect of the work is the progress made by a simple rephrasing; we made quite significant progress through rather elementary methods, and we suspect much more can be made within this framework.

The reader may notice that we have shown that `SPDKE` is an example of a commutative action-based key exchange, and that breaking all such protocols can be reduced to the Abelian Hidden Shift Problem. Indeed, this work shows the algebraic machinery of `SPDKE` is a candidate for what Couveignes calls a *hard homogenous space*[9] [14], which was not known until now. In line with the naming conventions in this area we propose a renaming of `SPDKE` to `SPDH`, which stands for 'Semidirect Product Diffie–Hellman', and should be pronounced *spud*.

We would also like to stress the following sentiment. The purpose of this paper is not to claim a general purpose break of `SPDKE` (or, indeed, `SPDH`) - the algorithm presented is subexponential in complexity, which has been treated as tolerable in classical contexts. Instead, the point is to show a connection between `SDLP` and a known hardness problem, thereby providing insight on a problem about which little was known.

Acknowledgements

We thank Chloe Martindale of the University of Bristol and Samuel Jaques of the University of Oxford for fruitful discussion on this topic at the recent ICMS conference *Foundations and Applications of Lattice-based Cryptography.* We furthermore thank the organisers of this conference for facilitating this conversation.

References

[1] John Baena, Pierre Briaud, Daniel Cabarcas, Ray A. Perlner, Daniel Smith-Tone, and Javier A. Verbel. "Improving Support-Minors rank attacks: applications to GeMSS and Rainbow". In: *IACR Cryptol. ePrint Arch.* (2021), p. 1677. URL: `https://eprint.iacr.org/2021/1677`.

[2] Christopher Battarbee, Delaram Kahrobaei, and Siamak F Shahandashti. "Semidirect Product Key Exchange: the State of Play". In: *arXiv preprint arXiv:2202.05178* (2022).

[9] Another major example of which arises from the theory of isogenies between elliptic curves - see, for example, [9]

[3] Christopher Battarbee, Delaram Kahrobaei, and Siamak F. Shahandashti. "Cryptanalysis of Semidirect Product Key Exchange Using Matrices Over Non-Commutative Rings". In: *Mathematical Cryptology* 1.2 (2022), pp. 2–9.

[4] Christopher Battarbee, Delaram Kahrobaei, Dylan Tailor, and Siamak F Shahandashti. "On the efficiency of a general attack against the MOBS cryptosystem". In: *arXiv:2111.05806; to appear in Journal of Mathematical Cryptology* (2022).

[5] Ward Beullens. "Breaking Rainbow Takes a Weekend on a Laptop". In: *IACR Cryptol. ePrint Arch.* (2022), p. 214. URL: https://eprint.iacr.org/2022/214.

[6] Daniel Brown, Neal Koblitz, and Jason Legrow. "Cryptanalysis of 'MAKE'". In: *Journal of Mathematical Cryptology* 16.1 (2022), pp. 98–102.

[7] Antoine Casanova, Jean-Charles Faugere, Gilles Macario-Rat, Jacques Patarin, Ludovic Perret, and Jocelyn Ryckeghem. "GeMSS : a great multivariate short signature". PhD thesis. UPMC-Paris 6 Sorbonne Universités; INRIA Paris Research Centre, PolSys Team, 2017.

[8] Wouter Castryck and Thomas Decru. *An efficient key recovery attack on SIDH (preliminary version)*. Cryptology ePrint Archive, Paper 2022/975. 2022. URL: https://eprint.iacr.org/2022/975.

[9] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. "CSIDH: an efficient post-quantum commutative group action". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2018, pp. 395–427.

[10] Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. *Report on Post-Quantum Cryptography*. Research report NISTIR 8105. NIST, 2016. URL: http://csrc.nist.gov/publications/drafts/nistir-8105/nistir_8105_draft.pdf.

[11] Andrew Childs, David Jao, and Vladimir Soukharev. "Constructing elliptic curve isogenies in quantum subexponential time". In: *Journal of Mathematical Cryptology* 8.1 (2014), pp. 1–29.

[12] Andrew M Childs and Gábor Ivanyos. "Quantum computation of discrete logarithms in semigroups". In: *Journal of Mathematical Cryptology* 8.4 (2014), pp. 405–416.

[13] Andrew M Childs and Wim Van Dam. "Quantum algorithms for algebraic problems". In: *Reviews of Modern Physics* 82.1 (2010), p. 1.

[14] Jean-Marc Couveignes. "Hard homogeneous spaces". In: *Cryptology ePrint Archive* (2006). URL: https://eprint.iacr.org/2006/291.pdf.

[15] Dima Grigoriev and Vladimir Shpilrain. "Tropical cryptography II: extensions by homomorphisms". In: *Communications in Algebra* 47.10 (2019), pp. 4224–4229.

[16] Maggie Habeeb, Delaram Kahrobaei, Charalambos Koupparis, and Vladimir Shpilrain. "Public key exchange using semidirect product of (semi) groups". In: *International Conference on Applied Cryptography and Network Security*. Springer. 2013, pp. 475–486.

[17] Mika Hirvensalo. *Quantum computing*. Springer Science & Business Media, 2003.

[18] John Mackintosh Howie. *Fundamentals of semigroup theory*. 12. Oxford University Press, 1995.

[19] Steve Isaac and Delaram Kahrobaei. "A closer look at the tropical cryptography". In: *International Journal of Computer Mathematics: Computer Systems Theory* (2021), pp. 1–6.

[20] Delaram Kahrobaei, Ramon Flores, and Marialaura Noce. "Group-based Cryptography in the Quantum Era". In: *The Notices of American Mathematical Society, to appear* (2022). URL: `https://arxiv.org/abs/2202.05917`.

[21] Delaram Kahrobaei and Vladimir Shpilrain. "Using semidirect product of (semi) groups in public key cryptography". In: *Conference on Computability in Europe*. Springer. 2016, pp. 132–141.

[22] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2020.

[23] Greg Kuperberg. "A subexponential-time quantum algorithm for the dihedral hidden subgroup problem". In: *SIAM Journal on Computing* 35.1 (2005), pp. 170–188.

[24] Luciano Maino and Chloe Martindale. *An attack on SIDH with arbitrary starting curve*. Cryptology ePrint Archive, Paper 2022/1026. 2022. URL: `https://eprint.iacr.org/2022/1026`.

[25] Chris Monico. "Remarks on MOBS and cryptosystems using semidirect products". In: *arXiv preprint arXiv:2109.11426* (2021).

[26] Hart Montgomery and Mark Zhandry. "Full Quantum Equivalence of Group Action DLog and CDH, and More". In: *Cryptology ePrint Archive* (2022).

[27] Alexei Myasnikov and Vitaliĭ Roman'kov. "A linear decomposition attack". In: *Groups Complexity Cryptology* 7.1 (2015), pp. 81–94.

[28] Nael Rahman and Vladimir Shpilrain. "MAKE: A matrix action key exchange". In: *Journal of Mathematical Cryptology* 16.1 (2022), pp. 64–72.

[29] Nael Rahman and Vladimir Shpilrain. "MOBS: Matrices Over Bit Strings public key exchange". In: *https://eprint.iacr.org/2021/560* (2021).

[30] Oded Regev. "A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space". In: *arXiv preprint quant-ph/0406151* (2004).

[31] Anton Stolbunov. "Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves". In: *Advances in Mathematics of Communications* 4.2 (2010), p. 215.

[32] Chengdong Tao, Albrecht Petzoldt, and Jintai Ding. "Efficient Key Recovery for All HFE Signature Variants". In: *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. Lecture Notes in Computer Science. Springer, 2021, pp. 70–93. URL: `https://doi.org/10.1007/978-3-030-84242-0%5C_4`.