Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data*

Yevgeniy Dodis† Rafail Ostrovsky‡ Leonid Reyzin§ Adam Smith¶

April 28, 2006

Abstract

We provide formal definitions and efficient secure techniques for

- turning noisy information into keys usable for *any* cryptographic application, and, in particular,
- reliably and securely authenticating biometric data.

Our techniques apply not just to biometric information, but to any keying material that, unlike traditional cryptographic keys, is (1) not reproducible precisely and (2) not distributed uniformly. We propose two primitives: a *fuzzy extractor* reliably extracts nearly uniform randomness R from its input; the extraction is error-tolerant in the sense that R will be the same even if the input changes, as long as it remains reasonably close to the original. Thus, R can be used as a key in a cryptographic application. A *secure sketch* produces public information about its input w that does not reveal w, and yet allows exact recovery of w given another value that is close to w. Thus, it can be used to reliably reproduce error-prone biometric inputs without incurring the security risk inherent in storing them.

We define the primitives to be both formally secure and versatile, generalizing much prior work. In addition, we provide nearly optimal constructions of both primitives for various measures of "closeness" of input data, such as Hamming distance, edit distance, and set difference.

*A preliminary version of this work appeared in Eurocrypt 2004 [DRS04].

†`dodis@cs.nyu.edu`. New York University, Department of Computer Science, 251 Mercer St., New York, NY 10012 USA.

‡`rafail@cs.ucla.edu`. University of California, Los Angeles, Department of Computer Science, Box 951596, 3732D BH, Los Angeles, CA 90095 USA.

§`reyzin@cs.bu.edu`. Boston University, Department of Computer Science, 111 Cummington St., Boston MA 02215 USA.

¶`adam.smith@weizmann.ac.il`. Weizmann Institute of Science, Faculty of Mathematics and Computer Science, Rehovot 76100, Israel. Currently supported by the Louis L. and Anita M. Perlman Postdoctoral Fellowship. The research reported here was done while the author was a student at the Computer Science and Artificial Intelligence Laboratory at MIT.

Contents

1 Introduction

Cryptography traditionally relies on uniformly distributed and precisely reproducible random strings for its secrets. Reality, however, makes it difficult to create, store, and reliably retrieve such strings. Strings that are neither uniformly random nor reliably reproducible seem to be more plentiful. For example, a random person's fingerprint or iris scan is clearly not a uniform random string, nor does it get reproduced precisely each time it is measured. Similarly, a long pass-phrase (or answers to 15 questions [FJ01] or a list of favorite movies [JS02]) is not uniformly random and is difficult to remember for a human user. This work is about using such nonuniform and unreliable secrets in cryptographic applications. Our approach is rigorous and general, and our results have both theoretical and practical value.

To illustrate the use of random strings on a simple example, let us consider the task of password authentication. A user Alice has a password w and wants to gain access to her account. A trusted server stores some information $y = f(w)$ about the password. When Alice enters w, the server lets Alice in only if $f(w) = y$. In this simple application, we assume that it is safe for Alice to enter the password for the verification. However, the server's long-term storage is not assumed to be secure (e.g., y is stored in a publicly readable `/etc/passwd` file in UNIX [MT79]). The goal, then, is to design an efficient f that is hard to invert (i.e., given y it is hard to find w' s.t. $f(w') = y$), so that no one can figure out Alice's password from y. Recall that such functions f are called *one-way functions*.

Unfortunately, the solution above has several problems when used with passwords w available in real life. First, the definition of a one-way function assumes that w is *truly uniform*, and guarantees nothing if this is not the case. However, human-generated and biometric passwords are far from uniform, although they do have some unpredictability in them. Second, Alice has to reproduce her password *exactly* each time she authenticates herself. This restriction severely limits the kinds of passwords that can be used. Indeed, a human can precisely memorize and reliably type in only relatively short passwords, which do not provide an adequate level of security. Greater levels of security are achieved by longer human-generated and biometric passwords, such as pass-phrases, answers to questionnaires, handwritten signatures, fingerprints, retina scans, voice commands, and other values selected by humans or provided by nature, possibly in combination (see [Fry00] for a survey). However, two biometric readings are rarely identical, even though they are likely to be close; similarly, humans are unlikely to precisely remember their answers to multiple question from time to time, though such answers will likely be similar. In other words, the ability to tolerate a (limited) number of errors in the password while retaining security is crucial if we are to obtain greater security than provided by typical user-chosen short passwords.

The password authentication described above is just one example of a cryptographic application where the issues of nonuniformity and error-tolerance naturally come up. Other examples include any cryptographic application, such as encryption, signatures, or identification, where the secret key comes in the form of noisy nonuniform data.

OUR DEFINITIONS. As discussed above, an important general problem is to convert noisy nonuniform inputs into reliably reproducible, uniformly random strings. To this end, we propose a new primitive, termed *fuzzy extractor*. It extracts a uniformly random string R from its input w in a noise-tolerant way. Noise tolerance means that if the input changes to some w' but remains close, the string R can be reproduced exactly. To assist in reproducing R from w', the fuzzy extractor outputs a nonsecret string P. It is important to note R remains uniformly random even given P.

Our approach is general: R extracted from w can be used as a key in a cryptographic application, but, unlike traditional keys, need not be stored (because it can be recovered from any w' that is close to w). We define fuzzy extractors to be *information-theoretically* secure, thus allowing them to be used in

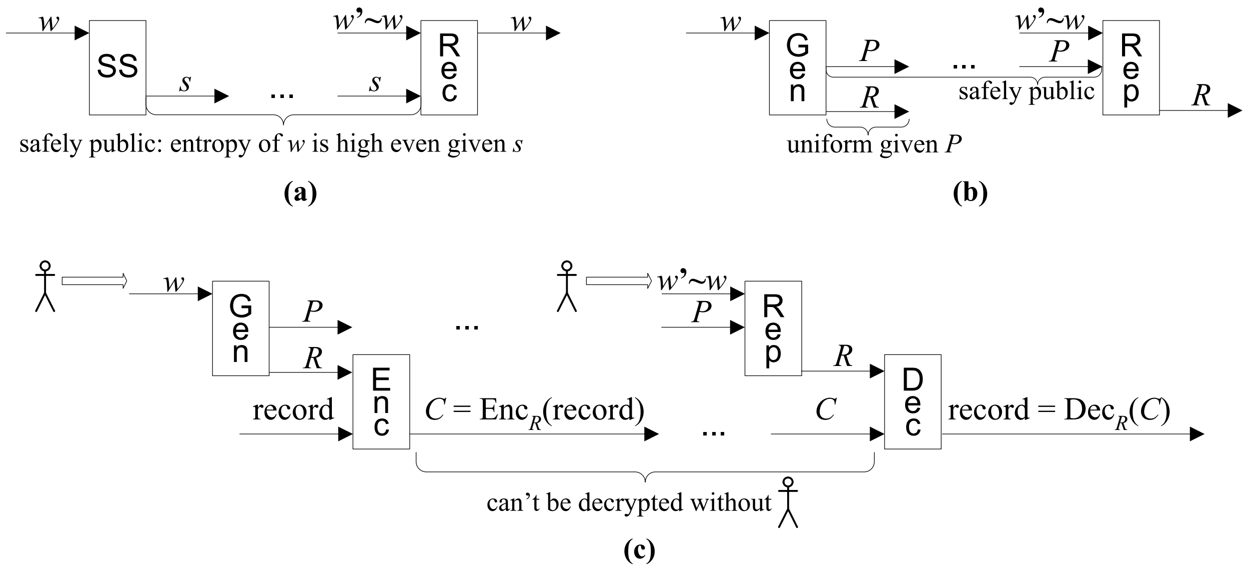


Figure 1: **(a)** secure sketch; **(b)** fuzzy extractor; **(c)** a sample application: user who encrypts a sensitive record using a cryptographically strong, uniform key R extracted from biometric w via a fuzzy extractor; both P and the encrypted record need not be kept secret, because no one can decrypt the record without a w' that is close.

cryptographic systems without introducing additional assumptions (of course, the cryptographic application itself will typically have computational, rather than information-theoretic, security).

For a concrete example of how to use fuzzy extractors, in the password authentication case, the server can store $(P, f(R))$. When the user inputs w' close to w, the server reproduces the actual R using P and checks if $f(R)$ matches what it stores. Similarly, R can be used for symmetric encryption, for generating a public-secret key pair, or any other application.[1]

As a step in constructing fuzzy extractors, and as an interesting object in its own right, we propose another primitive, termed *secure sketch*. It allows precise reconstruction of a noisy input, as follows: on input w, a procedure outputs a sketch s. Then, given s and a value w' close to w, it is possible to recover w. The sketch is secure in the sense it does not reveal much about w: w retains much of its entropy even if s is known. Thus, instead of storing w for fear that later readings will be noisy, it is possible to store s instead, without compromising the privacy of w. A secure sketch, unlike a fuzzy extractor, allows for the precise reproduction of the original input, but does not address nonuniformity.

Secure sketches, fuzzy extractors and a sample encryption application are illustrated in Figure 1.

Secure sketches and extractors can be viewed as providing fuzzy key storage: they allow recovery of the secret key (w or R) from a faulty reading w' of the password w, by using some public information (s or P). In particular, fuzzy extractors can be viewed as error- and nonuniformity-tolerant secret key *key-encapsulation mechanisms* [Sho01].

Because different biometric information has different error patterns, we do not assume any particular notion of closeness between w' and w. Rather, in defining our primitives, we simply assume that w comes

[1] Naturally, the security of the resulting system depends on the possible adversarial attacks; in particular, in this work we do not consider active attacks on P or scenarios in which the adversary can force multiple invocations of the extractor with related w and gets to observe the different P values. See [Boy04, BDK$^+$05] for follow-up work that considers attacks on the fuzzy extractor itself.

from some metric space, and that w' is no more than a certain distance from w in that space. We consider particular metrics only when building concrete constructions.

GENERAL RESULTS. Before proceeding to construct our primitives for concrete metrics, we make some observations about our definitions. We demonstrate that fuzzy extractors can be built out of secure sketches by utilizing (the usual) strong randomness extractors [NZ96], such as, for example, pairwise-independent (also known as universal$_2$) hash functions [CW79, WC81]. We also provide a general technique for constructing secure sketches from transitive families of isometries, which is instantiated in concrete constructions later in the paper. Finally, we define a notion of a *biometric embedding* of one metric space into another, and show that the existence of a fuzzy extractor in the target space implies, combined with a biometric embedding of the source into the target, the existence of a fuzzy extractor in the source space.

These general results help us in building and analyzing our constructions.

OUR CONSTRUCTIONS. We provide constructions of secure sketches and fuzzy extractors in three metrics: Hamming distance, set difference, and edit distance.

Hamming distance (i.e., the number of bit positions that differ between w and w') is perhaps the most natural metric to consider. We observe that the "fuzzy-commitment" construction of Juels and Wattenberg [JW99] based on error-correcting codes can be viewed as a (nearly optimal) secure sketch. We then apply our general result to convert it into a nearly optimal fuzzy extractor. While our results on the Hamming distance essentially use previously known constructions, they serve as an important stepping stone for the rest of the work.

The set difference metric (i.e., size of the symmetric difference of two input sets w and w') is appropriate whenever the noisy input is represented as a subset of features from a universe of possible features.[2] We demonstrate the existence of optimal (with respect to entropy loss) secure sketches and fuzzy extractors for this metric. However, this result is mainly of theoretical interest, because (1) it relies on optimal constant-weight codes, which we do not know how construct and (2) it produces sketches of length proportional to the universe size. We then turn our attention to more efficient constructions for this metric in order handle exponentially large universes. We provide two such constructions.

First, we observe that the "fuzzy vault" construction of Juels and Sudan [JS02] can be viewed as a secure sketch in this metric (and then converted to a fuzzy extractor using our general result). We provide a new, simpler analysis for this construction, which bounds the entropy lost from w given s. This bound is quite high unless one makes the size of the output s very large. We then improve the Juels-Sudan construction to reduce the entropy loss and the length of s to near optimal. Our improvement in the running time and in the length of s is exponential for large universe sizes. However, this improved Juels-Sudan construction retains a drawback of the original: it is able to handle only sets of the same fixed size (in particular, $|w'|$ must equal $|w|$.)

Second, we provide an entirely different construction, called PinSketch, that maintains the exponential improvements in sketch size and running time and also handles variable set size. To obtain it, we note that in the case of a small universe, a set can be simply encoded as its characteristic vector (1 if an element is in the set, 0 if it is not), and set difference becomes Hamming distance. Even though the length of such a vector becomes unmanageable as the universe size grows, we demonstrate that this approach can be made to work quite efficiently even for exponentially large universes (in particular, because it is not necessary to ever actually write the vector down). This involves a result that may be of independent interest: we show

[2] A perhaps unexpected application of the set difference metric was explored in [JS02]: a user would like to encrypt a file (e.g., her phone number) using a small subset of values from a large universe (e.g., her favorite movies) in such a way that those and only those with a similar subset (e.g., similar taste in movies) can decrypt it.

that BCH codes can be decoded in time polynomial in the *weight* of the received corrupted word (i.e., in *sublinear* time if the weight is small).

Finally, edit distance (i.e., the number of insertions and deletions needed to convert one string into the other) comes up, for example, when the password is entered as a string, due to typing errors or mistakes made in handwriting recognition. We discuss two approaches for secure sketches and fuzzy extractors for this metric. First, we observe that a recent low-distortion embedding of Ostrovsky and Rabani [OR05] immediately gives a construction for edit distance. The construction performs well when the number of errors to be corrected is very small (say n^α for $\alpha < 1$) but cannot tolerate a large number of errors. Second, we give a biometric embedding (which is less demanding than a low-distortion embedding, but suffices for obtaining fuzzy extractors) from the edit distance metric into the set difference metric. Composing it with a fuzzy extractor for set difference gives a different construction for edit distance, which does better when t is large; it can handle as many as $O(n/\log^2 n)$ errors with meaningful entropy loss.

Most of the above constructions are quite practical; some implementations are available [HJR].

EXTENDING RESULTS FOR PROBABILISTIC NOTIONS OF CORRECTNESS. The definitions and constructions just described use a very strong error model: we require that secure sketches and fuzzy extractors accept *every* secret w' which is sufficiently close to the original secret w, with probability 1. Such a stringent model is useful, as it makes no assumptions on the stochastic and computational properties of the error process. However, slightly relaxing the error conditions allows constructions which tolerate a (provably) much larger number of errors, at the price of restricting the settings in which the constructions can be applied. In Section 8, we extend the definitions and constructions of earlier sections to several relaxed error models.

It is well-known that in the standard setting of error-correction for a binary communication channel, one can tolerate many more errors when the errors are random and independent than when the errors are determined adversarially. In contrast, we present fuzzy extractors that meet Shannon's bounds for correcting random errors and, moreover, can correct the same number of errors even when errors are adversarial. In our setting, therefore, under a proper relaxation of the correctness condition, adversarial errors are no stronger than random ones. The constructions are quite simple, and draw on existing techniques from the coding literature [BBR88, DGL04, Gur03, Lan04, MPSW05].

RELATION TO PREVIOUS WORK. Since our work combines elements of error correction, randomness extraction and password authentication, there has been a lot of related work.

The need to deal with nonuniform and low-entropy passwords has long been realized in the security community, and many approaches have been proposed. For example, Kelsey et al. [KSHW97] suggested using $f(w, r)$ in place of w for the password authentication scenario, where r is a public random "salt," to make a brute-force attacker's life harder. While practically useful, this approach does not add any entropy to the password, and does not formally address the needed properties of f. Another approach, more closely related to ours, is to add biometric features to the password. For example, Ellison et al. [EHMS00] proposed asking the user a series of n personalized questions, and using these answers to encrypt the "actual" truly random secret R. A similar approach using user's keyboard dynamics (and, subsequently, voice [MRLW01a, MRLW01b]) was proposed by Monrose et al. [MRW99]. These approaches require the design of a secure "fuzzy encryption." The above works proposed heuristic designs (using various forms of Shamir's secret sharing), but gave no formal analysis. Additionally, error tolerance was addressed only by brute force search.

A formal approach to error tolerance in biometrics was taken by Juels and Wattenberg [JW99] (for less formal solutions, see [DFMP99, MRW99, EHMS00]), who provided a simple way to tolerate errors in *uniformly distributed* passwords. Frykholm and Juels [FJ01] extended this solution and provided entropy analysis to which ours is similar. Similar approaches have been explored earlier in seemingly unrelated literature on cryptographic information reconciliation, often in the context of quantum cryptography (where Alice and Bob wish to derive a secret key from secrets that have small Hamming distance), particularly [BBR88, BBCS91]. Our construction for the Hamming distance is essentially the same as a component of the quantum oblivious transfer protocol of [BBCS91].

Juels and Sudan [JS02] provided the first construction for a metric other than Hamming: they constructed a "fuzzy vault" scheme for the set difference metric. The main difference is that [JS02] lacks a cryptographically strong definition of the object constructed. In particular, their construction leaks a significant amount of information about their analog of R, even though it leaves the adversary with provably "many valid choices" for R. In retrospect, their informal notion is closely related to our secure sketches. Our constructions in Section 6 improve exponentially over the construction of [JS02] for storage and computation costs, in the setting when the set elements come from a large universe.

Linnartz and Tuyls [LT03] defined and constructed a primitive very similar to a fuzzy extractor (that line of work was continued in [VTDL03].) The definition of [LT03] focuses on the continuous space $\mathbb{R}^n$, and assumes a particular input distribution (typically a known, multivariate Gaussian). Thus, our definition of a fuzzy extractor can be viewed as a generalization of the notion of a "shielding function" from [LT03]. However, our constructions focus on discrete metric spaces.

Other approaches have also been taken for guaranteeing the privacy of noisy data. Csirmaz and Katona [CK03] considered quantization for correcting errors in "physical random functions." (This corresponds roughly to secure sketches with no public storage.) Barral, Coron and Naccache [BCN04] proposed a system for offline, private comparison of fingerprints. Although seemingly similar, the problem they study is complementary to ours, and the two solutions can be combined to yield systems which enjoy the benefits of both.

Work on privacy amplification, e.g., [BBR88, BBCM95], as well as work on derandomization and hardness amplification, e.g., [HILL99, NZ96], also addressed the need to extract uniform randomness from a random variable about which some information has been leaked. A major focus of follow-up research has been the development of (ordinary, not fuzzy) extractors with short seeds (see [Sha02] for a survey). We use extractors in this work (though for our purposes, pairwise independent hashing is sufficient). Conversely, our work has been applied recently to privacy amplification: Ding [Din05] used fuzzy extractors for noise tolerance in Maurer's bounded storage model [Mau93].

Independently of our work, similar techniques appeared in the literature on noncryptographic information reconciliation [MTZ03, CT04] (where the goal is communication efficiency rather than secrecy). The surprising relationship between secure sketches and efficient information reconciliation is explored further in Section 9, which discusses, in particular, how our secure sketches for set differences provide more efficient solutions to the set and string reconciliation problems.

2 Preliminaries

Unless explicitly stated otherwise, all logarithms below are base 2. The *Hamming weight* (or just *weight*) of a string is the number of nonzero characters in it. We use U_ℓ to denote the uniform distribution on ℓ-bit binary strings. If an algorithm (or a function) f is randomized, we use the semicolon when we wish to make the randomness explicit: i.e., we denote by $f(x;r)$ the result of computing f on input x with randomness r. If X is a probability distribution, then $f(X)$ is the distribution induced on the image of f by applying the (possibly probabilistic) function f. If X is a random variable, we will (slightly) abuse notation and also denote by X the probability distribution on the range of the variable.

MIN-ENTROPY, STATISTICAL DISTANCE, AND STRONG EXTRACTORS. When discussing security, one is often interested in the probability that the adversary predicts a random value (e.g., guesses a secret key). The adversary's best strategy, of course, is to guess the most likely value. Thus, *predictability* of a random variable A is $\max_a \Pr[A = a]$, and, correspondingly, *min-entropy* $\mathbf{H}_\infty(A)$ is $-\log(\max_a \Pr[A = a])$ (min-entropy can thus be viewed as the "worst-case" entropy [CG88]).

Min-entropy of a distribution tells us how many nearly uniform random bits can be extracted from it. The notion of "nearly" is defined as follows. The *statistical distance between* two probability distributions A and B is $\mathbf{SD}(A, B) = \frac{1}{2}\sum_v |\Pr(A = v) - \Pr(B = v)|$.

Recall the definition of *strong randomness extractors* [NZ96].

Definition 1. Let X be a polynomial time probabilistic function $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^\ell$ with random input of length r. It is an efficient (n, m, ℓ, ϵ)-*strong extractor* if for all min-entropy m distributions W, $\mathbf{SD}((\mathsf{Ext}(W; X), X), (U_\ell, X)) \leq \epsilon$, where X is uniform on $\{0,1\}^r$.

Strong extractors can extract at most $\ell = m - 2\log\left(\frac{1}{\epsilon}\right) + O(1)$ nearly random bits [RTS00]. Many constructions match this bound (see Shaltiels' survey [Sha02] for references). Extractor constructions are often complex since they seek to minimize the length of the seed X. For our purposes, the length of X will be less important, so pairwise independent hash functions will already give us the optimal $\ell = m - 2\log\left(\frac{1}{\epsilon}\right) + 2$ (see Lemma 4.1).

METRIC SPACES. A metric space is a set $\mathcal{M}$ with a distance function $\mathsf{dis} : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^+ = [0, \infty)$. For purposes of this of work, $\mathcal{M}$ will always be a finite set, the distance function will only take on integer values (with $\mathsf{dis}(x, y) = 0$ if and only if $x = y$), and will obey symmetry $\mathsf{dis}(x, y) = \mathsf{dis}(y, x)$ and the triangle inequality $\mathsf{dis}(x, z) \leq \mathsf{dis}(x, y) + \mathsf{dis}(y, z)$ (we adopt these requirements for simplicity of exposition, even though the definitions and most of the results below can be generalized to remove these restrictions).

We will concentrate on the following metrics.

1. *Hamming metric*. Here $\mathcal{M} = \mathcal{F}^n$ for some alphabet $\mathcal{F}$, and $\mathsf{dis}(w, w')$ is the number of positions in which the strings w and w' differ.

2. *Set difference metric*. Here $\mathcal{M}$ consists of all subsets of a universe $\mathcal{U}$. For two sets w, w', their symmetric difference $w \triangle w' \stackrel{\text{def}}{=} \{x \in w \cup w' \mid x \notin w \cap w'\}$. The distance between two sets w, w' is $|w \triangle w'|$. [3] We will sometimes restrict $\mathcal{M}$ to contain only s-element subsets for some s.

3. *Edit metric*. Here $\mathcal{M} = \mathcal{F}^*$, and the distance between w and w' is defined to be the smallest number of character insertions and deletions needed to transform w into w'. [4] (This is different from the Hamming metric because insertions and deletions shift the characters that are to the right of the insertion/deletion point.)

As already mentioned, all three metrics seem natural for biometric data.

CODES AND SYNDROMES. Since we want to achieve error tolerance in various metric spaces, we will use *error-correcting codes* for a particular metric. A code C is a subset $\{w_0, \ldots, w_{K-1}\}$ of K elements of $\mathcal{M}$. The map from i to w_i, which we will also sometimes denote by C, is called *encoding*. The *minimum*

[3] In the preliminary version of this work [DRS04], we worked with this metric scaled by $\frac{1}{2}$, that is the distance was $\frac{1}{2}|w \triangle w'|$. Not scaling makes more sense, particularly when w and w' are of potentially different sizes since $|w \triangle w'|$ may be odd. It also agrees with the hamming distance of characteristic vectors; see Section 6.

[4] Again, in [DRS04], we worked with this metric scaled by $\frac{1}{2}$. Likewise, this makes little sense when strings can be of different lengths, and we avoid it here.

distance of C is the smallest $d > 0$ such that for all $i \neq j$ we have $\mathsf{dis}(w_i, w_j) \geq d$. In our case of integer metrics, this means that one can detect up to $(d-1)$ "errors" in an element of $\mathcal{M}$. The *error-correcting distance* of C is the largest number $t > 0$ such that for every $w \in \mathcal{M}$ there exists at most one codeword c in the ball of radius t around w: $\mathsf{dis}(w, c) \leq t$ for at most one $c \in C$. This means that one can correct up to t errors in an w element of $\mathcal{M}$; we will use the term *decoding* for the map that finds, given w, the $c \in C$ such that $\mathsf{dis}(w, c) \leq t$ (note that for some w, such c may not exist, but if it exists, it will be unique; note also that decoding is not the inverse of encoding in our terminology). For integer metrics by triangle inequality we are guaranteed that $t \geq \lfloor (d-1)/2 \rfloor$. Since error correction will be more important than error detection in our applications, we denote the corresponding codes as $(\mathcal{M}, K, t)$-codes. For efficiency purposes, we will often want encoding and decoding to be polynomial-time.

For the Hamming metric over $\mathcal{F}^n$, we will sometimes call $k = \log_{|\mathcal{F}|} K$ the *dimension* of the code, and denote the code itself as an $[n, k, d = 2t+1]_{\mathcal{F}}$-code, following the standard notation in the literature. We will denote by $A_{|\mathcal{F}|}(n, d)$ the maximum K possible in such a code (omitting the subscript when $|\mathcal{F}| = 2$), and by $A(n, d, s)$ the maximum K for such a code over $\{0, 1\}^n$ with the additional restriction that all codewords have exactly s ones.

If the code is linear (i.e., $\mathcal{F}$ is a field, $\mathcal{F}^n$ is a vector space over $\mathcal{F}$ and C is a linear subspace), then one can fix a parity-check matrix H as any matrix whose rows generate the orthogonal space $C^{\perp}$. Then for any $v \in \mathcal{F}^n$, the syndrome $\mathsf{syn}(v) \stackrel{\text{def}}{=} Hv$. The syndrome of a vector is its projection onto subspace that is orthogonal to the code, and can thus be intuitively viewed as the vector modulo the code. Note that $v \in C \Leftrightarrow \mathsf{syn}(v) = 0$. Note also that M is an $(n-k) \times n$ matrix, and that $\mathsf{syn}(v)$ is $n-k$ bits long.

The syndrome captures all the information necessary for decoding. That is, suppose a codeword c is sent through a channel and the word $w = c + e$ is received. First, the syndrome of w is the syndrome of e: $\mathsf{syn}(w) = \mathsf{syn}(c) + \mathsf{syn}(e) = 0 + \mathsf{syn}(e) = \mathsf{syn}(e)$. Moreover, for any value u, there is at most one word e of weight less than $d/2$ such that $\mathsf{syn}(e) = u$ (because the existence of a pair of distinct words e_1, e_2 would mean that $e_1 - e_2$ is a codeword of weight less than d, but since 0^n is also a codeword and the minimum distance of the code is d, this is impossible). Thus, knowing syndrome $\mathsf{syn}(w)$ is enough to determine the error pattern e if not too many errors occurred.

3 New Definitions

3.1 Average Min-Entropy

Recall that *predictability* of a random variable A is $\max_a \Pr[A = a]$, and its *min-entropy* $\mathbf{H}_\infty(A)$ is $-\log(\max_a \Pr[A = a])$. Consider now a pair of (possibly correlated) random variables A, B. If the adversary finds out the value b of B, then predictability of A becomes $\max_a \Pr[A = a \mid B = b]$. On average, the adversary's chance of success in predicting A is then $\mathbb{E}_{b \leftarrow B}[\max_a \Pr[A = a \mid B = b]]$. Note that we are taking the *average* over B (which is not under adversarial control), but the *worst case* over A (because prediction of A is adversarial once b is known). Again, it is convenient to talk about security in log-scale, which is why define the *average min-entropy* of A given B as simply the logarithm of the above:

$$\tilde{\mathbf{H}}_\infty(A \mid B) \stackrel{\text{def}}{=} -\log\left(\mathbb{E}_{b \leftarrow B}\left[\max_a \Pr[A = a \mid B = b]\right]\right) = -\log\left(\mathbb{E}_{b \leftarrow B}\left[2^{-\mathbf{H}_\infty(A|B=b)}\right]\right).$$

Because other notions of entropy have been studied in cryptographic literature, a few words are in order to explain why this definition is the "right one." Note the importance of taking the logarithm *after* taking the average (in contrast, for instance, to conditional Shannon entropy). One may think it more natural to define average min-entropy as $\mathbb{E}_{b \leftarrow B}[\mathbf{H}_\infty(A \mid B=b)]$, thus reversing the order of $\log$ and $\mathbb{E}$. However, this notion is unlikely to be useful in a security application. For a simple example, consider the case when A and B are 1000-bit strings distributed as follows: $B = U_{1000}$ and A is equal the value b of B if the first bit of b is 0, and U_{1000} (independent of B) otherwise. Then for half of the values of b, $\mathbf{H}_\infty(A \mid B=b) = 0$, while for the other half, $\mathbf{H}_\infty(A \mid B=b) = 1000$, so $\mathbb{E}_{b \leftarrow B}[\mathbf{H}_\infty(A \mid B=b)] = 500$. However, it would be obviously incorrect to say that A has 500 bits of security. In fact, an adversary who knows the value b of B has a slightly greater than 50% chance of predicting the value of A by outputting b. Our definition correctly captures this 50% chance of prediction, because $\tilde{\mathbf{H}}_\infty(A \mid B)$ is slightly less than 1. In fact, our definition of average min-entropy is simply the logarithm of predictability.

Lemma 4.2 justifies our definition further by demonstrating that almost m nearly-uniform bits can be extracted from a random variable whose average min-entropy is m. Such nearly-uniform random bits can be used in any cryptographic context that requires uniform random bits (e.g., for secret keys), reducing security by at most their distance from uniform.

The following simple lemma will be helpful later.

Lemma 3.1. *If B has 2^λ possible values, then $\tilde{\mathbf{H}}_\infty(A \mid B) \geq \mathbf{H}_\infty((A,B)) - \lambda \geq \mathbf{H}_\infty(A) - \lambda$. Furthermore, for all random variables C, $\tilde{\mathbf{H}}_\infty(A \mid (B,C)) \geq \tilde{\mathbf{H}}_\infty((A,B) \mid C) - \lambda \geq \tilde{\mathbf{H}}_\infty(A \mid C) - \lambda$.*

Proof. It suffices to prove the second sentence (the first follows from taking C to be constant).

$$\begin{aligned}
\tilde{\mathbf{H}}_\infty(A \mid (B,C)) &= -\log \mathbb{E}_{(b,c) \leftarrow (B,C)} \left[\max_a \Pr[A=a \mid B=b \wedge C=c]\right] \\
&= -\log \sum_{(b,c)} \max_a \Pr[A=a \mid B=b \wedge C=c] \Pr[B=b \wedge C=c] \\
&= -\log \sum_{(b,c)} \max_a \Pr[A=a \wedge B=b \wedge C=c] \\
&= -\log \sum_b \mathbb{E}_{c \leftarrow C} \left[\max_a \Pr[A=a \wedge B=b \mid C=c]\right] \\
&= -\log \sum_b 2^{-\tilde{\mathbf{H}}_\infty((A,B)\mid C)} = -\log 2^\lambda 2^{-\tilde{\mathbf{H}}_\infty((A,B)\mid C)} = \tilde{\mathbf{H}}_\infty((A,B) \mid C) - \lambda\,.
\end{aligned}$$

The second inequality follows from $\Pr[A=a \wedge B=b \mid C=c] \leq \Pr[A=a \mid C=c]$. □

See Appendix C for a generalization of average min-entropy and a discussion of the relationship between this notion and other notions of entropy.

3.2 Secure Sketches

Let $\mathcal{M}$ be a metric space with distance functions dis.

Definition 2. An $(\mathcal{M}, m, \tilde{m}, t)$-*secure sketch* is a pair of randomized procedures, "sketch" (SS) and "recover" (Rec), with the following properties:

1. The sketching procedure SS on input $w \in \mathcal{M}$ returns a bit string $s \in \{0,1\}^*$.

2. The recovery procedure Rec takes an element $w' \in M$ and a bit string $s \in \{0,1\}^*$. The *correctness* property of secure sketches guarantees that if $\mathsf{dis}(w,w') \leq t$, then $\mathsf{Rec}(w', \mathsf{SS}(w)) = w$. If $\mathsf{dis}(w,w') > t$, then no guarantee is provided about the output of Rec.

3. The *security* property guarantees that for any distribution W over $\mathcal{M}$ with min-entropy m, the value of W can be recovered by the adversary who observes s with probability no greater than $2^{-\tilde{m}}$. That is, $\tilde{\mathbf{H}}_\infty(W \mid \mathsf{SS}(W)) \geq \tilde{m}$.

A secure sketch is *efficient* if SS and Rec run in expected polynomial time.

The quantity $m - \tilde{m}$ is called the *entropy loss* of a secure sketch. In analyzing the security of our secure sketch constructions below, we will typically bound the entropy loss regardless of m, thus obtaining families of secure sketches that work for all m. It should be also pointed out (Lemma B.1) that a secure sketch with entropy loss λ for a particular m will have at most the same entropy loss for any $m' < m$. Unfortunately, this statement does not always hold for $m' > m$. This necessitates the slightly stronger definition we discuss next.

It may well be that adversary's information i about the password w is probabilistic, so that sometimes i reveals a lot about w, but most of the time w stays hard to predict even given i. In this case, the previous definition of secure sketch is hard to apply: it provides no guarantee if $\mathbf{H}_\infty(W|i)$ is not fixed to at least m for some bad values of i. A more robust definition would provide the same guarantee for all pairs of variables (W, I) such that predicting the value of W given the value of I is hard. We therefore define an *average-case* secure sketch as a secure sketch with the augmented security property: for any random variables W over $\mathcal{M}$ and I over $\{0,1\}^*$ such that $\tilde{\mathbf{H}}_\infty(W \mid I) \geq m$, we have $\tilde{\mathbf{H}}_\infty(W \mid (\mathsf{SS}(W), I)) \geq \tilde{m}$. Note that an average-case secure sketch is also a secure sketch (take I to be empty).

This definition has the advantage that it composes naturally, as shown in Lemma 4.5. All of our constructions will in fact be average-case secure sketches. However, we will generally omit the term "average-case" and the variable I for simplicity of exposition, unless the average-case property is important in the particular context.

3.3 Fuzzy Extractors

Definition 3. An $(\mathcal{M}, m, \ell, t, \epsilon)$ *fuzzy extractor* is a pair of randomized procedures, "generate" (Gen) and "reproduce" (Rep), with the following properties

1. The generation procedure Gen on input $w \in \mathcal{M}$ outputs an extracted string $R \in \{0,1\}^\ell$ and a helper string $P \in \{0,1\}^*$.

2. The reproduction procedure Rep takes an element $w' \in M$ and a bit string $P \in \{0,1\}^*$ as inputs. The *correctness* property of fuzzy extractors guarantees that if $\mathsf{dis}(w, w') \leq t$ and R, P were generated by $(R, P) \leftarrow \mathsf{Gen}(w)$, then $\mathsf{Rep}(w', P) = R$. If $\mathsf{dis}(w, w') > t$, then no guarantee is provided about the output of Rep.

3. The *security* property guarantees that for any distribution W on $\mathcal{M}$ of min-entropy m, the string R is close to uniform even to those who observe P: namely, if $(R, P) \leftarrow \mathsf{Gen}(W)$, then $\mathbf{SD}\left((R, P), (U_\ell, P)\right) \leq \epsilon$.

A fuzzy extractor is *efficient* if Gen and Rep run in expected polynomial time.

In other words, fuzzy extractors allow one to extract some randomness R from w and then successfully reproduce R from any string w' that is close to w. The reproduction is done with the help of the helper string P produced during the initial extraction; yet P need not remain secret, because R looks truly random even given P. To justify our terminology, notice that strong extractors (as defined in Section 2) can indeed be seen as "nonfuzzy" analogs of fuzzy extractors, corresponding to $t = 0$, $P = X$, and $\mathcal{M} = \{0,1\}^n$.

We reiterate that the nearly-uniform random bits output by a fuzzy extractor can be used in any cryptographic context that requires uniform random bits (e.g., for secret keys). The slight nonuniformity of the bits may decrease security, but by no more than their distance ϵ from uniform. By choosing ϵ sufficiently small (e.g., 2^{-100}) one can make the decrease in security irrelevant.

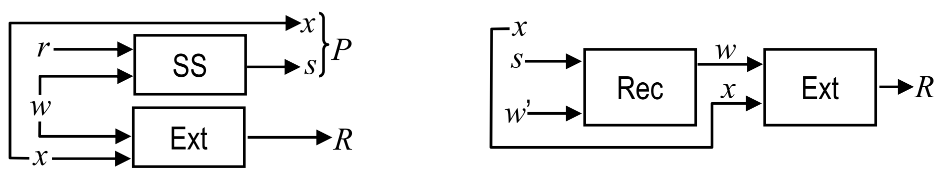
Similarly to secure sketches, the quantity $m - \ell$ is called the *entropy loss* of a fuzzy extractor. Also similarly, a more robust definition is that of an *average-case* fuzzy extractor, which requires that if $\mathbf{H}_\infty(W \mid I) \geq m$, then $\mathbf{SD}\left((R, P, I), (U_\ell, P, I)\right) \leq \epsilon$.

4 Metric-Independent Results

In this section we demonstrate some general results that do not depend on specific metric spaces. They will be helpful in obtaining specific results for particular metric spaces below.

4.1 Construction of Fuzzy Extractors from Secure Sketches

Not surprisingly, secure sketches are quite useful in constructing fuzzy extractors. Specifically, we construct fuzzy extractors from secure sketches and strong extractors as follows: apply SS to w to obtain s, and a strong extractor Ext with randomness x to w obtain R. Store (s, x) as the helper string P. To reproduce R from w' and $P = (s, x)$, first use $\mathsf{Rec}(w', s)$ to recover w and then $\mathsf{Ext}(w, x)$ to get R.



In order to apply Ext to w, we will assume that one can represent elements of $\mathcal{M}$ using n bits. The strong extractor Ext we use is the standard pairwise-independent hashing [CW79, WC81], i.e., a family functions $\{H_x : \{0,1\}^n \to \{0,1\}^\ell\}_{x \in X}$, such that for all $a, b \in \{0,1\}^n$, $\Pr_{x \in X}[H_x(a) = H_x(b)] = 2^{-\ell}$. Such an extractor has (optimal) entropy loss $2 \log\left(\frac{1}{\epsilon}\right) - O(1)$. The entropy loss of the resulting fuzzy extractor is equal to the entropy loss of the secure sketch plus the entropy loss of the extractor. This further justifies the definition of "average min-entropy," demonstrating that it connects to actual extractable uniform randomness: a pairwise-independent hash function extracts all of the entropy left after the secure sketch is applied (except for the necessary $2 \log\left(\frac{1}{\epsilon}\right)$ loss due to the fundamental constrains on extractors [RTS00]).

Lemma 4.1 (Fuzzy Extractors from Sketches). *Assume* $(\mathsf{SS}, \mathsf{Rec})$ *is an* $(\mathcal{M}, m, \tilde{m}, t)$*-secure sketch, and let* Ext *be the* $(n, \tilde{m}, \ell, \epsilon)$*-strong extractor given by pairwise-independent hashing (in particular,* $\ell = \tilde{m} - 2 \log\left(\frac{1}{\epsilon}\right) + 2$*). Then the following* $(\mathsf{Gen}, \mathsf{Rep})$ *is a* $(\mathcal{M}, m, \ell, t, \epsilon)$*-fuzzy extractor:*

- $\mathsf{Gen}(w; r, x)$*: set* $P = (\mathsf{SS}(w; r), x)$, $R = \mathsf{Ext}(w; x)$*, and output* (R, P).
- $\mathsf{Rep}(w', (s, x))$*: recover* $w = \mathsf{Rec}(w', s)$ *and output* $R = \mathsf{Ext}(w; x)$

Proof. Lemma 4.1 follows directly from the intermediate result below (Lemma 4.2), which explains our choice of the measure $\tilde{\mathbf{H}}_\infty(A \mid B)$ for the average min-entropy. Lemma 4.2 says that pairwise independent hashing extracts randomness from the random variable A as if the min-entropy of A given $B = b$ were always *at least* (rather than *on average*) $\tilde{\mathbf{H}}_\infty(A \mid B)$. □

Lemma 4.2. *If A, B are random variables such that $A \in \{0,1\}^n$ and $\tilde{\mathbf{H}}_\infty(A \mid B) \geq \tilde{m}$, and $\{H_x\}_{x \in X}$ is a family of pairwise independent hash function from n bits to ℓ bits, then $\mathbf{SD}\left((B, X, H_X(A))\ ,\ (B, X, U_\ell)\right) \leq \epsilon$ as long as $\ell \leq \tilde{m} - 2\log\left(\frac{1}{\epsilon}\right) + 2$.*

Proof. The particular extractor we chose (namely, pairwise independent hashing) has a smooth tradeoff between the entropy of the input and the quality of the output. For any random variable C, the leftover hash lemma (see [HILL99, Lemma 1], as well as references therein for earlier versions) states:

$$\mathbf{SD}\left((X, H_X(C))\ ,\ (X, U_\ell)\right) \leq \frac{1}{2}\sqrt{2^{-\mathbf{H}_\infty(C)}2^\ell}$$

(in [HILL99], the lemma is formulated in terms of Rényi entropy of order two of C; the change to $\mathbf{H}_\infty(C)$ is allowed because the latter is no greater than the former).

In our setting we have a bound on the *expected* value of $2^{-\mathbf{H}_\infty(A|B=b)}$, namely $\mathbb{E}_b\left[2^{-\tilde{\mathbf{H}}_\infty(A|B=b)}\right] \leq 2^{-\tilde{m}}$. Using the fact that $\mathbb{E}\left[\sqrt{Z}\right] \leq \sqrt{\mathbb{E}\left[Z\right]}$ (by Jensen's inequality), we get:

$$\mathbb{E}_b\left[\mathbf{SD}\left((X, H_X(A \mid B = b))\ ,\ (X, U_\ell)\right)\right] \leq \frac{1}{2}\sqrt{2^{\ell-\tilde{m}}}.$$

Now the distance of $(B, X, H_X(A))$ from (B, X, U_ℓ) is the average over values b of B of the distance of $(X, H_X(A \mid B = b))$ from (X, U_ℓ):

$$\begin{aligned}
\mathbf{SD}\left((B, X, H_X(A)), (B, X, U_\ell)\right) &= \tfrac{1}{2}\textstyle\sum_b \sum_{x,s} \mid \Pr[X = x \wedge H_x(A) = s \wedge B = b] \\
&\qquad - \Pr[X = x \wedge U_\ell = s \wedge B = b] \mid \\
&= \tfrac{1}{2}\textstyle\sum_b \sum_{x,s} \mid \Pr[X = x \wedge H_x(A) = s \mid B = b] \\
&\qquad - \Pr[X = x \wedge U_ell = s \mid B = b] \mid \Pr[B = b] \\
&= \mathbb{E}_b\left[\mathbf{SD}\left((X, H_X(A \mid B = b)), (X, U_\ell)\right)\right] \\
&\leq \tfrac{1}{2}\sqrt{2^{\ell-\tilde{m}}}\,.
\end{aligned}$$

The extractor we use always has $\ell \leq \tilde{m} - 2\log\left(\frac{1}{\epsilon}\right) + 2$, and so the statistical distance is at most ϵ. ☐

Remark 1. The advantage of pairwise independent hashing over a general extractor is the convex tradeoff between the entropy of the input and the distance from uniform of the output, which we exploited in the proof above (in other words, we used properties of pairwise independent hashing for inputs whose entropy was different from $\tilde{m}$). However, one can prove an analog to Lemma 4.2 using any $\left(n, \tilde{m} - \log\left(\frac{1}{\epsilon_2}\right), \ell, \epsilon\right)$-strong extractor for some ϵ_2. In this general case, without further assumptions on the extractor, the resulting reduction leads to $(\mathcal{M}, m, \ell, t, \epsilon + \epsilon_2)$ fuzzy extractors. Note that because of fundamental bounds on extractors, $\tilde{m} - \log\left(\frac{1}{\epsilon_2}\right) - \ell \geq 2\log\left(\frac{1}{\epsilon}\right) - O(1)$, so the resulting fuzzy extractor's entropy loss will be at least $2\log\left(\frac{1}{\epsilon}\right) + \log\left(\frac{1}{\epsilon_2}\right) - O(1)$ rather than $2\log\left(\frac{1}{\epsilon}\right) - O(1)$, and the quality of extracted randomness will be lower, because it will be $\epsilon + \epsilon_2$ rather than ϵ away from uniform. The proof simply uses Markov's inequality, as follows: $\tilde{\mathbf{H}}_\infty(A \mid B) \geq \tilde{m}$ implies that the event $\mathbf{H}_\infty(A \mid B = b) \geq \tilde{m} - \log\left(\frac{1}{\epsilon}\right)$ happens with probability at least $1 - \epsilon_2$, so in all but ϵ_2 fraction of the cases, the application of the extractor will produce the desired results. Similarly, one can prove this by going through the definition of smooth conditional min-entropy; see Appendix C.

Remark 2. The same result (with the same proof) holds for building average-case fuzzy extractors from average-case secure sketches.

4.2 Secure Sketches for Transitive Metric Spaces

We give a general technique for building secure sketches in *transitive* metric spaces, which we now define. A permutation π on a metric space $\mathcal{M}$ is an *isometry* if it preserves distances, i.e. $\mathsf{dis}(a,b) = \mathsf{dis}(\pi(a), \pi(b))$. A family of permutations $\Pi = \{\pi_i\}_{i \in \mathcal{I}}$ acts *transitively* on $\mathcal{M}$ if for any two elements $a, b \in \mathcal{M}$, there exists $\pi_i \in \Pi$ such that $\pi_i(a) = b$. Suppose we have a family Π of transitive isometries for $\mathcal{M}$ (we will call such $\mathcal{M}$ *transitive*). For example, in the Hamming space, the set of all shifts $\pi_x(w) = w \oplus x$ is such a family (see Section 5 for more details on this example).

Construction 1 (Secure Sketch For Transitive Metric Spaces). Let C be an $(\mathcal{M}, K, t)$-code. Then the general sketching scheme SS is the following: given an input $w \in \mathcal{M}$, pick uniformly at random a codeword $b \in C$, pick uniformly at random a permutation $\pi \in \Pi$ such that $\pi(w) = b$, and output $\mathsf{SS}(w) = \pi$ (it is crucial that each $\pi \in \Pi$ should have a canonical description that is independent of how π was chosen, and in particular independent of b and w; the number of possible outputs of SS should thus be $|\Pi|$). The recovery procedure Rec to find w given w' and the sketch π, is as follows: find the closest codeword b' to $\pi(w')$, and output $\pi^{-1}(b')$.

Let Γ be the number of elements $\pi \in \Pi$ such that $\min_{w,b} |\{\pi | \pi(w) = b\}| \geq \Gamma$. I.e., for each w and b, there are at least Γ choices for π. Then we obtain the following lemma.

Lemma 4.3. $(\mathsf{SS}, \mathsf{Rec})$ *is an* $(\mathcal{M}, m, m - \log|\Pi| + \log\Gamma + \log K, t)$*-secure sketch. It is efficient if operations on the code, as well as* π *and* π^{-1}*, can be implemented efficiently.*

Proof. Correctness is clear: when $\mathsf{dis}(w, w') \leq t$, then $\mathsf{dis}(b, \pi(w')) \leq t$, so decoding $\pi(w')$ will result in $b' = b$, which in turn means that $\pi^{-1}(b') = w$. The intuitive argument for security is as follows: we add $\log K + \log\Gamma$ bits of entropy by choosing b and π, and subtract $\log|\Pi|$ by publishing π. Since given π, w and b determine each other, the total entropy loss is $\log|\Pi| - \log K - \log\Gamma$. More formally, $\tilde{\mathbf{H}}_\infty(W \mid \mathsf{SS}(W)) = \mathbf{H}_\infty((W, \mathsf{SS}(W))) - \log|\Pi|$ by Lemma 3.1. Given a particular value of w, there are K equiprobable choices for b, and further at least Γ equiprobable choices for π once b is picked, and hence any given permutation π is chosen with probability at most $1/(K\Gamma)$ (because different choices for b result in different choices for π). Therefore, for all w and π, $\Pr[W = w \wedge \mathsf{SS}(w) = \pi] \leq \Pr[W = w]/(K\Gamma)$, hence $\mathbf{H}_\infty((W, \mathsf{SS}(W))) \geq \mathbf{H}_\infty(W) + \log K + \log\Gamma$. □

Naturally, security loss will be smaller if the code C is denser.

We will discuss concrete instantiations of this approach in Section 5 and Section 6.1.

4.3 Changing Metric Spaces via Biometric Embeddings

We now introduce a general technique that allows one to build fuzzy extractors and secure sketches in some metric space $\mathcal{M}_1$ from fuzzy extractors and secure sketches in some other metric space $\mathcal{M}_2$. Below, we let $\mathsf{dis}(\cdot,\cdot)_i$ denote the distance function in $\mathcal{M}_i$. The technique is to *embed* $\mathcal{M}_1$ into $\mathcal{M}_2$ so as to "preserve" relevant parameters for fuzzy extraction.

Definition 4. A function $f : \mathcal{M}_1 \to \mathcal{M}_2$ is called a (t_1, t_2, m_1, m_2)-biometric embedding if the following two conditions hold:

- for any $w_1, w_1' \in \mathcal{M}_1$ such that $\mathsf{dis}(w_1, w_1')_1 \leq t_1$, we have $\mathsf{dis}(f(w_1), f(w_2))_2 \leq t_2$.
- for any distribution W_1 on $\mathcal{M}_1$ of min-entropy at least m_1, $f(W_1)$ has min-entropy at least m_2.

The following lemma is immediate (correctness of the resulting fuzzy extractor follows from the first condition, and security follows from the second):

Lemma 4.4. *If f is a (t_1, t_2, m_1, m_2)-biometric embedding of $\mathcal{M}_1$ into $\mathcal{M}_2$ and $(\mathsf{Gen}(\cdot), \mathsf{Rep}(\cdot,\cdot))$ is an $(\mathcal{M}_2, m_2, \ell, t_2, \epsilon)$-fuzzy extractor, then $(\mathsf{Gen}(f(\cdot)), \mathsf{Rep}(f(\cdot), \cdot))$ is an $(\mathcal{M}_1, m_1, \ell, t_1, \epsilon)$-fuzzy extractor.*

It is easy to define *average-case* biometric embeddings (in which $\tilde{\mathbf{H}}_\infty(W_1 \mid I) \geq m_1 \Rightarrow \tilde{\mathbf{H}}_\infty(f(W_1) \mid I) \geq m_2$), which would result in an analogous lemma for average-case fuzzy extractors.

For a similar result to hold for secure sketches, we need biometric embeddings with an additional property.

Definition 5. A function $f : \mathcal{M}_1 \to \mathcal{M}_2$ is called a (t_1, t_2, λ)-*biometric embedding with recovery information g* if:

- for any $w_1, w_1' \in \mathcal{M}_1$ such that $\mathsf{dis}(w_1, w_1')_1 \leq t_1$, we have $\mathsf{dis}(f(w_1), f(w_2))_2 \leq t_2$.
- $g : M_1 \to \{0,1\}^*$ is a function with range size 2^λ, and $w_1 \in M_1$ is uniquely determined by $(f(w_1), g(w_1))$.

The following lemma is also immediate (correctness follows from the first condition on biometric embeddings, and security follows from Lemma 3.1).

Lemma 4.5. *Let f be (t_1, t_2, λ) biometric embedding with recovery information g. Let $(\mathsf{SS}, \mathsf{Rec})$ be $(\mathcal{M}_2, m_1 - \lambda, \tilde{m}_2, t_2)$ average-case secure sketch. Let $\mathsf{SS}'(w) = (\mathsf{SS}(f(w)), g(w))$. Let $\mathsf{Rec}'(w', (s, r))$ be the function obtained by computing $\mathsf{Rec}(w', s)$ to get $f(w)$ and then inverting $(f(w), r)$ to get w. Then $(\mathsf{SS}', \mathsf{Rec}')$ is a $(\mathcal{M}_1, m_1, \tilde{m}_2, t_1)$ average-case secure sketch.*

It should be noted that a similar lemma does not hold without the average-case qualifier on $(\mathsf{SS}, \mathsf{Rec})$, because the adversary receives extra information $g(w)$. In some cases, this may reduce the entropy of $f(W)$ below $m_1 - \lambda$, thus rendering non-average-case secure sketches inapplicable. Average-case secure sketches suffice, because *average* min-entropy $\tilde{\mathbf{H}}_\infty(f(W) \mid g(W)) \geq m_1 - \lambda$ by Lemma 3.1.

We will see the utility of this novel type of embedding in Section 7.

5 Constructions for Hamming Distance

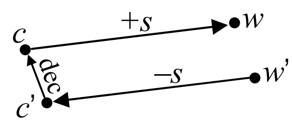
In this section we consider constructions for the space $\mathcal{M} = \mathcal{F}^n$ under the Hamming distance metric. Let $F = |\mathcal{F}|$ and $f = \log_2 F$.

SECURE SKETCHES: THE CODE-OFFSET CONSTRUCTION. For the case of $\mathcal{F} = \{0,1\}$, Juels and Wattenberg [JW99] considered a notion of "fuzzy commitment." [5] Given a $[n, k, 2t+1]_2$ error-correcting code C (not necessarily linear), they fuzzy-commit to x by publishing $w \oplus C(x)$. Their construction can be rephrased in our language to give a very simple construction of secure sketches for general $\mathcal{F}$.

We start with a $[n, k, 2t+1]_\mathcal{F}$ error-correcting code C (not necessarily linear). The idea is to use C to correct errors in w even though w may not be in C. This is accomplished by shifting the code so that a codeword matches up with w, and storing the shift as the sketch. To do so, we need to view $\mathcal{F}$ as an additive cyclic group of order $\mathcal{F}$ (in case of most common error-correcting codes, $\mathcal{F}$ will anyway be a field).

[5] In their interpretation, one commits to x by picking a random w and publishing $\mathsf{SS}(w; x)$.

Construction 2 (Code-Offset Construction). On input w, select a random codeword c (this is equivalent to choosing a random $x \in \mathcal{F}^k$ and computing $C(x)$), and set $\mathsf{SS}(w)$ to be the shift needed to get from c to w: $\mathsf{SS}(w) = w - c$. Then $\mathsf{Rec}(w', s)$ is computed by subtracting the shift s from w' to get $c' = w' - s$; decoding c' to get c (note that because $\mathsf{dis}(w', w) \leq t$, so is $\mathsf{dis}(c', c)$); and computing w by shifting back to get $w = c + s$.



In the case of $\mathcal{F} = \{0, 1\}$, addition and subtraction are the same, and we get that computation of the sketch is the same as the Juels-Wattenberg commitment: $\mathsf{SS}(w) = w \oplus C(x)$. In this case, to recover w given w' and $s = \mathsf{SS}(w)$, compute $c' = w' \oplus s$, decode c' to get c, and compute $w = c \oplus s$.

When the code C is linear, this scheme can be simplified as follows.

Construction 3 (Syndrome Construction). Set $\mathsf{SS}(w) = \mathsf{syn}(w)$. To compute $\mathsf{Rec}(w', s)$, find the unique vector $e \in \mathcal{F}^n$ of Hamming weight $\leq t$ such that $\mathsf{syn}(e) = \mathsf{syn}(w') - s$, and output $w = w' - e$.

As explained in Section 2, finding the short error-vector e from its syndrome this is the same as decoding the code. It is easy to see that two constructions above are equivalent: given $\mathsf{syn}(w)$ one can sample from $w - c$ by choosing a random string v with $\mathsf{syn}(v) = \mathsf{syn}(w)$; conversely, $\mathsf{syn}(w - c) = \mathsf{syn}(w)$. To show that Rec finds the correct w, observe that $\mathsf{dis}(w' - e, w') \leq t$ by the constraint on the weight of e, and $\mathsf{syn}(w' - e) = \mathsf{syn}(w') - \mathsf{syn}(e) = \mathsf{syn}(w') - (\mathsf{syn}(w') - s) = s$. There can be only one value within distance t of w' whose syndrome is s (else by subtracting two such values we get a codeword that is closer than $2t + 1$ to 0, but 0 is also a codeword), so $w' - e$ must be equal to w.

As mentioned in the introduction, the syndrome construction has appeared before as a component of some cryptographic protocols over quantum and other noisy channels [BBCS91, Cré97], though it has not been analyzed the same way.

Both schemes are $(\mathcal{F}^n, m, m - (n - k)f, t)$ secure sketches. For the randomized scheme, the intuition for understanding the entropy loss is as follows: we add k random elements of $\mathcal{F}$ and publish n elements of $\mathcal{F}$. The formal proof is simply Lemma 4.3, because addition in $\mathcal{F}^n$ is a family of transitive isometries. For the syndrome scheme, this follows from Lemma 3.1, because the syndrome is $(n - k)$ elements of $\mathcal{F}$.

We thus obtain the following theorem.

Theorem 5.1. *Given an $[n, k, 2t+1]_{\mathcal{F}}$ error-correcting code, one can construct an $(\mathcal{F}^n, m, m-(n-k)f, t)$ secure sketch, which is efficient if encoding and decoding are efficient. Furthermore, if the code is linear, then the sketch is deterministic and its output is $(n - k)$ symbols long.*

In Appendix A we present some generic lower bounds on secure sketches and fuzzy extractors. Recall that $A_F(n, d)$ denotes the maximum number K of codewords possible in a code of distance d over n-character words from an alphabet of size F. Then by Lemma A.1, we obtain that the entropy loss of a secure sketch for the Hamming metric is at least $nf - \log_2 A_F(n, 2t+1)$ when the input is uniform (that is, when $m = nf$), because $K(\mathcal{M}, t)$ from Lemma A.1 is in this case equal to $A_F(n, 2t+1)$ (since a code that corrects t Hamming errors must have minimum distance at least $2t + 1$). This means that if the underlying code is optimal (i.e., $K = A_F(n, 2t + 1)$), then code-offset construction above is optimal for the case of uniform inputs, because its entropy loss is $nf - \log_F K \log_2 F = nf - \log_2 K$. Of course, we do not know the exact value of $A_F(n, d)$, let alone efficiently decodable codes which meet the bound, for many settings of F, n and d. Nonetheless, the code-offset scheme gets as close to optimality as is possible from coding constraints. If better efficient codes are invented, then better (i.e., lower loss or higher error-tolerance) secure sketches will result.

FUZZY EXTRACTORS. As a warm-up, consider the case when, W is uniform ($m = n$) and look at the code-offset sketch construction: $v = w - C(x)$. For $\mathsf{Gen}(w)$, output $R = x$, $P = v$. For $\mathsf{Rep}(w', P)$, decode $w' - P$ to obtain $C(x)$ and apply C^{-1} to obtain x. The result, quite clearly, is an $(\mathcal{F}^n, nf, kf, t, 0)$ fuzzy extractor, since v is truly random and independent of x when w is random. In fact, this is exactly the usage proposed by Juels and Wattenberg [JW99] except they viewed the above fuzzy extractor as a way to use w to "fuzzy commit" to x, without revealing information about x.

Unfortunately, the above construction setting $R = x$ only works for uniform W, since otherwise v would leak information about x.

In general, we use the construction in Lemma 4.1 combined with Theorem 5.1 to obtain the following theorem.

Theorem 5.2. *Given any $[n, k, 2t+1]_{\mathcal{F}}$ code C and any m, ϵ, there exists an $(\mathcal{M}, m, \ell, t, \epsilon)$ fuzzy extractor, where $\ell = m + kf - nf - 2\log\left(\frac{1}{\epsilon}\right) + 2$. The generation* Gen *and recovery* Rep *are efficient if C has efficient encoding and decoding.*

6 Constructions for Set Difference

We now turn to inputs that are subsets of a universe $\mathcal{U}$; let $n = |\mathcal{U}|$. This corresponds to representing an object by a list of its features. Examples include "minutiae" (ridge meetings and endings) in a fingerprint, short strings which occur in a long document, or lists of favorite movies.

Recall that the distance between two sets w, w' is the size of their symmetric difference: $\mathsf{dis}(w, w') = |w \triangle w'|$. We will denote this metric space by $\mathsf{SDif}(\mathcal{U})$. A set w can be viewed as its *characteristic vector* in $\{0, 1\}^n$, with 1 at position $x \in \mathcal{U}$ if $x \in w$, and 0 otherwise. Such representation of sets makes set difference the same as the Hamming metric. However, we will mostly focus on settings where n is much larger than the size of the w, so that representing a set w by n bits is much less efficient than, say, writing down a list of elements in the w, which requires only $|w| \log n$ bits.

LARGE VERSUS SMALL UNIVERSES. More specifically, we will distinguish two broad categories of settings. Let s denote the size of the sets the are given as inputs to the secure sketch (or fuzzy extractor) algorithms. Most of this section studies situations where the universe size n is super-polynomial in the set size s. We call this the "large universe" setting. In contrast, the "small universe" setting refers to situations in which $n = poly(s)$. We want our various constructions to run in polynomial time and use polynomial storage space. In the large universe setting, the n-bit string representation of a set becomes too large to be usable—we will strive for solutions that are polynomial in s and $\log n$.

In fact, in many applications—for example, when the input is a list of book titles—it is possible that the actual universe is not only large, but also difficult to enumerate, making it difficult to even find the position in the characteristic vector corresponding to $x \in w$. In that case, it is natural to enlarge the universe to a well-understood class—for example, to include all possible strings of a certain length, whether or not they are actual book titles. This has the advantage that the position of x in the characteristic vector is simply x itself; however, because the universe is now even larger, the dependence of running time on n becomes even more important.

FIXED VERSUS FLEXIBLE SET SIZE. In some situations, all objects are represented by feature sets of exactly the same size s, while in others the sets may be of arbitrary size. In particular, the original set w and

	Entropy Loss	Storage	Time	Set size	Notes
Juels-Sudan [JS02]	$t\log n + \log\left(\binom{n}{r}/\binom{n-s}{r-s}\right) + 2$	$r \log n$	$poly(r\log(n))$	Fixed	r is a parameter $s \le r \le n$
Generic syndrome	$n - \log A(n, 2t+1)$	$n - \log A(n, 2t+1)$ (for linear codes)	$poly(n)$	Flexible	ent. loss $\approx t\log(n)$ when $t \ll n$
Permutation-based	$\log\binom{n}{s} - \log A(n, 2t+1, s)$	$O(n\log n)$	$poly(n)$	Fixed	ent. loss $\approx t\log n$ when $t \ll n$
Improved JS	$t\log n$	$t\log n$	$poly(s\log n)$	Fixed	
PinSketch	$t\log(n+1)$	$t\log(n+1)$	$poly(s\log n)$	Flexible	See Section 6.3 for running time

Table 1: Summary of Secure Sketches for Set Difference.

the corrupted set w' from which we would like to recover the original need not be of the same size. We refer to these two settings as "fixed" and "flexible" set size, respectively. When the set size is fixed, the distance $\mathsf{dis}(w, w')$ is always even: $\mathsf{dis}(w, w') = t$ if and only if w and w' agree on exactly $s - \frac{t}{2}$ points. We will denote the restriction of $\mathsf{SDif}(\mathcal{U})$ to s-element subsets by $\mathsf{SDif}_s(\mathcal{U})$.

SUMMARY. As a point of reference, we will see below that $\log\binom{n}{s} - \log A(n, 2t+1, s)$ is a lower bound on the entropy loss of any secure sketch for set difference (whether or not the set size is fixed). Recall that $A(n, 2t+1, s)$ represents the size of the largest code for Hamming space with minimum distance $2t+1$, in which every word has weight exactly s. In the large universe setting, where $t \ll n$, the lower bound is approximately $t\log n$. The relevant lower bounds are discussed at the end of Sections 6.1 and 6.2.

In the following sections we will present several schemes which meet this lower bound. The setting of small universes is discussed in Section 6.1. We discuss the code-offset construction (from Section 5), as well as a permutation-based scheme which is tailored to fixed set size. The latter scheme is optimal for this metric, but impractical.

In the remainder of the section, we discuss schemes for the large universe setting. In Section 6.2 we give an improved version of the scheme of Juels and Sudan [JS02]. Our version achieves optimal entropy loss and storage $t\log n$ for fixed set size (notice the entropy loss doesn't depend on the set size s, although the running time does). The new scheme provides an exponential improvement over the original parameters (which are analyzed in Appendix D). Finally, in Section 6.3 we describe how to adapt syndrome decoding algorithms for BCH codes to our application. The resulting scheme, called PinSketch, has optimal storage and entropy loss $t\log(n+1)$, handles flexible set sizes, and is probably the most practical of the schemes presented here. Another scheme achieving similar parameters (but less efficiently) can be adapted from information reconciliation literature [MTZ03]; see Section 9 for more details.

We do not discuss fuzzy extractors beyond mentioning here that each secure sketch presented in this section can be converted to a fuzzy extractor using Lemma 4.1. We have already seen an example of such conversion in Section 5.

Table 1 summarizes the constructions discussed in this section.

6.1 Small Universes

When the universe size is polynomial in s, there are a number of natural constructions. The most direct one, given previous work, is the construction of Juels and Sudan [JS02]. Unfortunately, that scheme requires a fixed set size and achieves relatively poor parameters (see Appendix D).

We suggest two possible constructions: first, to represent sets as n-bit strings and use the constructions of Section 5. The second construction, presented below, requires a fixed set size but achieves slightly improved parameters by going through "constant-weight" codes.

PERMUTATION-BASED SKETCH. Recall the general construction of Section 4.2 for transitive metric spaces. Let Π be a set of all permutations on $\mathcal{U}$. Given $\pi \in \Pi$, make it a permutation on $\mathsf{SDif}_s(\mathcal{U})$ naturally: $\pi(w) = \{\pi(x) | x \in w\}$. This makes Π is a family of transitive isometries on $\mathsf{SDif}_s(\mathcal{U})$, and thus the results of Section 4.2 apply.

Let $C \subseteq \{0,1\}^n$ be any $[n, k, 2t+1]$ binary code in which all words have weight exactly s. Such codes have been studied extensively (see, e.g., [AVZ00, BSSS90] for a summary of known upper and lower bounds). View elements of the code as sets of size s. We obtain the following scheme, which produces a sketch of length $O(n \log n)$.

Construction 4 (Permutation-Based Sketch). On input $w \subseteq \mathcal{U}$ of size s, choose $b \subseteq \mathcal{U}$ at random from the code C, and choose a random permutation $\pi : \mathcal{U} \to \mathcal{U}$ such that $\pi(w) = b$ (that is, choose a random matching between w and b and a random matching between $\mathcal{U} - w$ and $\mathcal{U} - b$). Output $\mathsf{SS}(w) = \pi$ (say, by listing $\pi(1), ..., \pi(n)$). To recover w from w' such that $\mathsf{dis}((, w), w') \leq t$ and π, compute $b' = \pi^{-1}(w')$, decode the characteristic vector of b' to obtain b, and output $w = \pi(b)$.

This construction is efficient as long as decoding is efficient (everything else takes time $O(n \log n)$). By Lemma 4.3, its entropy loss is $\log \binom{n}{s} - k$: here $|\Pi| = n!$ and $\Gamma = s!(n-s)!$, so $\log |\Pi| - \log \Gamma = \log n!/(s!(n-s)!)$.

COMPARING THE HAMMING SCHEME WITH THE PERMUTATION SCHEME. The code-offset construction was shown to have entropy loss $n - \log A(n, 2t+1)$ if an optimal code is used; the random permutation scheme has entropy loss $\log \binom{n}{s} - \log A(n, 2t+1, s)$ for an optimal code. The Bassalygo-Elias inequality (see [vL92]) shows that the bound on the random permutation scheme is always at least as good as the bound on the code offset scheme: $A(n,d) \cdot 2^{-n} \leq A(n,d,s) \cdot \binom{n}{s}^{-1}$. This implies that $n - \log A(n,d) \geq \log \binom{n}{s} - \log A(n,d,s)$. Moreover, standard packing arguments give better constructions of constant-weight codes than they do of ordinary codes. [6] In fact, the random permutations scheme is optimal for this metric, just as the code-offset scheme is optimal for the Hamming metric. We show this as follows. Restrict t to be even, because $\mathsf{dis}(w, w')$ is always even if $|w| = |w'|$. Then the minimum distance of a code over $\mathsf{SDif}_s(\mathcal{U})$ that corrects up to t errors must be at least $2t+1$.[7] Therefore by Lemma A.1, we get that that the min-entropy loss of a secure sketch must be at least $\log \binom{n}{s} - \log A(n, 2t+1, s)$, in the case of a uniform input w. Thus in principle, it is better to use the random permutation scheme. Nonetheless, there are caveats. First, we do not know of *explicitly* constructed constant-weight codes that beat the Elias-Bassalygo inequality and would thus lead to better entropy loss for the random permutation scheme than for the Hamming scheme (see [BSSS90] for more on constructions of constant-weight codes and [AVZ00] for upper bounds). Second, much more is known about efficient implementation of decoding for ordinary codes than for constant-weight codes; for example, one can find off-the-shelf hardware and software for decoding many binary codes. In practice, the Hamming-based scheme is likely to be more useful.

[6] This comes from the fact that the intersection of a ball of radius d with the set of all words of weight s is much smaller than the ball of radius d itself.

[7] Indeed, suppose not. Then take two codewords, c_1 and c_2 such that $\mathsf{dis}(c_1, c_2) \leq 2t$. There are k elements in c_1 that are not in c_2 (call their set $c_1 - c_2$), and k elements in c_2 that are not in c_1 (call their set $c_2 - c_1$), with $k \leq t$. Starting with c_1, remove $t/2$ elements of $c_1 - c_2$ and add $t/2$ elements of $c_2 - c_1$ to obtain a set w (note that here we are using that t is even; if $k < t/2$, then use k elements). Then $\mathsf{dis}(c_1, w) \leq t$ and $\mathsf{dis}(c_2, w) \leq t$, and so if the received word is w, the receiver cannot be certain whether the sent word was c_1 or c_2, and hence cannot correct t errors.

6.2 Improving the Construction of Juels and Sudan

We now turn to the large universe setting, where n is super-polynomial in the set size s, and we would like operations to be polynomial in s and $\log n$.

Juels and Sudan [JS02] proposed a secure sketch for the set difference metric with fixed set size (called a "fuzzy vault" in that paper). We present their original scheme here with an analysis of the entropy loss in Appendix D. In particular, our analysis shows that the original scheme has good entropy loss only when the storage space is very large.

We suggest an improved version of the Juels-Sudan scheme which is simpler and achieves much better parameters. The entropy loss and storage space of the new scheme are both $t \log n$, which is optimal. (The same parameters are also achieved by the BCH-based construction PinSketch in Section 6.3.) Our scheme has the advantage of being even simpler to analyze, and the computations are simpler. As with the original Juels-Sudan scheme, we assume $n = |\mathcal{U}|$ is a prime power and work over $\mathcal{F} = GF(n)$.

An intuition for the scheme is that the numbers $y_{s+1}, ..., y_r$ from the JS scheme need not be chosen at random. One can instead evaluate them as $y_i = p'(x_i)$ for some polynomial p'. One can then represent the entire list of pairs (x_i, y_i) implicitly, using only a few of the coefficients of p'. The new sketch is deterministic (this was not the case for our preliminary version in [DRS04]). Its implementation is available [HJR].

Construction 5 (Improved JS Secure Sketch for Sets of Size s).
To compute $\mathsf{SS}(w)$:

1. Let $p'()$ be the unique monic polynomial of degree exactly s such that $p'(x) = 0$ for all $x \in w$. (That is, let $p'(z) \stackrel{\text{def}}{=} \sum_{x \in w}(z - x)$.)
2. Output the coefficients of $p'()$ of degree $s - 1$ down to $s - t$.
   This is equivalent to computing and outputting the first t symmetric polynomials of the values in A, i.e. if $w = \{x_1, ..., x_s\}$, then output

$$\sum_i x_i,\ \sum_{i \neq j} x_i x_j,\ \ldots,\ \sum_{S \subseteq [s], |S| = t} \left(\prod_{i \in S} x_i \right).$$

To compute $\mathsf{Rec}(w', p')$, where $w' = \{a_1, a_2, \ldots, a_s\}$,

1. Create a new polynomial p_{high}, of degree s which shares the top $t + 1$ coefficients of p', that is let $p_{\text{high}}(z) \stackrel{\text{def}}{=} z^s + \sum_{i=s-t}^{s-1} a_i z^i$.
2. Evaluate p_{high} on all points in w' to obtain s pairs (a_i, b_i).
3. Use $[s, s - t, t + 1]_{\mathcal{U}}$ Reed-Solomon decoding (see, e.g., [Bla83, vL92]) to search for a polynomial p_{low} of degree $s - t - 1$ such that $p_{\text{low}}(a_i) = b_i$ for at least $s - t/2$ of the a_i values. If no such polynomial exists, then stop and output "fail."
4. Output the list of zeroes (roots) of the polynomial $p_{\text{high}} - p_{\text{low}}$ (see, e.g., [Sho05] for root-finding algorithms; they can sped up by first factoring out the known roots—namely, $(z - a_i)$ for the $s - t/2$ values of a_i that were not deemed erroneous in the previous step).

To see that this secure sketch can tolerate t set difference errors, suppose $\mathsf{dis}(w, w') \leq t$. Let p' be as in the sketch algorithm, that is $p'(z) = \prod_{x \in w}(z - x)$. The polynomial p' is monic, that is its leading term is

z^s. We can divide the remaining coefficients into two groups: the high coefficients, denoted $a_{s-t}, ..., a_{s-1}$, and the low coefficients, denoted by $b_1, ..., b_{s-t-1}$:

$$p'(z) = \underbrace{z^s + \sum_{i=s-t}^{s-1} a_i z^i}_{p_{\text{high}}(z)} + \underbrace{\sum_{i=0}^{s-t-1} b_i z^i}_{q(z)} .$$

We can write p' as $p_{\text{high}} + q$ where q has degree $s - t - 1$. The recovery algorithm gets the coefficients of p_{high} as input. For any point x in w, we have $0 = p'(x) = p_{\text{high}}(x) + q(x)$. Thus, p_{high} and $-q$ agree at all points in w. Since the set w intersects w' in at least $s - t/2$ points, the polynomial $-q$ satisfies the conditions of Step 3 in Rec. That polynomial is unique, since no two distinct polynomials of degree $s-t-1$ can get the correct b_i on more than $s - t/2$ a_is (else, they agree on at least $s - t$ points, which is impossible). Therefore, the recovered polynomial p_{low} must be $-q$; hence $p_{\text{high}}(x) - p_{\text{low}}(x) = p'(x)$. Thus, Rec computes the correct p' and therefore finds correctly the set w, which consists of the roots of p'.

Since the output of SS is t field elements, the entropy loss of the scheme is at most $t \log n$ by Lemma 3.1. (We will see below that this bound is tight, since any sketch must lose at least $t \log n$ in some situations.) We have proved:

Theorem 6.1 (Analysis of Improved JS). *Construction 5 is a* $(\mathsf{SDif}_s(\mathcal{U}), m, m - t \log n, t)$ *secure sketch. The entropy loss and storage of the scheme are at most* $t \log n$*, and both the sketch generation* $\mathsf{SS}()$ *and the recovery procedure* $\mathsf{Rec}()$ *run in time polynomial in* s*,* t *and* $\log n$*.*

LOWER BOUNDS FOR FIXED SET SIZE IN A LARGE UNIVERSE. The short length of the sketch makes this scheme feasible for essentially any ratio of set size to universe size (we only need $\log n$ to be polynomial in s). Moreover, for large universes the entropy loss $t \log n$ is essentially optimal for uniform inputs (i.e., when $m = \log \binom{n}{s}$). We show this as follows. As already mentioned in the Section 6.1, Lemma A.1 shows that for a uniformly distributed input, the best possible entropy loss is $m - m' \geq \log \binom{n}{s} - \log A(n, 2t + 1, s)$. Using a bound of Agrell *et al.* [AVZ00], Theorem 12 (and noting that $A(n, 2t + 1, s) = A(n, 2t + 2, s)$, because distances in $\mathsf{SDif}_s(\mathcal{U})$ are even), the entropy loss is at least:

$$m - m' \geq \log \binom{n}{s} - \log A(n, 2t + 1, s) \geq \log \binom{n}{s} - \log \left(\binom{n}{s-t} \Big/ \binom{s}{s-t} \right) = \log \binom{n-s+t}{t}$$

When $n \gg s$, this last quantity is roughly $t \log n$, as desired.

6.3 Large Universes via the Hamming Metric: Sublinear-Time Decoding

In this section, we show that the syndrome construction of Section 5 can in fact be adapted for small sets in large universe, using specific properties of algebraic codes. We will show that BCH codes, which contain Hamming and Reed-Solomon codes as special cases, have these properties. As opposed to the constructions of the previous section, the construction of this section is flexible and can accept input sets of any size.

Thus we obtain a sketch for sets of flexible size, with entropy loss and storage $t \log(n + 1)$. We will assume that n is one less than a power of 2: $n = 2^m - 1$ for some integer m, and will identify $\mathcal{U}$ with the nonzero elements of the binary finite field of degree m: $\mathcal{U} = GF(2^m)^*$.

SYNDROME MANIPULATION FOR SMALL-WEIGHT WORDS. Suppose now that we have a small set $w \subseteq \mathcal{U}$ of size s, where $n \gg s$. Let x_w denote the characteristic vector of w (see the beginning of

Section 6). Then the syndrome construction says that $\mathsf{SS}(w) = \mathsf{syn}(x_w)$. This is an $(n-k)$-bit quantity. Note that the syndrome construction gives us no special advantage over the code-offset construction when the universe is small: storing the n-bit $x_w + C(r)$ for a random k-bit r is not a problem. However, it's a substantial improvement when $n \gg n-k$.

If we want to use $\mathsf{syn}(x_w)$ as the sketch of w, then we must choose a code with $n-k$ very small. In particular, the entropy of w is at most $\log \binom{n}{s} \approx s \log n$, and so the entropy loss $n-k$ had better be at most $s \log n$. Binary BCH codes are suitable for our purposes: they are a family of $[n,k,\delta]_2$ linear codes with $\delta = 2t+1$ and $k = n - tm$ (assuming $n = 2^m - 1$) (see, e.g. [vL92]). These codes are optimal for $t \ll n$ by the Hamming bound, which implies that $k \le n - \log \binom{n}{t}$ [vL92].[8] Using the syndrome sketch with a BCH code C, we get entropy loss $n - k = t \log(n+1)$, essentially the same as the $t \log n$ of the improved Juels-Sudan scheme (recall that $\delta \ge 2t+1$ allows us to correct t set difference errors).

The only problem is that the scheme appears to require computation time $\Omega(n)$, since we must compute $\mathsf{syn}(x_w) = Hx_w$ and, later, run a decoding algorithm to recover x_w. For BCH codes, this difficulty can be overcome. A word of small weight w can be described by listing the positions on which it is nonzero. We call this description the *support* of x_w and write $\mathsf{supp}(x_w)$ (note that $\mathsf{supp}(x_w) = w$; see the discussion of enlarging the universe appropriately at the beginning of Section 6).

The following lemma holds for general BCH codes (which include binary BCH codes and Reed-Solomon codes as special cases). We state it for binary codes since that is most relevant to the application:

Lemma 6.2. *For a $[n,k,\delta]$ binary BCH code C one can compute:*

- $\mathsf{syn}(x)$*, given* $\mathsf{supp}(x)$*, in time polynomial in δ, $\log n$, and* $|\mathsf{supp}(x)|$
- $\mathsf{supp}(x)$*, given* $\mathsf{syn}(x)$ *(when x has weight at most $(\delta-1)/2$), in time polynomial in δ and $\log n$.*

The proof of Lemma 6.2 requires a careful reworking of the standard BCH decoding algorithm. The details are presented in Appendix E. For now, we present the resulting secure sketch for set difference.

Construction 6 (PinSketch).
To compute $\mathsf{SS}(w) = \mathsf{syn}(x_w)$:

1. Let $s_i = \sum_{x \in w} x^i$ (computations in $GF(2^m)$).
2. Output $\mathsf{SS}(w) = (s_1, s_3, s_5, ..., s_{2t-1})$.

To recover $\mathsf{Rec}(w', (s_1, s_3, \ldots, s_{2t-1})$:

1. Compute $(s'_1, s'_3, \ldots, s'_{2t-1}) = \mathsf{SS}(w') = \mathsf{syn}(x_{w'})$;
2. Let $\sigma_i = s'_i - s_i$ (in $GF(2^m)$, so "$-$" is the same as "$+$").
3. Compute $supp(v)$ such that $syn(v) = (\sigma_1, \sigma_3, \ldots, \sigma_{2t-1})$ and $|supp(v)| \le t$ by Lemma 6.2.
4. If $\mathsf{dis}(w, w') \le t$, then $\mathsf{supp}(v) = w \triangle w'$. Thus, output $w = w' \triangle \mathsf{supp}(v)$.

An implementation of this construction, including the reworked BCH decoding algorithm, is available [HJR].

The bound on entropy loss is easy to see: the output is $t \log(n+1)$ bits long, and hence the entropy loss is at most $t \log(n+1)$ by Lemma 3.1. We obtain:

Theorem 6.3. *PinSketch is a $(\mathsf{SDif}(\mathcal{U}), m, m - t\log(n+1), t)$ secure sketch for set difference with storage $t \log(n+1)$. The algorithms* SS *and* Rec *both run in time polynomial in t and $\log n$.*

[8] The Hamming bound is based on the observation that for any code of distance δ, the balls of radius $\lfloor(\delta-1)/2\rfloor$ centered at various codewords must be disjoint. Each such ball contains $\binom{n}{\lfloor(\delta-1)/2\rfloor}$ points, and so $2^k\binom{n}{\lfloor(\delta-1)/2\rfloor} \le 2^n$. In our case $\delta = 2t+1$ and so the bound yields $k \le n - \log\binom{n}{t}$.

7 Constructions for Edit Distance

The space of interest in this section is the space $\mathcal{F}^*$ for some alphabet $\mathcal{F}$, with distance between two strings defined as the number of character insertions and deletions needed to get from one string to the other. Denote this space by $\mathsf{Edit}_{\mathcal{F}}(n)$. Let $F = |\mathcal{F}|$.

First, note that applying the generic approach for transitive metric spaces (as with the Hamming space and the set difference space for small universe sizes) does not work here, because the edit metric is not known to be transitive. Instead, we consider embeddings of the edit metric on $\{0,1\}^n$ into the Hamming or set-difference metric of much larger dimension. We look at two types: standard low-distortion embeddings, and "biometric" embeddings as defined in Section 4.3.

For the binary edit distance space of dimension n, we obtain secure sketches and fuzzy extractors correcting t errors with entropy loss roughly $tn^{o(1)}$, using a standard embedding, and $2.38\sqrt[3]{tn\log n}$, using a relaxed embedding. The first technique works better when t is small, say $n^{1-\gamma}$ for a constant $\gamma > 0$. The second technique is better when t is large; it is meaningful roughly as long as $t < \frac{n}{15\log^2 n}$.

7.1 Low-Distortion Embeddings

A (standard) embedding with distortion D is an injection $\psi : \mathcal{M}_1 \hookrightarrow \mathcal{M}_2$ such that for any two points $x, y \in \mathcal{M}_1$, the ratio $\frac{\mathsf{dis}(\psi(x),\psi(y))}{\mathsf{dis}(x,y)}$ is at least 1 and at most D.

When the preliminary version of this paper appeared [DRS04], no non-trivial embeddings were known mapping edit distance into ℓ_1 or the Hamming metric (i.e. known embeddings had distortion $O(n)$). Recently, Ostrovsky and Rabani [OR05] gave a embedding of the edit metric over $\mathcal{F} = \{0,1\}$ into ℓ_1 with subpolynomial distortion. The embedding can in fact be interpreted as mapping to the Hamming space $\{0,1\}^d$ where $d = \mathrm{poly}(n)$. [9]

Fact 7.1 ([OR05]). *There is a polynomial-time computable embedding* $\psi_{\mathrm{ed}} : \mathsf{Edit}_{\{0,1\}}(n) \hookrightarrow \{0,1\}^{\mathrm{poly}(n)}$ *with distortion* $D_{\mathrm{ed}}(n) \stackrel{\mathrm{def}}{=} 2^{O(\sqrt{\log n \log\log n})}$.

We can compose this embedding with constructions for the Hamming distance to obtain a secure sketch (and hence fuzzy extractor) for edit distance which will be good when t, the number of errors to be corrected, is quite small. Recall that instantiating the syndrome construction (Construction 3) with a BCH code allows one to correct t' errors out of d at the cost of $t' \log d$ bits of entropy.

Construction 7. For any length n and error threshold t, let ψ_{ed} be the embedding of Fact 7.1 from $\mathsf{Edit}_{\{0,1\}}(n)$ into $\{0,1\}^d$ (where $d = \mathrm{poly}(n)$) and let syn be the syndrome of a BCH code correcting $t' = tD_{\mathrm{ed}}(n)$ errors in $\{0,1\}^d$. On input $w \in \mathsf{Edit}_{\{0,1\}}(n)$, output

$$\mathsf{SS}(w) = \mathsf{syn}(\psi_{\mathrm{ed}}(w)).$$

To recover w from w' and $s = \mathsf{SS}(w)$, compute $\psi_{ed}^{-1}(\mathsf{Rec}(\psi_{\mathrm{ed}}(w', s)))$, where Rec is from Construction 3.

To extend this construction to general $\mathcal{F}$, represent each character of $\mathcal{F}$ as a string $\log F$ bits. This is an embedding $\mathcal{F}^n$ into $\{0,1\}^{n\log F}$, which increases edit distance by a factor of at most $\log F$. Then $t' = t(\log F)D_{\mathrm{ed}}(n)$ and $d = \mathrm{poly}(n, \log F)$.

[9] The embedding of [OR05] produces strings of integers in the space $\{1, ..., O(\log n)\}^{\mathrm{poly}(n)}$, equipped with ℓ_1 distance. One can convert this into the Hamming metric with only a logarithmic blowup in length by representing each integer in unary.

Proposition 7.2. *For any n, t, m, there is a (m, m', t)-secure sketch for* $\mathsf{Edit}_{\{0.1\}}(n)$ *where* $m' = m - t2^{O(\sqrt{\log n \log\log n})}$. *In particular, for any $\alpha < 1$, there exists a secure sketch correcting n^α errors with entropy loss $n^{\alpha+o(1)}$.*

Proof. The sketch of Construction 7 has length $t' \log d = t2^{O(\sqrt{\log n \log\log n})} \log \text{poly}(n)$. Because $2^{O(\sqrt{\log n \log\log n})}$ grows faster than $\log n$, this is the same as $t2^{O(\sqrt{\log n \log\log n})}$. And the sketch length is an upper bound on the entropy loss by Lemma 3.1. □

The generalization of this lemma for larger alphabets (again, by the same embedding) requires changing the formula for m' as $m' = m - t(\log F)2^{O(\sqrt{\log(n \log F) \log\log(n \log F)})}$.

Note that the peculiar-looking distortion function from Fact 7.1 increases more slowly than any polynomial in n, but still faster than any polynomial in $\log n$. In sharp contrast, the best lower bound states that any embedding of Edit_n in ℓ_1 (and hence Hamming) must have distortion at least $3/2$ (Andoni, Deza, Gupta, Indyk and Raskhodnikova, [ADG+03]). Closing the gap between the two bounds remains an open problem.

7.2 Relaxed Embeddings for the Edit Metric

In this section, we show that a relaxed notion of embedding, called a *biometric embedding* in Section 4.3, can produce fuzzy extractors and secure sketches that are better than what one can get from the embedding of [OR05] when t is large (they are also much simpler algorithmically, which makes them more practical). We first discuss fuzzy extractors and later extend the technique to secure sketches.

FUZZY EXTRACTORS. Recall that unlike low-distortion embeddings, biometric embeddings do not care about relative distances, as long as points that were "close" (closer than t_1) do not become "distant" (farther apart than t_2). The only additional requirement of a biometric embedding is that it preserve some min-entropy: we do not want too many points to collide together. We now describe such an embedding from the edit distance to the set difference.

A c-*shingle* (Broder, [Bro97]) is a length-c consecutive substring of a given string w. A c-*shingling* [Bro97] of a string w of length n is the set (ignoring order or repetition) of all $(n - c + 1)$ c-shingles of w. (For instance, a 3-shingling of "abcdecdegh" is {abc, bcd, cde, dec, ecd, dea, eah}).Thus, the range of the c-shingling operation consists of all nonempty subsets of size at most $n - c + 1$ of $\mathcal{F}^c$. Let $\mathsf{SDif}(\mathcal{F}^c)$ stand for the set difference metric over subsets of $\mathcal{F}^c$ and SH_c stand for the c-shingling map from $\mathsf{Edit}_{\mathcal{F}}(n)$ to $\mathsf{SDif}(\mathcal{F}^c)$. We now show that SH_c is a good biometric embedding.

Lemma 7.3. *For any c, SH_c is a $(t_1, t_2 = (2c-1)t_1, m_1, m_2 = m_1 - \lceil \frac{n}{c} \rceil \log_2(n - c + 1))$-biometric embedding of $\mathsf{Edit}(n)$ into $\mathsf{SDif}(\mathcal{F}^c)$.*

Proof. Let $w, w' \in \mathsf{Edit}(n)$ be such that $\mathsf{dis}(w, w') \le t_1$ and I be the sequence of at most t_1 insertions and deletions that transforms w into w'. It is easy to see that each character deletion or insertion adds at most $(2c - 1)$ to the symmetric difference between $\mathsf{SH}_c(w)$ and $\mathsf{SH}_c(w')$, which implies that $\mathsf{dis}(\mathsf{SH}_c(w), \mathsf{SH}_c(w')) \le (2c-1)t_1$, as needed.

For $w \in \mathcal{F}^n$, define $g_c(w)$ as follows. Compute $\mathsf{SH}_c(w)$ and store the resulting shingles in lexicographic order $h_1 \ldots h_k$ ($k \le n - c + 1$). Next, naturally partition w into $\lceil n/c \rceil$ c-shingles $s_1 \ldots s_{\lceil n/c \rceil}$, all disjoint except for (possibly) the last two, which overlap by $c\lceil n/c \rceil - n$ characters. Next, for $1 \le j \le \lceil n/c \rceil$, set p_j to be the index $i \in \{0 \ldots k\}$ such that $s_j = h_i$. In other words, p_j tells the index of the j-th disjoint shingle of w in the alphabetically-ordered k-set $\mathsf{SH}_c(w)$. Set $g_c(w) = (p_1, \ldots, p_{\lceil n/c \rceil})$. (For instance, g_3("abcdecdeah") $= (1, 5, 4, 6)$, representing the alphabetical order of "abc", "dec", "dea" and "eah" in

SH_3("abcdecdeah").) The number of possible values for $g_c(w)$ is at most $(n-c+1)^{\lceil \frac{n}{c} \rceil}$, and w can be completely recovered from $\mathsf{SH}_c(w)$ and $g_c(w)$.

Now, assume W is any distribution of min-entropy at least m_1 on $\mathsf{Edit}(n)$. Applying Lemma 3.1, we get $\tilde{\mathbf{H}}_\infty(W \mid g_c(W)) \geq m_1 - \lceil \frac{n}{c} \rceil \log_2(n-c+1)$. Since $\Pr(W = w \mid g_c(W) = g) = \Pr(\mathsf{SH}_c(W) = \mathsf{SH}_c(w) \mid g_c(W) = g)$ (because given $g_c(w)$, $\mathsf{SH}_c(w)$ uniquely determines w and vice versa), by applying the definition of $\tilde{\mathbf{H}}_\infty$, we obtain $\mathbf{H}_\infty(\mathsf{SH}_c(W)) \geq \tilde{\mathbf{H}}_\infty(\mathsf{SH}_c(W) \mid g_c(W)) = \tilde{\mathbf{H}}_\infty(W \mid g_c(W))$. □

By Theorem 6.3, for universe $\mathcal{F}^c$ of size F^c and distance threshold $t_2 = (2c-1)t_1$, we can construct a secure sketch for the set difference metric with entropy loss $t_2\lceil \log(F^c+1) \rceil$ ($\lceil \cdot \rceil$ because Theorem 6.3 requires the universe size to be one less than a power of 2). By Lemma 4.1, we can obtain a fuzzy extractor from such a sketch, with additional entropy loss $2\log\left(\frac{1}{\epsilon}\right) - 2$. Applying Lemma 4.4 to the above embedding and this fuzzy extractor, we obtain a fuzzy extractor for $\mathsf{Edit}(n)$, any input entropy m, any distance t, and any security parameter ϵ, with the following entropy loss:

$$\left\lceil \frac{n}{c} \right\rceil \cdot \log_2(n-c+1) + (2c-1)t\lceil \log(F^c+1) \rceil + 2\log\left(\frac{1}{\epsilon}\right) - 2$$

(the first component of the entropy loss comes from the embedding, the second from the secure sketch for set difference, and the third from the extractor). The above sequence of lemmas results in the following construction, parameterized by shingle length c and a family of pairwise independent hash function $\mathcal{H} = \{\mathsf{SDif}(\mathcal{F}^c) \to \{0,1\}^l\}_{x \in X}$, where l is equal to the input entropy m minus the entropy loss above.

Construction 8 (Fuzzy Extractor for Edit Distance).
To compute $\mathsf{Gen}(w)$ for $|w| = n$:

1. Compute $\mathsf{SH}_c(w)$ by computing $n-c+1$ shingles $(v_1, v_2, \ldots, v_{n-c+1})$ and removing duplicates to form the shingle set v from w.
2. Compute $s = \mathsf{syn}(x_v)$ as in Construction 6.
3. Select a hash function $H_x \in \mathcal{H}$ and output $(R = H_x(v)), P = (s, x))$.

To compute $\mathsf{Rep}(w', (s, x))$:

1. Compute $\mathsf{SH}_c(w')$ as above to get v'.
2. Use $\mathsf{Rec}(v', s)$ from in Construction 6 to recover v.
3. Output $R = H_x(v)$.

We thus obtain the following theorem.

Theorem 7.4. *For any n, m, c and $0 < \epsilon \leq 1$, there is an efficient $(\mathsf{Edit}(n), m, m - \lceil \frac{n}{c} \rceil \log_2(n-c+1) - (2c-1)t\lceil \log(F^c+1) \rceil - 2\log\left(\frac{1}{\epsilon}\right) + 2, t, \epsilon)$-fuzzy extractor.*

Note that the choice of c is a parameter; by ignoring $\lceil \cdot \rceil$ and replacing $n-c+1$ with n, $2c-1$ with $2c$ and F^c+1 with F^c, we get that the minimum entropy loss occurs near

$$c = \left(\frac{n \log n}{4t \log F}\right)^{1/3}$$

and is about $2.38\,(t \log F)^{1/3}\,(n \log n)^{2/3}$ (2.38 is really $\sqrt[3]{4} + 1/\sqrt[3]{2}$). In particular, if the original string has a linear amount of entropy $\theta(n \log F)$, then we can tolerate $t = \Omega(n \log^2 F / \log^2 n)$ insertions and deletions while extracting $\theta(n \log F) - 2\log\left(\frac{1}{\epsilon}\right)$ bits. The number of bits extracted is linear; if the string length n is polynomial in the alphabet size F, then the number of errors tolerated is linear also.

SECURE SKETCHES. Observe that the proof of Lemma 7.3 actually demonstrates that our biometric embedding based on shingling is an embedding with recovery information g_c. Observe also that it is easy to reconstruct w from $\mathsf{SH}_c(w)$ and $g_c(w)$. Finally, note that PinSketch (Construction 6) is an average-case secure sketch (as are all secure sketches in this work). Thus, combining Theorem 6.3 with Lemma 4.5 we obtain the following theorem.

Construction 9 (Secure Sketch for Edit Distance). For $\mathsf{SS}(w)$, compute $v = \mathsf{SH}_c(w)$ and $s_1 = \mathsf{syn}(x_v)$ as in Construction 8. Compute $s_2 = g_c(w)$, writing each p_j as a string of $\lceil \log n \rceil$ bits. Output $s = (s_1, s_2)$. For $\mathsf{Rec}(w', (s_1, s_2))$, recover v as in Construction 8, sort it in alphabetical order, and recover w by stringing along elements of v according to indices in s_2.

Theorem 7.5. *For any n, m, c and $0 < \epsilon \leq 1$, there is an efficient $(\mathsf{Edit}(n), m, m - \lceil \frac{n}{c} \rceil \log_2(n - c + 1) - (2c-1)t\lceil \log(F^c + 1) \rceil, t)$ average-case secure sketch.*

The discussion about optimal values of c from above applies equally here.

Remark 3. In our definitions of secure sketches and fuzzy extractors, we required the original w and the (potentially) modified w' to come from the same space $\mathcal{M}$. This requirement was for simplicity of exposition. We can allow w' to come from a larger set, as long as distance from w is well-defined. In the case of edit distance, for instance, w' can be shorter or longer than w; all the above results will apply as long as it is still within t insertions and deletions.

8 Probabilistic Notions of Correctness

The error model considered so far in this work is very strong: we required that secure sketches and fuzzy extractors accept *every* secret w' within distance t of the original input w, with no probability of error.

Such a stringent model is useful as it makes no assumptions on either the exact stochastic properties of the error process or the adversary's computational limits. However, Lemma A.1 shows that secure sketches (and fuzzy extractors) correcting t errors can only be as "good" as error-correcting codes with minimum distance $2t + 1$. By slightly relaxing the correctness condition, we will see that one can tolerate many more errors. For example, there is no good code which can correct $n/4$ errors in the binary Hamming metric: by the Plotkin bound (see, e.g., [Sud01, Lecture 8]) a code with minimum distance greater than $n/2$ has at most $2n$ codewords. Thus, there is no secure sketch with residual entropy $m' \geq \log n$ which can correct $n/4$ errors with probability 1. However, with the relaxed notions of correctness below, one can tolerate arbitrarily close to $n/2$ errors, i.e., correct $n(\frac{1}{2} - \gamma)$ errors for any constant $\gamma > 0$, and still have residual entropy $\Omega(n)$.

In this section, we discuss three relaxed error models and show how the constructions of the previous sections can be modified to gain greater error-correction in these models. We will focus on secure sketches for the binary Hamming metric. The same constructions yield fuzzy extractors (by Lemma 4.1). Many of the observations here also apply to metrics other than Hamming.

A common point is that we will only require that the a corrupted input w' be recovered with probability at least $1-\alpha < 1$ (the probability space varies). We describe each model in terms of the additional assumptions made on the error process. We describe constructions for each model in the subsequent sections.

Random Errors Assume there is a *known* distribution on the errors which occur in the data. For the Hamming metric, the most common distribution is the binary symmetric channel BSC_p: each bit of the input is flipped with probability p and left untouched with probability $1 - p$. We require that for any input w, $\mathsf{Rec}(W', \mathsf{SS}(w)) = w$ with probability at least $1 - \alpha$ over the coins of SS and over W' drawn applying the noise distribution to w.

In that case, one can correct an error rate up to Shannon's bound on noisy channel coding. This bound is tight. Unfortunately, the assumption of a known noise process is too strong for most applications: there is no reason to believe we understand the exact distribution on errors which occur in complex data such as biometrics.[10] However, it provides a useful baseline by which to measure results for other models.

Input-dependent Errors The errors are adversarial, subject only to the conditions that (a) the error $\mathsf{dis}(w, w')$ is bounded to a maximum magnitude of t, and (b) the corrupted word *depends only on the input* w, and not on the secure sketch $\mathsf{SS}(w)$. Here we require that for any pair w, w' at distance at most t, we have $\mathsf{Rec}(w', \mathsf{SS}(w)) = w$ with probability at least $1 - \alpha$ over the coins of SS.

This model encompasses any complex noise process which has been observed to never introduce more than t errors. Unlike the assumption of a particular distribution on the noise, the bound on magnitude can be checked experimentally. Perhaps surprisingly, in this model we can tolerate just as large an error rate as in the model of random errors. That is, we can tolerate an error rate up to Shannon's coding bound and no more.

Computationally-bounded Errors The errors are adversarial and may depend on both w and the publicly stored information $\mathsf{SS}(w)$. However, we assume that the errors are introduced by a process of bounded computational power. That is, there is a probabilistic circuit of polynomial size (in the length n) which computes w' from w. The adversary cannot, for example, forge a digital signature and base the error pattern on the signature.

It is not clear whether this model allows correcting errors up to the Shannon bound, as in the two models above. The question is related to open questions on the construction of efficiently list-decodable codes. However, when the error rate is either very high or very low, then the appropriate list-decodable codes exist and we can indeed match the Shannon bound.

ANALOGUES FOR NOISY CHANNELS AND THE HAMMING METRIC. Models analogous to the ones above have been studied in the literature on codes for noisy binary channels (with the Hamming metric). Random errors and computationally-bounded errors both make obvious sense in the coding context [Sha48, MPSW05]. The second model — input-dependent errors — does not immediately make sense in a coding situation, since there is no data other than the transmitted codeword on which errors could depend. Nonetheless, there is a natural, analogous model for noisy channels: one can allow the sender and receiver to share either (1) common, secret random coins (see [DGL04, Lan04] and references therein) or (2) a side channel with which they can communicate a small number of noise-free, secret bits [Gur03].

Existing results on these three models for the Hamming metric can be transported to our context using the code-offset construction:

$$\mathsf{SS}(w; x) = w \oplus C(x) .$$

[10] Since the assumption here only plays a role in correctness, it is still more reasonable than assuming we know exact distributions on the data in proofs of *secrecy*. However, in both cases, we would like to enlarge the class of distributions for which we can provably satisfy the definition of security.

Roughly, any code which corrects errors in the models above will lead to a secure sketch (resp. fuzzy extractor) which corrects errors in the model. We explore the consequences for each of the three models in the next sections.

8.1 Random Errors

The random error model was famously considered by Shannon [Sha48]. He showed that for any discrete, memoryless channel, the rate at which information can be reliably transmitted is characterized by the maximum mutual information between the inputs and outputs of the channel. For the binary symmetric channel with crossover probability p, this means that there exist codes encoding k bits into n bits, tolerating error probability p in each bit if and only if

$$\frac{k}{n} < 1 - h(p) - \delta(n)$$

where $h(p) = -p \log p - (1-p)\log(1-p)$ and $\delta(n) = o(1)$. Computationally efficient codes achieving this bound were found later, most notably by Forney [For66]. We can use the code-offset construction $\mathsf{SS}(w; x) = w \oplus C(x)$ with an appropriate concatenated code [For66] or, equivalently, $\mathsf{SS}(w) = \mathsf{syn}_C(w)$ since the codes can be linear. We obtain:

Proposition 8.1. *For any error rate $0 < p < 1/2$ and constant $\delta > 0$, for large enough n there exist secure sketches with entropy loss $(h(p) + \delta)n$, which correct error rate of p in the data with high probability (roughly $2^{-c_\delta n}$ for a constant $c_\delta > 0$).*

The probability here is taken over the errors *only (the distribution on input strings w can be arbitrary).*

The quantity $h(p)$ is less than 1 for any p in the range $(0, 1/2)$. In particular, one can get non-trivial secure sketches even for a very high error rate p as long as it is less that $1/2$; in contrast, no secure sketch which corrects errors with probability 1 can tolerate $t \geq n/4$. Note that several other works on biometric cryptosystems consider the model of randomized errors and obtain similar results, though the analyses assume that the distribution on inputs is uniform [TG04, CZ04].

A MATCHING IMPOSSIBILITY RESULT. The bound above is tight. The matching impossibility result also applies to input-dependent and computationally-bounded errors, since random errors are a special case of both more complex models.

We start with an intuitive argument: If a secure sketch allows recovering from random errors with high probability, than it must contain enough information about w to describe the error pattern (since given w' and $\mathsf{SS}(w)$, one can recover the error pattern with high probability). Describing the outcome of n independent coin flips with probability p of heads requires $nh(p)$ bits, and so the sketch must reveal $nh(p)$ bits about w.

In fact, that argument simply shows that $nh(p)$ bits of Shannon information are leaked about w, whereas we are concerned with min-entropy loss as defined in Section 3. To make the argument more formal, let W be uniform over $\{0,1\}^n$ and observe that with high probability over the output of the sketching algorithm, $v = \mathsf{SS}(w)$, the conditional distribution $W_v = W|_{\mathsf{SS}(W)=v}$ forms a good code for the binary symmetric channel. That is, for most values v, if we sample a random string w from $W|_{\mathsf{SS}(W)=v}$ and send it through a binary symmetric channel, we will be able to recover the correct value w. That means there exists some v such that both (a) W_v is a good code and (b) $\mathbf{H}_\infty(W_v)$ is close to $\tilde{\mathbf{H}}_\infty(W|\mathsf{SS}(W))$. Shannon's noisy coding theorem says that such a code can have entropy at most $n(1 - h(p) + o(1))$. Thus the construction above is optimal:

Proposition 8.2. *For any error rate $0 < p < 1/2$, any secure sketch* SS *which corrects random errors (with rate p) with probability at least $2/3$ has entropy loss at least $n(h(p) - o(1))$; that is $\tilde{\mathbf{H}}_\infty(W|\mathsf{SS}(W)) \leq n(1 - h(p) - o(1))$ when W is drawn uniformly from $\{0,1\}^n$.*

8.2 Randomizing Input-dependent Errors

Assuming errors distributed randomly according to a known distribution seems very limiting. In the Hamming metric, one can construct a secure sketch which achieves the same result as with random errors for every error process where the magnitude of the error is bounded, as long as the errors are independent of the output of $\mathsf{SS}(W)$. The same technique was used previously by Bennett et al. [BBR88, p. 216] and, in a slightly different context, Lipton [Lip94, DGL04].

The idea is to choose a random permutation $\pi : [n] \rightarrow [n]$, permute the bits of w before applying the sketch, and store the permutation π along with $\mathsf{SS}(\pi(w))$. Specifically, let C be a linear code tolerating a p fraction of random errors with redundancy $n - k \approx nh(p)$. Let

$$\mathsf{SS}(w;\pi) = \pi,\ \mathsf{syn}_C(\pi(w))$$

where $\pi : [n] \rightarrow [n]$ and, for $w = w_1 \cdots w_n \in \{0,1\}^n$, $\pi(w)$ denotes the permuted string $w_{\pi(1)} w_{\pi(2)} \cdots w_{\pi(n)}$. The recovery algorithm operates in the obvious way: it first permutes the input w' according to π^{-1}, then runs the usual syndrome recovery algorithm to recover $\pi(w)$.

For any particular pair w, w', the difference $w \oplus w'$ will be mapped to a random vector of the same weight by π, and any code for the binary symmetric channel (with rate $p \approx t/n$) will correct such an error with high probability.

Thus we can construct a sketch with entropy loss $n(h(t/n) - o(1))$ which corrects any t flipped bits with high probability. This is optimal by the lower bound for random errors (Proposition 8.2), since a sketch for data-dependent errors will also correct random errors.

An alternative approach to input-dependent errors is discussed in the last paragraph of Section 8.3.

8.3 Handling Computationally-Bounded Errors Via List Decoding

As mentioned above, many results on noisy coding for other error models in Hamming space extend to secure sketches. The previous sections discussed random, and randomized, errors. In this section, we discuss constructions [Gur03, Lan04, MPSW05] which transform a *list decodable* code, defined below, into uniquely decodable codes for a particular error model. These transformations can also be used in the setting of secure sketches, leading to better tolerance of computationally bounded errors. For some ranges of parameters, this yields optimal sketches, that is, sketches which meet the Shannon bound on the fraction of tolerated errors.

LIST-DECODABLE CODES. A code C in a metric space $\mathcal{M}$ is called *list-decodable* with list size L and distance t if for every point $x \in \mathcal{M}$, there are at most L codewords within distance t of $\mathcal{M}$. A list-decoding algorithm takes as input a word x and returns the corresponding list $c_1, c_2, \ldots$ of codewords. The most interesting setting is when L is a small polynomial (in the description size $\log|\mathcal{M}|$), and there exists an efficient list-decoding algorithm. It is then feasible for an algorithm to go over each word in the list and accept if it has some desirable property. There are many examples of such codes for the Hamming space; for a survey see Guruswami's thesis [Gur01].

Similarly, we can define a *list-decodable secure sketch* with size L and distance t as follows: for any pair of words $w, w' \in \mathcal{M}$ at distance at most t, the algorithm $\mathsf{Rec}(w', \mathsf{SS}(w))$ returns a list of at most L points in $\mathcal{M}$; if $\mathsf{dis}(w, w') \le t$, then one of the words in the list must be w itself. The simplest way to obtain a list-decodable secure sketch is to use the code-offset construction of Section 5 with a list-decodable code for the Hamming space. One obtains a different example by running the improved Juels-Sudan scheme for set difference (Construction 5), replacing ordinary decoding of Reed-Solomon codes with list decoding. This yields a significant improvement in the number of errors tolerated at the price of returning a list of possible candidates for the original secret.

SIEVING THE LIST. Given a list-decodable secure sketch SS, all that's needed is to store some additional information which allows the receiver to disambiguate w from the list. Let's suggestively name the additional information $Tag(w; R)$, where R is some additional randomness (perhaps a key). Given a list-decodable code C, the sketch will typically look like:

$$\mathsf{SS}(w; x = (\ w \oplus C(x),\ Tag(w)\ ).$$

On inputs w' and (Δ, tag), the recovery algorithm consists of running the list decoding algorithm on $w' \oplus \Delta$ to obtain a list of possible codewords $C(x_1), \ldots, C(x_L)$. There is a corresponding list of candidate inputs $w_1, ..., w_L$, where $w_i = C(x_i) \oplus \Delta$, and the algorithm outputs the first w_i in the list such that $Tag(w_i) = tag$. We will choose the function $Tag()$ so that the adversary can not arrange to have two values in the list with valid tags.

We consider two $Tag()$ functions, inspired by [Gur03, Lan04, MPSW05].

1. Recall that for computationally bounded errors, the corrupted string w' depends on *both* w and $\mathsf{SS}(w)$, but w' is computed by a probabilistic circuit of size polynomial in n.

   Consider $Tag(w) = \mathsf{hash}(w)$, where hash is drawn from a collision-resistant function family. More specifically, we will use some extra randomness R to choose a key key for a collision-resistant hash family. The output of the sketch is then

   $$\mathsf{SS}(w; x, r) = (\ w \oplus C(x),\ key(R),\ \mathsf{hash}_{key(R)}(w)\ ).$$

   If the list-decoding algorithm for the code C runs in polynomial time, then the adversary succeeds only if he can find a value $w_i \neq w$ such that $\mathsf{hash}_{key}(w_i) = \mathsf{hash}_{key}(w)$, that is only by finding a collision for the hash function. By assumption, a polynomially-bounded adversary succeeds only with negligible probability.

   The additional entropy loss, beyond that of the code-offset part of the sketch, is bounded above by the output length of the hash function. If α is the desired bound on the adversary's success probability, then for standard assumptions on hash functions this loss will be polynomial in $\log(1/\alpha)$.

   In principle this transformation can yield sketches which achieve the optimal entropy loss $n(h(t/n) - o(1))$, since codes with polynomial list size L are known to exist for error rates approaching the Shannon bound. However, in order to use the construction the code must also be equipped with a reasonably efficient algorithm for finding such a list. This is necessary both so that recovery will be efficient and, more subtly, for the proof of security to go through (that way we can assume that the polynomial-time adversary knows the list of words generated during the recovery procedure). We do not know of *efficient* (i.e. polynomial-time constructible and decodable) binary list-decodable codes which meet the Shannon bound for all choices of parameters. However, when the error rate is near $\frac{1}{2}$ such codes are known [GS00]. Thus, this type of construction yields essentially optimal sketches when the error rate is near $1/2$. This is quite similar to analogous results on channel coding [MPSW05]. Relatively little is known about the performance of efficiently list-decodable codes in other parameter ranges; for a recent survey, see Guruswami's thesis [Gur01].

2. A similar, even simpler, transformation can be used in the setting of input-dependent errors (i.e., when the depend errors only on the input and not on the sketch, but the adversary is not assumed to be computationally bounded). One can store $Tag(w) = I, h_I(w)$ where $\{h_i\}_{i\in\mathcal{I}}$ comes from an pairwise-independent hash family from $\mathcal{M}$ to $\{0,1\}^\ell$, where $\ell = \log\left(\frac{1}{\alpha}\right) + \log L$ and α is the probability of an incorrect decoding.

   The proof is simple: the values $w_1, ..., w_L$ do not depend on I, and so for any value $w_i \neq w$, the probability that $h_I(w_i) = h_I(w)$ is $2^{-\ell}$. There are at most L possible candidates, and so the probability that any one of the elements in the list is accepted is at most $L \cdot 2^{-\ell} = \alpha$ The additional entropy loss incurred is at most $\ell = \log\left(\frac{1}{\alpha}\right) + \log(L)$.

   In principle, this transformation can do as well as the randomization approach of the previous section. However, we do not know of efficient binary list-decodable codes meeting the Shannon bound for most parameter ranges. Thus, in general, randomizing the errors (as in the previous section) works better in the input-dependent setting.

9 Secure Sketches and Efficient Information Reconciliation

Suppose Alice holds a set w and Bob holds a set w' that are close to each other. They wish to reconcile the sets: to discover the symmetric difference $w \triangle w'$ so that they can take whatever appropriate (application-dependent) action to make their two sets agree. Moreover, they wish to do this communication-efficiently, without having to transmit entire sets to each other. This problem is known as set reconciliation and naturally arises in various settings.

Let $(\mathsf{SS}, \mathsf{Rec})$ be a secure sketch for set difference that can handle distance up to t; furthermore, suppose that $|w \triangle w'| \leq t$. Then if Bob receives $s = \mathsf{SS}(w)$ from Alice, he will be able to recover w, and therefore $w \triangle w'$, from s and w'. Similarly, Alice will be able find $w \triangle w'$ upon receiving $s' = \mathsf{SS}(w')$ from Bob. This will be communication-efficient if $|s|$ is small. Note that our secure sketches for set difference of Sections 6.2, 6.3 are indeed short—in fact, they are secure precisely because they are short. Thus, they also make good set reconciliation schemes.

Conversely, a good (single-message) set reconciliation scheme makes a good secure sketch: simply make the message the sketch. The entropy loss will be at most the length of the message, which is short in a communication-efficient scheme. Thus, the set reconciliation scheme CPISync of [MTZ03] makes a good secure sketch. In fact, it is quite similar to the secure sketch of Section 6.2, except instead of the top t coefficients of the characteristic polynomial it uses the values of the polynomial at t points.

PinSketch of Section 6.3, when used for set reconciliation, achieves the same parameters as CPISync of [MTZ03], except decoding is faster, because instead of spending t^3 time to solve a system of linear equations, it spends t^2 time for Euclid's algorithm. Thus, it can be substituted wherever CPISync is used, such as PDA synchronization [STA03] and PGP key server updates [Min]. Furthermore, optimizations that improve computational complexity of CPISync through the use of interaction [MT02] can also be applied to PinSketch.

Of course, secure sketches for other metrics are similarly related to information reconciliation for those metrics. In particular, ideas for edit distance very similar to ours were independently considered in the context of information reconciliation by [CT04].

Acknowledgments

This work evolved over several years and discussions with many people enriched our understanding of the material at hand. In roughly chronological order, we thank Piotr Indyk for discussions about embeddings and for his help in the proof of Lemma 7.3; Madhu Sudan, for helpful discussions about the construction of [JS02] and the uses of error-correcting codes; Venkat Guruswami, for enlightenment about list decoding; Pim Tuyls, for pointing out relevant previous work; Chris Peikert, for pointing out the model of computationally bounded adversaries from [MPSW05]; Ari Trachtenberg, for finding an error in the preliminary version of Appendix E; Ronny Roth, for discussions about efficient BCH decoding; Kevin Harmon and Soren Johnson, for their implementation work; and Silvio Micali, for suggestions on presenting our results.

The work of the Y.D. was partly funded by the National Science Foundation under CAREER Award No. CCR-0133806 and Trusted Computing Grant No. CCR-0311095, and by the New York University Research Challenge Fund 25-74100-N5237. The work of the L.R. was partly funded by the National Science Foundation under Grant No. CCR-0311485. The work of the A.S. was partly funded by US A.R.O. grant DAAD19-00-1-0177 and by a Microsoft Fellowship.

References

[ACM99] *Sixth ACM Conference on Computer and Communication Security*. ACM, November 1999.

[ADG+03] A. Andoni, M. Deza, A. Gupta, P. Indyk, and S. Raskhodnikova. Lower bounds for embedding edit distance into normed spaces. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 523–526, 2003.

[AVZ00] Erik Agrell, Alexander Vardy, and Kenneth Zeger. Upper bounds for constant-weight codes. *IEEE Transactions on Information Theory*, 46(7):2373–2395, 2000.

[BBCM95] Charles H. Bennett, Gilles Brassard, Claude Crépeau, and Ueli M. Maurer. Generalized privacy amplification. *IEEE Transactions on Information Theory*, 41(6):1915–1923, 1995.

[BBCS91] Charles H. Bennett, Gilles Brassard, Claude Crépeau, and Marie-Hélène Skubiszewska. Practical quantum oblivious transfer. In J. Feigenbaum, editor, *Advances in Cryptology—CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 351–366. Springer-Verlag, 1992, 11–15 August 1991.

[BBR88] C. Bennett, G. Brassard, and J. Robert. Privacy amplification by public discussion. *SIAM Journal on Computing*, 17(2):210–229, 1988.

[BCN04] C. Barral, J.-S. Coron, and D. Naccache. Externalized fingerprint matching. Technical Report 2004/021, Cryptology e-print archive, `http://eprint.iacr.org`, 2004.

[BDK+05] Xavier Boyen, Yevgeniy Dodis, Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Secure remote authentication using biometric data. In Ronald Cramer, editor, *Advances in Cryptology—EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 147–163. Springer-Verlag, 2005.

[Bla83] Richard E. Blahut. *Theory and practice of error control codes*. Addison Wesley Longman, Reading, MA, 1983. 512 p.

[Boy04] Xavier Boyen. Reusable cryptographic fuzzy extractors. In *Eleventh ACM Conference on Computer and Communication Security*. ACM, October 25–29 2004.

[Bro97] Andrei Broder. On the resemblence and containment of documents. In *Compression and Complexity of Sequences*, 1997.

[BSSS90] Andries E. Brouwer, James B. Shearer, Neil J. A. Sloane, and Warren D. Smith. A new table of constant weight codes. *IEEE Transactions on Information Theory*, 36(6):1334–1380, 1990.

[CG88] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, 1988.

[CK03] L. Csirmaz and G.O.H. Katona. Geometrical cryptography. In *Proc. International Workshop on Coding and Cryptography*, 2003.

[Cré97] Claude Crépeau. Efficient cryptographic protocols based on noisy channels. In Walter Fumy, editor, *Advances in Cryptology—EUROCRYPT 97*, volume 1233 of *Lecture Notes in Computer Science*, pages 306–317. Springer-Verlag, 11–15 May 1997.

[CT04] V. Chauhan and A. Trachtenberg. Reconciliation puzzles. In *IEEE Globecom, Dallas, TX*, 2004.

[CW79] J.L. Carter and M.N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.

[CZ04] Gérard Cohen and Gilles Zémor. Generalized coset schemes for the wire-tap channel: Application to biometrics. In *IEEE International Symp. on Information Theory*, 2004.

[DFMP99] G.I. Davida, Y. Frankel, B.J. Matt, and R. Peralta. On the relation of error correction and cryptography to an off line biometric based identification scheme. In *Proceedings of WCC99, Workshop on Coding and Cryptography, Paris, France*, 11-14 January 1999.

[DGL04] Yan Zhong Ding, P. Gopalan, and Richard J. Lipton. Error correction against computationally bounded adversaries. Manuscript. Appeared initially as [Lip94], 2004.

[Din05] Yan Zong Ding. Error correction in the bounded storage model. In Kilian [Kil05].

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.

[EHMS00] Carl Ellison, Chris Hall, Randy Milbert, and Bruce Schneier. Protecting keys with personal entropy. *Future Generation Computer Systems*, 16:311–318, February 2000.

[FJ01] Niklas Frykholm and Ari Juels. Error-tolerant password recovery. In *Eighth ACM Conference on Computer and Communication Security*, pages 1–8. ACM, November 5–8 2001.

[For66] G. David Forney. *Concatenated Codes*. PhD thesis, MIT, 1966.

[Fry00] N. Frykholm. Passwords: Beyond the terminal interaction model. Master's thesis, Umeå University, 2000.

[GS00] Venkatesan Guruswami and Madhu Sudan. List decoding algorithms for certain concatenated codes. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 181–190, Portland, Oregon, 21–23 May 2000.

[Gur01] V. Guruswami. *List Decoding of Error-Correcting Codes*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2001.

[Gur03] Venkatesan Guruswami. List decoding with side information. In *IEEE Conference on Computational Complexity*, pages 300–. IEEE Computer Society, 2003.

[HILL99] J. Håstad, R. Impagliazzo, L.A. Levin, and M. Luby. Construction of pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HJR] Kevin Harmon, Soren Johnson, and Leonid Reyzin. An implementation of syndrome encoding and decoding for binary BCH codes, secure sketches and fuzzy extractors. Available at `http://www.cs.bu.edu/~reyzin/code/fuzzy.html`.

[JS02] Ari Juels and Madhu Sudan. A fuzzy vault scheme. In *IEEE International Symposium on Information Theory*, 2002.

[JW99] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In ACM [ACM99], pages 28–36.

[Kil05] Joe Kilian, editor. *First Theory of Cryptography Conference — TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*. Springer-Verlag, February 10–12 2005.

[KO63] A.A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, 7:595–596, 1963.

[KS95] E. Kaltofen and V. Shoup. Subquadratic-time factoring of polynomials over finite fields. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 398–406, Las Vegas, Nevada, 29 May–1 June 1995.

[KSHW97] John Kelsey, Bruce Schneier, Chris Hall, and David Wagner. Secure applications of low-entropy keys. In Eiji Okamoto, George I. Davida, and Masahiro Mambo, editors, *ISW*, volume 1396 of *Lecture Notes in Computer Science*, pages 121–134. Springer, 1997.

[Lan04] Michael Langberg. Private codes or succinct random codes that are (almost) perfect. In *FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS'04)*, pages 325–334, Washington, DC, USA, 2004. IEEE Computer Society.

[Lip94] Richard J. Lipton. A new approach to information theory. In Patrice Enjalbert, Ernst W. Mayr, and Klaus W. Wagner, editors, *STACS*, volume 775 of *Lecture Notes in Computer Science*, pages 699–708. Springer, 1994. The full version of this paper is in preparation [DGL04].

[LT03] J.-P. M. G. Linnartz and P. Tuyls. New shielding functions to enhance privacy and prevent misuse of biometric templates. In *AVBPA*, pages 393–402, 2003.

[Mau93] Ueli Maurer. Secret key agreement by public discussion from common information. *IEEE Transactions on Information Theory*, 39(3):733–742, 1993.

[Min] Yaron Minsky. The SKS OpenPGP key server. `http://www.nongnu.org/sks`.

[MPSW05] Silvio Micali, Chris Peikert, Madhu Sudan, and David Wilson. Optimal error correction against computationally bounded noise. In Kilian [Kil05].

[MRLW01a] Fabian Monrose, Michael K. Reiter, Qi Li, and Susanne Wetzel. Cryptographic key generation from voice. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2001.

[MRLW01b] Fabian Monrose, Michael K. Reiter, Qi Li, and Susanne Wetzel. Using voice to generate cryptographic keys. In *2001: A Speaker Odyssey. The Speaker Recognition Workshop*, 2001.

[MRW99] Fabian Monrose, Michael K. Reiter, and Susanne Wetzel. Password hardening based on keystroke dynamics. In ACM [ACM99], pages 73–82.

[MT79] Robert Morris and Ken Thomson. Password security: A case history. *Communications of the ACM*, 22(11):594–597, 1979.

[MT02] Yaron Minsky and Ari Trachtenberg. Scalable set reconciliation. In *40th Annual Allerton Conference on Communication, Control and Computing, Monticello, IL*, October 2002. See also tehcnial report BU-ECE-2002-01.

[MTZ03] Yaron Minsky, Ari Trachtenberg, and Richard Zippel. Set reconciliation with nearly optimal communication complexity. *IEEE Transactions on Information Theory*, 49(9):2213–2218, 2003.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–53, 1996.

[OR05] Rafail Ostrovsky and Yuval Rabani. Low distortion embeddings for edit distance. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, pages 218–224, Baltimore, Maryland, 22–24 May 2005.

[RTS00] Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Computing*, 13(1):2–24, 2000.

[RW04] Renato Renner and Stefan Wolf. Smooth rényi entropy and applications. In *Proceedings of IEEE International Symposium on Information Theory*, page 233, June 2004.

[RW05] Renato Renner and Stefan Wolf. Simple and tight bounds for information reconciliation and privacy amplification. In Bimal Roy, editor, *Advances in Cryptology—ASIACRYPT 2005*, Lecture Notes in Computer Science, Chennai, India, 4–8 December 2005. Springer-Verlag.

[Sha48] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948. Reprinted in D. Slepian, editor, *Key Papers in the Development of Information Theory*, IEEE Press, NY, 1974.

[Sha02] Ronen Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the EATCS*, 77:67–95, 2002.

[Sho01] Victor Shoup. A proposal for an ISO standard for public key encryption. Available at `http://eprint.iacr.org/2001/112`, 2001.

[Sho05] Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2005. Available from `http://shoup.net`.

[SKHN75] Yasuo Sugiyama, Masao Kasahara, Shigeichi Hirasawa, and Toshihiko Namekawa. A method for solving key equation for decoding Goppa codes. *Information and Control*, 27(1):87–99, 1975.

[STA03] David Starobinski, Ari Trachtenberg, and Sachin Agarwal. Efficient PDA synchronization. *IEEE Transactions on Mobile Computing*, 2(1):40–51, 2003.

[Sud01] Madhu Sudan. Lecture notes for an algorithmic introduction to coding theory. Course taught at MIT, December 2001.

[TG04] Pim Tuyls and Jasper Goseling. Capacity and examples of template-protecting biometric authentication systems. In Davide Maltoni and Anil K. Jain, editors, *ECCV Workshop BioAW*, volume 3087 of *Lecture Notes in Computer Science*, pages 158–170. Springer, 2004.

[vL92] J.H. van Lint. *Introduction to Coding Theory*. Springer-Verlag, 1992.

[VTDL03] E. Verbitskiy, P. Tuyls, D. Denteneer, and J.-P. Linnartz. Reliable biometric authentication with privacy protection. In *Proc. 24th Benelux Symposium on Information theory*, 2003.

[vzGG03] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2003.

[WC81] M.N. Wegman and J.L. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22:265–279, 1981.

A Lower Bounds from Coding

Recall that an $(\mathcal{M}, K, t)$ code is a subset of the metric space $\mathcal{M}$ which can *correct* t errors (this is slightly different from the usual notation of coding theory literature).

Let $K(\mathcal{M}, t)$ be the largest K for which there exists and $(\mathcal{M}, K, t)$-code. Given any set S of 2^m points in $\mathcal{M}$, we let $K(\mathcal{M}, t, S)$ be the largest K such that there exists an $(\mathcal{M}, K, t)$-code all of whose K points belong to S. Finally, we let $L(\mathcal{M}, t, m) = \log(\min_{|S|=2^m} K(n, t, S))$. Of course, when $m = \log |\mathcal{M}|$, we get $L(\mathcal{M}, t, n) = \log K(\mathcal{M}, t)$. The exact determination of quantities $K(\mathcal{M}, t)$ and $K(\mathcal{M}, t, S)$ form the main problem of coding theory, and is typically very hard. To the best of our knowledge, the quantity $L(\mathcal{M}, t, m)$ was not explicitly studied in any of three metrics that we study, and its exact determination seems very hard as well.

We give two simple lower bounds on the entropy loss (one for secure sketches, the other for fuzzy extractors) which, somewhat surprisingly, show that our constructions for the Hamming and set difference metrics are essentially optimal, at least when the original input distribution is uniform.

Lemma A.1. *The existence of* $(\mathcal{M}, m, m', t)$ *secure sketch implies that* $m' \leq L(\mathcal{M}, t, m)$. *In particular, when* $m = \log |\mathcal{M}|$ *(i.e., when the password is truly uniform),* $m' \leq \log K(\mathcal{M}, t)$.

Proof. Assume SS is such secure sketch. Let S be any set of size 2^m in $\mathcal{M}$, and let W be uniform over S. Then we must have $\tilde{\mathbf{H}}_\infty(W \mid \mathsf{SS}(W)) \geq m'$. In particular, there must be some value v such that $\mathbf{H}_\infty(W \mid \mathsf{SS}(W) = v) \geq m'$. But this means that conditioned on $\mathsf{SS}(W) = v$, there are at least $2^{m'}$ points

w in S (call this set T) which could produce $\mathsf{SS}(W) = v$. We claim that these $2^{m'}$ values of w form a code of error-correcting distance t. Indeed, otherwise there would be a point $w' \in \mathcal{M}$ such that $\mathsf{dis}(w_0, w') \leq t$ and $\mathsf{dis}(w_1, w') \leq t$ for some $w_0, w_1 \in T$. But then we must have that $\mathsf{Rec}(w', v)$ is equal to both w_0 and w_1, which is impossible. Thus, the set T above must form an $(\mathcal{M}, 2^{m'}, t)$-code inside S, which means that $m' \leq \log K(\mathcal{M}, t, S)$. Since S was arbitrary, the bound follows. □

Lemma A.2. *The existence of $(\mathcal{M}, m, \ell, t, \epsilon)$-fuzzy extractors implies that $\ell \leq L(\mathcal{M}, t, m) - \log(1-\epsilon)$. In particular, when $m = \log |\mathcal{M}|$ (i.e., when the password is truly uniform), $\ell \leq \log K(\mathcal{M}, t) - \log(1-\epsilon)$.*

Proof. Assume $(\mathsf{Gen}, \mathsf{Rep})$ is such a fuzzy extractor. Let S be any set of size 2^m in $\mathcal{M}$, and let W be uniform over S. Then we must have $\mathbf{SD}\left((R, P), (U_\ell, P)\right) \leq \epsilon$. In particular, there must be some value p of P such that R is ϵ-close to U_ℓ conditioned on $P = p$. In particular, this means that conditioned on $P = p$, there are at least $(1-\epsilon)2^\ell$ points $r \in \{0,1\}^\ell$ (call this set T) which could be extracted with $P = p$. Now, map every $r \in T$ to some arbitrary $w \in S$ which could have produced r with nonzero probability given $P = p$, and call this map C. C must define a code with error-correcting distance t by the same reasoning as in Lemma A.1. □

Observe that, as long as $\epsilon < 1/2$, we have $0 < -\log(1-\epsilon) < 1$, so the lowerbounds on secure sketches and fuzzy extractors differ by less than a bit.

B Lowering Input Entropy Preserves Entropy Loss

The following lemma shows that a secure sketch that is good for high min-entropy distributions is also good for lower min-entropy distributions. As already stated above, the converse does not necessarily hold.

Lemma B.1. *Suppose f is a probabilistic function of x: each $x \in X$ defines a probability distribution which we denote by $f(x)$. Suppose further that for some $m \leq \log |X|$ and λ, for any distribution A on X such that $\mathbf{H}_\infty(A) = m$, we have $\tilde{\mathbf{H}}_\infty(A|f(A)) \geq m - \lambda$. Then for any distribution B on X such that $\mathbf{H}_\infty(B) \leq m$, we have $\tilde{\mathbf{H}}_\infty(B|f(B)) \geq \mathbf{H}_\infty(B) - \lambda$.*

Proof.

$$\begin{aligned}
\tilde{\mathbf{H}}_\infty(B \mid f(B)) &= -\log \mathbb{E}_y \left[\max_x \Pr[B = x \mid f(B) = y]\right] \\
&= -\log \sum_y \max_x \Pr[B = x \wedge f(B) = y] \\
&= -\log \sum_y \max_x \Pr[f(x) = y \wedge B = x] \\
&= -\log \sum_y \max_x \Pr[f(x) = y] \Pr[B = x] \,.
\end{aligned}$$

Now design a distribution A as follows. Start with the distribution B. For each x such that $\Pr[B = x] > 2^{-m}$, set $\Pr[A = x] = 2^{-m}$. This will decrease the total sum of the probabilities below 1; to compensate, increase the probability of every other x, in turn, up to at most 2^{-m}, until the total sum of the probabilities is 1 (this is possible because $m \leq \log |X|$). It is important to note that as a result, for each x, either $\Pr[A = x] \geq \Pr[B = x]$ or $\Pr[A = x] = 2^{-m}$. Let $h = \mathbf{H}_\infty(B) \leq m$. Then $\Pr[B = x] \leq 2^{-h}$. Therefore for each x, $\Pr[B=x] \le 2^{m-h}\Pr[A=x]$. Therefore, for each y, $\max_x \Pr[f(x)=y]\Pr[B=x] \le 2^{m-h}\max_x \Pr[f(x)=y]\Pr[A=x]$. Plugging this in above, we get

$$\begin{aligned}\tilde{\mathbf{H}}_\infty(B \mid f(B)) &\ge -\log 2^{m-h}\sum_y \max_x \Pr[f(x)=y]\Pr[A=x] \\ &= h-m+\tilde{\mathbf{H}}_\infty(A \mid f(A)) = h-\lambda.\end{aligned}$$

□

C On Smooth Variants of Average Min-Entropy and the Relationship to Smooth Rényi Entropy

Min-entropy is a rather fragile measure: a single high-probability element can ruin the min-entropy of an otherwise good distribution. This is often circumvented within proofs by considering a distribution which is close to the distribution of interest, but which has higher entropy. Renner and Wolf [RW04] systematized this approach with the notion of ϵ-*smooth* min-entropy (they use the term “Rényi entropy of order ∞” instead of “min-entropy”), which considers all distributions that are ϵ-close:

$$\mathbf{H}_\infty^\epsilon(A) = \max_{B\colon\, \mathbf{SD}(A,B)\le\epsilon} \mathbf{H}_\infty(B)\,.$$

Smooth min-entropy very closely relates to the amount of extractable nearly-uniform randomness: if one can map A to a distribution that is ϵ-close to U_m, then $\mathbf{H}_\infty^\epsilon(A) \ge m$; conversely, from any A such that $\mathbf{H}_\infty^\epsilon(A) \ge m$, and for any ϵ_2, one can extract $m-2\log\left(\frac{1}{\epsilon_2}\right)$ bits that are $\epsilon+\epsilon_2$-close to uniform (see [RW04] for a more precise statement; the proof of the first statement follows by considering the inverse map, and the proof of the second from the leftover hash lemma, which is discussed in more detail at Lemma 4.2). For some distributions, considering the smooth min-entropy will improve the number and quality of extractable random bits.

A smooth version of average min-entropy can also be considered, defined as

$$\tilde{\mathbf{H}}_\infty^\epsilon(A \mid B) = \max_{(C,D)\colon\, \mathbf{SD}((A,B),(C,D))\le\epsilon} \tilde{\mathbf{H}}_\infty(C \mid D)\,.$$

It similarly relates very closely to the number of extractable bits that look nearly-uniform to the adversary who knows the value of B, and is therefore perhaps a better measure for the quality of a secure sketch that is used to obtain a fuzzy extractor. All our results can be cast in terms of smooth entropies throughout, with appropriate modifications (if input entropy is ϵ-smooth, then output entropy will also be ϵ-smooth, and extracted random strings will be ϵ further away from uniform). We avoid doing so for simplicity of exposition. However, for some input distributions, particularly ones with few elements of relatively high probability, this will improve the result by giving more secure sketches or longer-output fuzzy extractors.

Finally, a word is in order on the relation of average min-entropy to conditional min-entropy, introduced by Renner and Wolf in [RW05], and defined as $\mathbf{H}_\infty(A \mid B) = -\log\max_{a,b}\Pr(A=a \mid B=b) = \min_b \mathbf{H}_\infty(A \mid B=b)$ (an ϵ-smooth version is defined analogously by considering all distributions (C,D) that are within ϵ of (A,B) and taking the maximum among them). This definition is too strict: it takes the worst-case b, while for randomness extraction (and many other settings, such as predictability by an adversary), average-case b suffices. Average min-entropy leads to more extractable bits. The following relations hold between the two notions: $\tilde{\mathbf{H}}_\infty^\epsilon(A \mid B) \ge \mathbf{H}_\infty^\epsilon(A \mid B)$ and $\mathbf{H}_\infty{}^{\epsilon+\epsilon_2}(A \mid B) \ge \tilde{\mathbf{H}}_\infty^\epsilon(A \mid B) -$

$\log\left(\frac{1}{\epsilon_2}\right)$ (for the case of $\epsilon = 0$, this follows by constructing a new distribution that eliminates all b for which $\mathbf{H}_\infty(A \mid B = b) < \tilde{\mathbf{H}}_\infty(A \mid B) - \log\left(\frac{1}{\epsilon_2}\right)$, which will be within ϵ_2 of the (A, B) by Markov's inequality; for $\epsilon > 0$, an analogous proof works). Note that by Lemma 3.1, this implies a simple chain rule for $\mathbf{H}^\epsilon_\infty$ (a more general one is given in [RW05, Section 2.4]): $\mathbf{H}_\infty{}^{\epsilon+\epsilon_2}(A \mid B) \ge \tilde{\mathbf{H}}^\epsilon_\infty((A, B)) - H_0(B) - \log\left(\frac{1}{\epsilon_2}\right)$, where $H_0(B)$ is the logarithm of the number of possible values of B.

D Analysis of the Original Juels-Sudan Construction

In this section we present a new analysis for the Juels-Sudan secure sketch for set difference. We will assume that $n = |\mathcal{U}|$ is a prime power and work over the field $\mathcal{F} = GF(n)$. On input set w, the original Juels-Sudan sketch is a list of r pairs of points (x_i, y_i) in $\mathcal{F}$, for some parameter r, $s < r \le n$. It is computed as follows:

1. Choose $p()$ at random from the set of polynomials of degree at most $k = s - t - 1$ over $\mathcal{F}$. Write $w = \{x_1, ..., x_s\}$, and let $y_i = p(x_i)$ for $i = 1, ..., s$.
2. Choose $r - s$ distinct points $x_{s+1}, ..., x_r$ at random from $\mathcal{F} - w$.
3. For $i = s + 1, ..., r$, choose $y_i \in \mathcal{F}$ at random such that $y_i \ne p(x_i)$.
4. Output $\mathsf{SS}(w) = \{(x_1, y_1), ..., (x_r, y_r)\}$ (in lexicographic order of x_i).

The parameter r dictates the amount of storage necessary, one on hand, and also the security of the scheme (that is, for $r = s$ the scheme leaks all information and for larger and larger r there is less information about w). Juels and Sudan actually propose two analyses for the scheme. First, they analyze the case where the secret w is distributed uniformly over all subsets of size s. Second, they provide an analysis of a nonuniform password distribution, but only for the case $r = n$ (that is, their analysis only applies in the small universe setting, where $\Omega(n)$ storage is acceptable). Here we give a simpler analysis which handles nonuniformity and any $r \le n$. We get the same results for a broader set of parameters.

Lemma D.1. *The entropy loss of the Juels-Sudan scheme is at most* $t \log n + \log\binom{n}{r} - \log\binom{n-s}{r-s} + 2$.

Proof. This is a simple application of Lemma 3.1. $\mathbf{H}_\infty((W, \mathsf{SS}(W)))$ can be computed as follows. Choosing the polynomial p (which can be uniquely recovered from w and $\mathsf{SS}(w)$) requires $s - t$ random choices from $\mathcal{F}$. The choice of the remaining x_i's requires $\log\binom{n-s}{r-s}$ bits, and choosing the $y_i's$ requires $r - s$ random choices from $\mathcal{F} - \{p(x_i)\}$. Thus, $\mathbf{H}_\infty((W, \mathsf{SS}(W))) = \mathbf{H}_\infty(W) + (s - t)\log n + \log\binom{n-s}{r-s} + (r - s)\log(n - 1)$. The output can be described in $\log\left(\binom{n}{r} n^r\right)$ bits. The result follows by Lemma 3.1 after observing that $(r - s)\log\frac{n}{n-1} < n \log\frac{n}{n-1} \le 2$. □

In the large universe setting, we will have $r \ll n$ (since we wish to have storage polynomial in s). In that setting, the bound on the entropy loss of the Juels-Sudan scheme is in fact very large. We can rewrite the entropy loss as $t \log n - \log\binom{r}{s} + \log\binom{n}{s} + 2$, using the identity $\binom{n}{r}\binom{r}{s} = \binom{n}{s}\binom{n-s}{r-s}$. Now the entropy of W is at most $\binom{n}{s}$, and so our lower bound on the remaining entropy is $(\log\binom{r}{s} - t \log n - 2)$. To make this quantity large requires making r very large.

E BCH Syndrome Decoding in Sublinear Time

We show that the standard decoding algorithm for BCH codes can be modified to run in time polynomial in the length of the syndrome. This works for BCH codes over any field $GF(q)$, which include Hamming codes in the binary case and Reed-Solomon for the case $n = q - 1$. BCH codes are handled in detail in many textbooks (e.g., [vL92]); our presentation here is quite terse. For simplicity, we only discuss primitive, narrow-sense BCH codes here; the discussion extends easily to the general case.

The algorithm discussed here has been revised due to an error pointed out by Ari Trachtenberg. Its implementation is available [HJR].

We'll use a slightly non-standard formulation of BCH codes. Let $n = q^m - 1$ (in the binary case of interest in Section 6.3, $q = 2$). We will work in two finite fields: $GF(q)$ and a larger extension field $\mathcal{F} = GF(q^m)$. BCH codewords, formally defined below, are then vectors in $GF(q)^n$. In most common presentations, one indexes the n positions of these vectors by discrete logarithms of the elements of $\mathcal{F}^*$: position i, for $1 \le i \le n$, corresponds to α^i, where α generates the multiplicative group $\mathcal{F}^*$. However, there is no inherent reason to do so: they can be indexed by elements of $\mathcal{F}$ directly rather than by their discrete logarithms. Thus, we say that a word has value p_x at position x, where $x \in \mathcal{F}^*$. If one ever needs to write down the entire n-character word in an ordered fashion, one can choose arbitrarily a convenient ordering of the elements of $\mathcal{F}$ (e.g., by using some standard binary representation of field elements); for our purposes this is not necessary, as we do not store entire n-bit words explicitly, but rather represent them by their supports: $\mathsf{supp}(v) = \{(x, p_x) \mid p_x \neq 0\}$. Note that for the binary case of interest in Section 6.3, we can define $\mathsf{supp}(v) = \{x \mid p_x \neq 0\}$, because p_x can take only two values: 0 or 1.

Our choice of representation will be crucial for efficient decoding: in the more common representation, the last step of the decoding algorithm requires one to find the position i of the error from the field element α^i. However, no efficient algorithms for computing discrete logarithm are known if q^m is large (indeed, a lot of cryptography is based on the assumption that such efficient algorithm does not exist). In our representation, the field element α^i will in fact be the position of the error.

Definition 6. The (narrow-sense, primitive) BCH code of designed distance δ over $GF(q)$ (of length $n \ge \delta$) is given by the set of vectors of the form $\left(c_x\right)_{x \in \mathcal{F}^*}$ such that each c_x is in the smaller field $GF(q)$, and the vector satisfies the constraints $\sum_{x \in \mathcal{F}^*} c_x x^i = 0$, for $i = 1, \ldots, \delta - 1$, with arithmetic done in the larger field $\mathcal{F}$.

To explain this definition, let us fix a generator α of the multiplicative group of the large field $\mathcal{F}^*$. For any vector of coefficients $\left(c_x\right)_{x \in \mathcal{F}^*}$, we can define a polynomial

$$c(z) = \sum_{x \in GF(q^m)^*} c_x z^{\mathsf{dlog}(x)}$$

where $\mathsf{dlog}(x)$ is the discrete logarithm of x with respect to α. The conditions of the definition are then equivalent to the requirement (more commonly seen in presentations of BCH codes) that $c(\alpha^i) = 0$ for $i = 1, \ldots, \delta - 1$, because $(\alpha^i)^{\mathsf{dlog}(x)} = (\alpha^{\mathsf{dlog}(x)})^i = x^i$.

We can simplify this somewhat. Because the coefficients c_x are in $GF(q)$, they satisfy $c_x^q = c_x$. Using the identity $(x + y)^q = x^q + y^q$, which holds even in the large field $\mathcal{F}$, we have $c(\alpha^i)^q = \sum_{x \neq 0} c_x^q x^{iq} = c(\alpha^{iq})$. Thus, roughly a $1/q$ fraction of the conditions in the definition are redundant: we only need to check that they hold for $i \in \{1, \ldots, \delta - 1\}$ such that $q \nmid i$.

The syndrome of a word (not necessarily a codeword) $(p_x)_{x \in \mathcal{F}^*} \in GF(q)^n$ with respect to the BCH code above is the vector

$$\mathsf{syn}(p) = p(\alpha^1), \ldots, p(\alpha^{\delta-1}), \quad \text{where} \quad p(\alpha^i) = \sum_{x \in \mathcal{F}^*} p_x x^i.$$

As mentioned above, we do not in fact have to include the values $p(\alpha^i)$ such that $q|i$.

COMPUTING WITH LOW-WEIGHT WORDS. A low-weight word $p \in GF(q)^n$ can be represented either as a long string or, more compactly, as a list of positions where it is nonzero and its values at those points. We call this representation the support list of p and denote it $\mathsf{supp}(p) = \{(x, p_x)\}_{x:p_x \neq 0}$.

Lemma E.1. *For a q-ary BCH code C of designed distance δ, one can compute:*

- $\mathsf{syn}(p)$ *from* $\mathsf{supp}(p)$ *in time polynomial in* δ, $\log n$, *and* $|\mathsf{supp}(p)|$, *and*
- $\mathsf{supp}(p)$ *from* $\mathsf{syn}(p)$ *(when p has weight at most $(\delta - 1)/2$), in time polynomial in δ and $\log n$.*

Proof. Recall that $\mathsf{syn}(p) = (p(\alpha), ..., p(\alpha^{\delta-1}))$ where $p(\alpha^i) = \sum_{x \neq 0} p_x x^i$. Part (1) is easy, since to compute the syndrome we only need to compute the powers of x. This requires about $\delta \cdot \mathsf{weight}(p)$ multiplications in $\mathcal{F}$. For Part (2), we adapt Berlekamp's BCH decoding algorithm, based on its presentation in [vL92]. Let $M = \{x \in \mathcal{F}^* | p_x \neq 0\}$, and define

$$\sigma(z) \stackrel{\text{def}}{=} \prod_{x \in M} (1 - xz) \quad \text{and} \quad \omega(z) \stackrel{\text{def}}{=} \sigma(z) \sum_{x \in M} \frac{p_x xz}{(1 - xz)}$$

Since $(1 - xz)$ divides $\sigma(z)$ for $x \in M$, we see that $\omega(z)$ is in fact a polynomial of degree at most $|M| = \mathsf{weight}(p) \leq (\delta - 1)/2$. The polynomials $\sigma(z)$ and $\omega(z)$ are known as the error locator polynomial and evaluator polynomial, respectively; observe that $\gcd(\sigma(z), \omega(z)) = 1$.

We will in fact work with our polynomials modulo z^δ. In this arithmetic the inverse of $(1 - xz)$ is $\sum_{\ell=1}^{\delta} (xz)^{\ell-1}$, that is

$$(1 - xz) \sum_{\ell=1}^{\delta} (xz)^{\ell-1} \equiv 1 \mod z^\delta.$$

We are given $p(\alpha^\ell)$ for $\ell = 1, ..., \delta$. Let $S(z) = \sum_{\ell=1}^{\delta-1} p(\alpha^\ell) z^\ell$. Note that $S(z) \equiv \sum_{x \in M} p_x \frac{xz}{(1-xz)} \mod z^\delta$. This implies that

$$S(z)\sigma(z) \equiv \omega(z) \mod z^\delta.$$

The polynomials $\sigma(z)$ and $\omega(z)$ satisfy the following four conditions: they are of degree at most $(\delta-1)/2$ each, they are relatively prime, the constant coefficient of σ is 1, and they satisfy this congruence. In fact, let $w'(z), \sigma'(z)$ be any nonzero solution this congruence, where degrees of $w'(z)$ and $\sigma'(z)$ are at most $(\delta - 1)/2$. Then $w'(z)/\sigma'(z) = \omega(z)/\sigma(z)$. (To see why this is so, multiply the initial congruence by $\sigma'()$ to get $\omega(z)\sigma'(z) \equiv \sigma(z)\omega'(z) \mod z^\delta$. Since the both sides of the congruence have degree at most $\delta - 1$, they are in fact equal as polynomials.) Thus, there is at most one solution $\sigma(z), \omega(z)$ satisfying all four conditions, which can be obtained from any $\sigma'(z), \omega'(z)$ by reducing the resulting fraction $\omega'(z)/\sigma'(z)$ to obtain the solution of minimal degree with the constant term of σ equal to 1.

Finally, the roots of $\sigma(z)$ are the points x^{-1} for $x \in M$, and the exact value of p_x can be recovered from $\omega(x^{-1}) = p_x \prod_{y \in M, y \neq x} (1 - yx^{-1})$ (this is only needed for $q > 2$, because for $q = 2$, $p_x = 1$). Note that it is possible that a solution to the congruence will be found even if the input syndrome is not a syndrome of any p with $\mathsf{weight}(p) > (\delta - 1)/2$ (it is also possible that a solution to the congruence will not be found at all, or that the resulting $\sigma(z)$ will not split into distinct nonzero roots). Such a solution will not give the correct p. Thus, if there is no guarantee that $\mathsf{weight}(p)$ is actually at most $(\delta - 1)/2$, it is necessary to recompute $\mathsf{syn}(p)$ after finding the solution, in order to verify that p is indeed correct.

Representing coefficients of $\sigma'(z)$ and $\omega'(z)$ as unknowns, we see that solving the congruence requires only solving a system of δ linear equations (one for each degree of z, from 0 to $\delta-1$) involving $\delta+1$ variables over $\mathcal{F}$, which can be done in $O(\delta^3)$ operations in $\mathcal{F}$ using, e.g., Gaussian elimination. The reduction of the fraction $\omega'(z)/\sigma'(z)$ requires simply running Euclid's algorithm for finding the g.c.d. of two polynomials of degree less than δ, which takes $O(\delta^2)$ operations in $\mathcal{F}$. Suppose the resulting σ has degree e. Then one can find the roots of σ as follows. First test that σ indeed has e distinct roots by testing that $\sigma(z)|z^{q^m}-z$ (this is a necessary and sufficient condition, because every element of $\mathcal{F}$ is a root of $z^{q^m}-z$ exactly once). This can be done by computing $(z^{q^m} \bmod \sigma(z))$ and testing if it equals $z \bmod \sigma$; it takes m exponentiations of a polynomial to the power q, i.e., $O((m\log q)e^2)$ operations in $\mathcal{F}$. Then apply an equal-degree-factorization algorithm (e.g., as described in [Sho05]), which also takes $O((m\log q)e^2)$ operations in $\mathcal{F}$. Finally, after taking inverses of the roots of $\mathcal{F}$ and finding p_x (which takes $O(e^2)$ operations in $\mathcal{F}$), recompute $\mathsf{syn}(p)$ to verify that it is equal to the input value.

Because $m\log q = \log(n+1)$ and $e \le (\delta-1)/2$, the total running time is $O(\delta^3+\delta^2\log n)$ operations in $\mathcal{F}$; each operation in $\mathcal{F}$ can done in time $O(\log^2 n)$, or faster using advanced techniques.

One can improve this running time substantially. The error locator polynomial $\sigma()$ can be found in $O(\log\delta)$ convolutions (multiplications) of polynomials over $\mathcal{F}$ of degree $(\delta-1)/2$ each [Bla83, Section 11.7] by exploiting the special structure of the system of linear equations being solved. Each convolution can be performed asymptotically in time $O(\delta\log\delta\log\log\delta)$ (see, e.g., [vzGG03]), the total time required to find σ gets reduced to $O(\delta\log^2\delta\log\log\delta)$ operation in $\mathcal{F}$. This replaces the δ^3 term in the above running time.

While this is asymptotically very good, Euclidean-algorithm-based decoding [SKHN75], which runs in $O(\delta^2)$ operations in $\mathcal{F}$, will find $\sigma(z)$ faster for reasonable values of δ (certainly for $\delta < 1000$). The algorithm finds σ as follows:

```
set R_old(z) ← z^(δ−1), R_cur(z) ← S(z)/z, V_old(z) ← 0, V_cur(z) ← 1.
while deg(R_cur(z)) ≥ (δ − 1)/2:
      divide R_old(z) by R_cur(z) to get quotient q(z) and remainder R_new(z);
      set V_new(z) ← V_old(z) − q(z)V_cur(z);
      set R_old(z) ← R_cur(z), R_cur(z) ← R_new(z), V_old(z) ← V_cur(z), V_cur(z) ← V_new(z).
set c ← V_cur(0); set σ(z) ← V_cur(z)/c and ω(z) ← z · R_cur(z)/c
```

In the above algorithm, if $c=0$, then the correct $\sigma(z)$ does not exist, i.e., $\mathsf{weight}(p) > (\delta-1)/2$. The correctness of this algorithm can be seen by observing that the congruence $S(z)\sigma(z) \equiv \omega(z) \pmod{z^\delta}$ can have z factored out of it (because $S(z)$, $\omega(z)$ and z^δ are all divisible by z) and rewritten as $(S(z)/z)\sigma(z) + u(z)z^{\delta-1} = \omega(z)/z$, for some $u(z)$. The obtained σ is easily shown to be the correct one (if one exists at all) by applying [Sho05, Theorem 18.7] (to use the notation of that theorem, set $n = z^{\delta-1}, y = S(z)/z, t^* = r^* = (\delta-1)/2, r' = \omega(z)/z, s' = u(z), t' = \sigma(z)$).

The root finding of σ can also be sped up. Asymptotically, detecting if a polynomial over $\mathcal{F} = GF(q^m) = GF(n+1)$ of degree e has e distinct roots and finding these roots can be performed in time $O(e^{1.815}(\log n)^{0.407})$ operations in $\mathcal{F}$ using the algorithm of Kaltofen and Shoup [KS95], or in time $O(e^2 + (\log n)e\log e\log\log e)$ operations in $\mathcal{F}$ using the EDF algorithm of Cantor and Zassenhaus[11]. For reasonable values of e, the Cantor-Zassenhaus EDF algorithm with Karatsuba's multiplication algorithm [KO63] for polynomials will be faster, giving root-finding running time of $O(e^2 + e^{\log_2 3}\log n)$ operations in $\mathcal{F}$. Note that if the actual weight e of p is close to the maximum tolerated $(\delta-1)/2$, then finding the roots of σ will actually take longer than finding σ. □

[11] See [Sho05, Section 21.3], and substitute the most efficient known polynomial arithmetic. For example, the procedures described in [vzGG03] take time $O(e\log e\log\log e)$ instead of time $O(e^2)$ to perform modular arithmetic operations with degree-e polynomials.

A DUAL VIEW OF THE ALGORITHM. Readers may be used to seeing a different, evaluation-based formulation of BCH codes, in which codewords are generated as follows. Let $\mathcal{F}$ again be an extension of $GF(q)$, and let n be the length of the code (note that $|\mathcal{F}^*|$ is not necessarily equal to n in this formulation). Fix distinct $x_1, x_2, \ldots, x_n \in \mathcal{F}$. For every polynomial c over the large field $\mathcal{F}$ of degree at most $n - \delta$, the vector $(c(x_1), c(x_2), \ldots c(x_n))$ is a codeword if and only if every coordinate of the vector happens to be in the smaller field: $c(x_i) \in GF(q)$ for all i. In particular, when $\mathcal{F} = GF(q)$, then every polynomial leads to a codeword, thus giving Reed-Solomon codes.

The syndrome in this formulation can be computed as follows: given a vector $y = (y_1, y_2, \ldots, y_n)$ find the interpolating polynomial $P = p_{n-1}x^{n-1} + p_{n-2}x^{n-2} + \cdots + p_0$ over $\mathcal{F}$ of degree at most $n - 1$ such that $P(x_i) = y_i$ for all i. The syndrome is then the negative top $\delta - 1$ coefficients of P: $\mathsf{syn}(y) = (-p_{n-1}, -p_{n-2}, \ldots, -p_{n-(\delta-1)})$. (It is easy to see that this is a syndrome: it is a linear function that is zero exactly on the codewords.)

When $n = |\mathcal{F}| - 1$, we can index the n-component vectors by elements of $\mathcal{F}^*$, writing codewords as $(c(x))_{x\in\mathcal{F}^*}$. In this case, the syndrome of $(y_x)_{x\in\mathcal{F}^*}$ defined as the negative top $\delta - 1$ coefficients of P such that $\forall x \in F^*, P(x) = y_x$ is equal to the syndrome defined following Definition 6 as $\sum_{x\in\mathcal{F}} y_x x^i$ for $i = 1, 2, \ldots, \delta - 1$.[12] Thus, when $n = |\mathcal{F}| - 1$, the codewords obtained via the evaluation-based definition are *identical* to the codewords obtain via Definition 6, because codewords are simply elements with the zero syndrome, and the syndrome maps agree.

This is an example of a remarkable duality between evaluations of polynomials and their coefficients: the syndrome can be viewed either as the evaluation of a polynomial whose coefficients are given by the vector, or as the coefficients of the polynomial whose evaluations are given by a vector.

The syndrome decoding algorithm above has a natural interpretation in the evaluation-based view. Our presentation is an adaptation of Welch-Berlekamp decoding as presented in, e.g., [Sud01, Chapter 10].

Suppose $n = |F| - 1$ and $x_1, ..., x_n$ are the non-zero elements of the field. Let $y = (y_1, y_2, \ldots, y_n)$ be a vector. We are given its syndrome $\mathsf{syn}(y) = (-p_{n-1}, -p_{n-2}, \ldots, -p_{n-(\delta-1)})$, where $p_{n-1}, \ldots, p_{n-(\delta-1)}$ are the top coefficients of the interpolating polynomial P. Knowing only $\mathsf{syn}(y)$, we need to find at most $(\delta - 1)/2$ locations x_i such that correcting all the corresponding y_i will result in a codeword. Suppose that codeword is given by a degree-$(n-\delta)$ polynomial c. Note that c agrees with P on all but the error locations. Let $\rho(z)$ be the polynomial of degree at most $(\delta - 1)/2$ whose roots are exactly the error locations. (Note that $\sigma(z)$ from the decoding algorithm above is the same $\rho(z)$ but with coefficients in reverse order, because the roots of σ are the inverses of the roots of ρ.) Then $\rho(z) \cdot P(z) = \rho(z) \cdot c(z)$ for $z = x_1, x_2, \ldots, x_n$. Since $x_1, .., x_n$ are all the nonzero field elements, $\prod_{i=1}^n (z - x_i) = z^n - 1$. Thus,

$$\rho(z) \cdot c(z) \quad = \quad \rho(z) \cdot P(z) \bmod \prod_{i=1}^{n} (z - x_i) \quad = \quad \rho(z) \cdot P(z) \bmod (z^n - 1)\,.$$

If we write the left-hand side as $\alpha_{n-1}x^{n-1} + \alpha_{n-2}x^{n-2} + \cdots + \alpha_0$, then the above equation implies that $\alpha_{n-1} = \cdots = \alpha_{n-(\delta-1)/2} = 0$ (because the degree if $\rho(z) \cdot c(z)$ is at most $n - (\delta + 1)/2$). Because $\alpha_{n-1}, \ldots, \alpha_{n-(\delta-1)/2}$ depend on the coefficients of ρ as well as on $p_{n-1}, \ldots, p_{n-(\delta-1)}$, but not on lower coefficients of P, we obtain a system of $(\delta - 1)/2$ equations for $(\delta - 1)/2$ unknown coefficients of ρ. A careful examination shows that it is essentially the same system as we had for $\sigma(z)$ in the algorithm above. The lowest-degree solution to this system is indeed the correct ρ, by the same argument which was used

[12] This statement can be shown as follows: because both maps are linear, it is sufficient to prove that they agree on a vector $(y_x)_{x\in\mathcal{F}^*}$ such that $y_a = 1$ for some $a \in \mathcal{F}^*$ and $y_x = 0$ for $x \neq a$. For such a vector, $\sum_{x\in\mathcal{F}} y_x x^i = a^i$. On the other hand, the interpolating polynomial $P(x)$ such that $P(x) = y_x$ is $-ax^{n-1} - a^2x^{n-2} - \cdots - a^{n-1}x - 1$ (indeed, $P(a) = -n = 1$; furthermore, multiplying $P(x)$ by $x - a$ gives $a(x^n - 1)$, which is zero on all of $\mathcal{F}^*$; hence $P(x)$ is zero for every $x \neq a$).

to prove the correctness of σ in Lemma E.1. The roots of ρ are the error-locations. For $q > 2$, the actual corrections that are needed at the error locations (in other words, the light vector corresponding to the given syndrome) can then be recovered by solving the linear system of equations implied by the value of the syndrome.