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

Yevgeniy Dodis[1], Leonid Reyzin[2], and Adam Smith[3]

[1] New York University, dodis@cs.nyu.edu
[2] Boston University, reyzin@cs.bu.edu
[3] MIT, asmith@csail.mit.edu

Abstract. We provide formal definitions and efficient secure techniques for

– turning biometric information into keys usable for *any* cryptographic application, and
– 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* extracts nearly uniform randomness R from its biometric 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 any cryptographic application. A *secure sketch* produces public information about its biometric 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.

In addition to formally introducing our new primitives, 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.

1 Introduction

Cryptography traditionally relies on uniformly distributed 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 [12] or a list of favorite movies [16]) 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). 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 [11] 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 "biometric" data.

Our Definitions. We propose two primitives, termed *secure sketch* and *fuzzy extractor*.

A secure sketch addresses the problem of error tolerance. It is a (probabilistic) function outputting a public value v about its biometric input w, that, while revealing little about w, allows its exact reconstruction from any other input w' that is sufficiently close. The price for this error tolerance is that the application will have to work with a lower level of entropy of the input, since publishing v effectively reduces the entropy of w. However, in a good secure sketch, this reduction will be small, and w will still have enough entropy to be useful, even if the adversary knows v. A secure sketch, however, does not address nonuniformity of inputs.

A fuzzy extractor addresses both error tolerance and nonuniformity. It reliably extracts a uniformly random string R from its biometric input w in an error-tolerant way. If the input changes but remains close, the extracted R remains the same. To assist in recovering R from w', a fuzzy extractor outputs a public string P (much like a secure sketch outputs v to assist in recovering w). However, R remains uniformly random even given P.

Our approach is general: our primitives can be naturally combined with *any* cryptographic system. Indeed, R extracted from w by a fuzzy extractor can be used as a key in any 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 our primitives to be *information-theoretically* secure, thus allowing them to be used in combination with any cryptographic system without additional assumptions (however, 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 $\langle P, f(R)\rangle$. When the user inputs w' close to w, the server recovers the actual R 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. Secure sketches and extractors can thus 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 (v or P). In particular, fuzzy extractors can be viewed as error- and nonuniformity-tolerant secret key *key-encapsulation mechanisms* [27].

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 from some metric space, and that w' is no more that a certain distance from w in that space. We only consider particular metrics 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 [24], such as, for example, pairwise-independent hash functions. We also demonstrate that the existence of secure sketches and fuzzy extractors over a particular metric space implies the existence of certain error-correcting codes in that space, thus producing lower bounds on the best parameters a secure fingerprint and fuzzy extractor can achieve. 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 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 [15] 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') comes up naturally whenever the biometric input is represented as a subset of features from a universe of possible features.[4] We demonstrate the existence of optimal (with respect to entropy loss) secure sketches (and therefore also 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, and provide two of them.

First, we observe that the "fuzzy vault" construction of Juels and Sudan [16] 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 v. Our bound on the loss is quite high unless one makes the size of the output v very large. We then provide an improvement to the Juels-Sudan construction to reduce the entropy loss to near optimal, while keeping v short (essentially as long as w).

Second, 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. However, the length of such a vector becomes unmanageable as the universe size grows. Nonetheless, we demonstrate that this approach can be made to work efficiently even for exponentially large universes. This involves a result that may be of independent interest: we show 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). The resulting secure sketch scheme compares favorably to the modified Juels-Sudan construction: it has the same near-optimal entropy loss, while the public output v is even shorter (proportional to the number of errors tolerated, rather than the input length).

Finally, edit distance (i.e., the number of insertions and deletions needed to convert one string into the other) naturally comes up, for example, when the password is entered as a string, due to typing errors or mistakes made in handwriting recognition. We construct a biometric embedding from the edit metric into the set difference metric, and then apply our general result to show such an

[4] A perhaps unexpected application of the set difference metric was explored in [16]: 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.

embedding yields a fuzzy extractor for edit distance, because we already have fuzzy extractors for set difference. We note that the edit metric is quite difficult to work with, and the existence of such an embedding is not a priori obvious: for example, low-distortion embeddings of the edit distance into the Hamming distance are unknown and seem hard [2]. It is the particular properties of biometric embeddings, as we define them, that help us construct this embedding.

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, Ellison et al. [10] propose asking the user a series of n personalized questions, and use these answers to encrypt the "actual" truly random secret R. A similar approach using user's keyboard dynamics (and, subsequently, voice [21, 22]) was proposed by Monrose et al [20]. Of course, this technique reduces the question to that of designing a secure "fuzzy encryption". While heuristic approaches were suggested in the above works (using various forms of Shamir's secret sharing), no formal analysis was given. Additionally, error tolerance was addressed only by brute force search.

A formal approach to error tolerance in biometrics was taken by Juels and Wattenberg [15] (for less formal solutions, see [8, 20, 10]), who provided a simple way to tolerate errors in *uniformly distributed* passwords. Frykholm and Juels [12] extended this solution; our analysis is quite similar to theirs in the Hamming distance case. Almost the same construction appeared implicitly in earlier, seemingly unrelated, literature on information reconciliation and privacy amplification (see, e.g., [3, 4, 7]). We discuss the connections between these works and our work further in Section 4.

Juels and Sudan [16] provided the first construction for a metric other than Hamming: they construct a "fuzzy vault" scheme for the set difference metric. The main difference is that [16] 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 notion can be viewed as an (information-theoretically) one-way function, rather than a semantically-secure key encapsulation mechanism, like the one considered in this work. Nonetheless, their informal notion is very closely related to our secure sketches, and we improve their construction in Section 5.

Linnartz and Tuyls [18] define and construct a primitive very similar to a fuzzy extractor (that line of work was continued in [28].) The definition of [18] 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 [18]. However, our constructions focus on discrete metric spaces.

Work on privacy amplification [3, 4], as well as work on de-randomization and hardness amplification [14, 24], also addressed the need to extract uniform randomness from a random variable about which some information has been leaked. A major focus of research in that literature has been the development of (ordinary, not fuzzy) extractors with short seeds (see [26] for a survey). We use extractors in this work (though for our purposes, pairwise independent hashing [3, 14] is sufficient). Conversely, our work has been applied recently to privacy amplification: Ding [9] uses fuzzy extractors for noise tolerance in Maurer's bounded storage model.

EXTENSIONS. We can relax the error correction properties of sketches and fuzzy extractors to allow *list decoding*: instead of outputting one correct secret, we can output a short list of secrets, one of which is correct. For many applications (e.g., password authentication), this is sufficient, while the advantage is that we can possibly tolerate many more errors in the password. Not surprisingly, by using list-decodable codes (see [13] and the references therein) in our constructions, we can achieve this relaxation and considerably improve our error tolerance. Other similar extensions would be to allow small error probability in error-correction, to ensure correction of only *average-case* errors, or to consider nonbinary alphabets. Again, many of our results will extend to these settings. Finally, an interesting new direction is to consider other metrics not considered in this work.

2 Preliminaries

Unless explicitly stated otherwise, all logarithms below are base 2. We use U_ℓ to denote the uniform distribution on ℓ-bit binary strings.

ENTROPY. The *min-entropy* $\mathbf{H}_\infty(A)$ of a random variable A is $-\log(\max_a \Pr(A = a))$. For a pair of (possibly correlated) random variables A, B, a conventional notion of "average min-entropy" of A given B would be $\mathbb{E}_{b \leftarrow B}\left[\mathbf{H}_\infty(A \mid B = b)\right]$. However, for the purposes of this paper, the following slightly modified notion will be more robust: we let $\tilde{\mathbf{H}}_\infty(A \mid B) = -\log\left(\mathbb{E}_{b \leftarrow B}\left[2^{-\mathbf{H}_\infty(A|B=b)}\right]\right)$. Namely, we define *average min-entropy* of A given B to be the logarithm of the average probability of the most likely value of A given B. One can easily verify that if B is an ℓ-bit string, then $\tilde{\mathbf{H}}_\infty(A \mid B) \geq \mathbf{H}_\infty(A) - \ell$.

STRONG EXTRACTORS. The *statistical distance between* two probability distributions A and B is $\mathbf{SD}(A, B) = \frac{1}{2}\sum_v |\Pr(A = v) - \Pr(B = v)|$. We can now define *strong randomness extractors* [24].

Definition 1. *An efficient* (n, m', ℓ, ϵ)-strong extractor *is a polynomial time probabilistic function* $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^\ell$ *such that for all min-entropy* m' *distributions* W*, we have* $\mathbf{SD}(\langle \mathsf{Ext}(W; X), X\rangle, \langle U_\ell, X\rangle) \leq \epsilon$*, where* $\mathsf{Ext}(W; X)$ *stands for applying* Ext *to* W *using (uniformly distributed) randomness* X*.*

Strong extractors can extract at most $\ell = m' - 2\log(1/\epsilon) + O(1)$ nearly random bits [25]. Many constructions match this bound (see Shaltiels' survey [26] 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 2-wise independent hash functions will already give us optimal $\ell = m' - 2\log(1/\epsilon)$ [3, 14].

Metric Spaces. A metric space is a set $\mathcal{M}$ with a distance function $\mathsf{dis} : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^+ = [0, \infty)$ which obeys various natural properties. In this work, $\mathcal{M}$ will always be a finite set, and the distance function will only take on integer values. The size of the $\mathcal{M}$ will always be denoted $N = |\mathcal{M}|$. We will assume that any point in $\mathcal{M}$ can be naturally represented as a binary string of appropriate length $O(\log N)$.

We will concentrate on the following metrics. (1) *Hamming metric.* Here $\mathcal{M} = \mathcal{F}^n$ over some alphabet $\mathcal{F}$ (we will mainly use $\mathcal{F} = \{0, 1\}$), and $\mathsf{dis}(w, w')$ is the number of positions in which they differ. (2) *Set Difference metric.* Here $\mathcal{M}$ consists of all s-element subsets in a universe $\mathcal{U} = [n] = \{1, ..., n\}$. The distance between two sets A, B is the number of points in A that are not in B. Since A and B have the same size, the distance is half of the size of their symmetric difference: $\mathsf{dis}(A, B) = \frac{1}{2}|A \triangle B|$. (3) *Edit metric.* Here again $\mathcal{M} = \mathcal{F}^n$, but the distance between w and w' is defined to be one half of the smallest number of character insertions and deletions needed to transform w into w'.

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

Coding. Since we want to achieve error tolerance in various metric spaces, we will use *error-correcting codes* in the corresponding metric space $\mathcal{M}$. A code C is a subset $\{w_1, \ldots, w_K\}$ of K elements of $\mathcal{M}$ (for efficiency purposes, we want the map from i to w_i to be polynomial-time). The *minimum 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 any codeword. 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 w_i in the ball of radius t around w: $\mathsf{dis}(w, w_i) \leq t$ for at most one i. Clearly, for integer metrics we have $t = \lfloor (d-1)/2 \rfloor$. Since error correction will be more important in our applications, we denote the corresponding codes by $(\mathcal{M}, K, t)$-codes. For the Hamming and the edit metrics on strings of length n over some alphabet $\mathcal{F}$, we will sometimes call $k = \log_{|F|} K$ the *dimension* on the code, and denote the code itself as an $[n, k, d = 2t + 1]$-code, following the standard notation in the literature.

3 Definitions and General Lemmas

Let $\mathcal{M}$ be a metric space on N points with distance function dis.

Definition 2. *An* $(\mathcal{M}, m, m', t)$-secure sketch *is a randomized map* $\mathsf{SS} : \mathcal{M} \to \{0, 1\}^*$ *with the following properties.*

1. *There exists a deterministic recovery function* Rec *allowing to recover* w *from its sketch* $\mathsf{SS}(w)$ *and any vector* w' *close to* w*: for all* $w, w' \in \mathcal{M}$ *satisfying* $\mathsf{dis}(w, w') \leq t$*, we have* $\mathsf{Rec}(w', \mathsf{SS}(w)) = w$.
2. *For all random variables* W *over* $\mathcal{M}$ *with min-entropy* m*, the average min-entropy of* W *given* $\mathsf{SS}(W)$ *is at least* m'*. That is,* $\tilde{\mathbf{H}}_\infty(W \mid \mathsf{SS}(W)) \geq m'$.

The secure sketch is efficient *if* SS *and* Rec *run in time polynomial in the representation size of a point in* $\mathcal{M}$*. We denote the random output of* SS *by* $\mathsf{SS}(W)$*, or by* $\mathsf{SS}(W;X)$ *when we wish to make the randomness explicit.*

We will have several examples of secure sketches when we discuss specific metrics. The quantity $m - m'$ is called the *entropy loss* of a secure sketch. Our proofs in fact bound $m - m'$, and the same bound holds for all values of m.

Definition 3. *An* $(\mathcal{M}, m, \ell, t, \epsilon)$ fuzzy extractor *is a given by two procedures* $(\mathsf{Gen}, \mathsf{Rep})$.

1. Gen *is a probabilistic generation procedure, which on input* $w \in \mathcal{M}$ *outputs an "extracted" string* $R \in \{0,1\}^\ell$ *and a public string* P*. We require that for any distribution* W *on* $\mathcal{M}$ *of min-entropy* m*, if* $\langle R, P\rangle \leftarrow \mathsf{Gen}(W)$*, then we have* $\mathbf{SD}\left(\langle R, P\rangle, \langle U_\ell, P\rangle\right) \leq \epsilon$.
2. Rep *is a deterministic reproduction procedure allowing to recover* R *from the corresponding public string* P *and any vector* w' *close to* w*: for all* $w, w' \in \mathcal{M}$ *satisfying* $\mathsf{dis}(w, w') \leq t$*, if* $\langle R, P\rangle \leftarrow \mathsf{Gen}(w)$*, then we have* $\mathsf{Rep}(w', P) = R$.

The fuzzy extractor is efficient *if* Gen *and* Rep *run in time polynomial in the representation size of a point in* $\mathcal{M}$.

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 public string P produced during the initial extraction; yet 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 $\mathcal{M} = \{0,1\}^n$).

Construction of Fuzzy Extractors from Secure Sketches. Not surprisingly, secure sketches come up very handy in constructing fuzzy extractors. Specifically, we construct fuzzy extractors from secure sketches and strong extractors. For that, we assume that one can naturally represent a point w in $\mathcal{M}$ using n bits. The strong extractor we use is the standard pairwise-independent hashing construction, which has (optimal) entropy loss $2\log\left(\frac{1}{\epsilon}\right)$. The proof of the following lemma uses the "left-over hash" (a.k.a. "privacy amplification") lemma of [14, 4], and can be found in the full version of our paper.

Lemma 1 (Fuzzy Extractors from Sketches). *Assume* SS *is a* $(\mathcal{M}, m, m', t)$*-secure sketch with recovery procedure* Rec*, and let* Ext *be the* (n, m', ℓ, ϵ)*-strong extractor based on pairwise-independent hashing (in particular,* $\ell = m' - 2\log\left(\frac{1}{\epsilon}\right)$*). Then the following* $(\mathsf{Gen}, \mathsf{Rep})$ *is a* $(\mathcal{M}, m, \ell, t, \epsilon)$*-fuzzy extractor:*

- $\mathsf{Gen}(W; X_1, X_2)$*: set* $P = \langle \mathsf{SS}(W; X_1), X_2\rangle$, $R = \mathsf{Ext}(W; X_2)$*, output* $\langle R, P\rangle$.
- $\mathsf{Rep}(W', \langle V, X_2\rangle)$*: recover* $W = \mathsf{Rec}(W', V)$ *and output* $R = \mathsf{Ext}(W; X_2)$.

Remark 1. One can prove an analogous form of Lemma 1 using any strong extractor. However, in general, the resulting reduction leads to fuzzy extractors with min-entropy loss $3\log\left(\frac{1}{\epsilon}\right)$ instead of $2\log\left(\frac{1}{\epsilon}\right)$. This may happen in the case when the extractor does not have a convex tradeoff between the input entropy and the distance from uniform of the output. Then one can instead use a high-probability bound on the min-entropy of the input (that is, if $\tilde{\mathbf{H}}_\infty(X|Y) \geq m'$ then the event $\mathbf{H}_\infty(X|Y=y) \geq m' - \log\left(\frac{1}{\epsilon}\right)$ happens with probability $1-\epsilon$).

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 $\mathcal{M}$ *transtive*). For example, in the Hamming space, the set of all shifts $\pi_x(w) = w \oplus x$ is such a family (see Section 4 for more details on this example).

Let C be an $(\mathcal{M}, K, t)$-code. Then the general sketching scheme is the following: given a input $w \in \mathcal{M}$, pick a random codeword $b \in C$, pick a random permutation $\pi \in \Pi$ such that $\pi(w) = b$, and output $\mathsf{SS}(w) = \pi$. To recover w given w' and the sketch π, find the closest codeword b' to $\pi(w')$, and output $\pi^{-1}(b')$. This works when $\mathsf{dis}((,w),w') \leq t$, because then $\mathsf{dis}((,b),\pi(w')) \leq t$, so decoding $\pi(w')$ will result in $b' = b$, which in turn means that $\pi^{-1}(b') = w$.

A bound on the entropy loss of this scheme, which follows simply from "counting" entropies, is $|``\pi"| - \log K$, where $|``\pi"|$ is the size, in bits, of a canonical description of π. (We omit the proof, as it is a simple generalization of the proof of Lemma 3.) Clearly, this quantity will be small if the family Π of transifitive isometries is small and the code C is dense. (For the scheme to be usable, we also need the operations on the code, as well as π and π^{-1}, to be implementable reasonably efficiently.)

Constructions from Biometric Embeddings. We now introduce a general technique that allows one to build good fuzzy extractors in some metric space $\mathcal{M}_1$ from good fuzzy extractors 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:*

- $\forall\, w_1, w_1' \in \mathcal{M}_1$ *such that* $\mathsf{dis}(w_1, w_1')_1 \leq t_1$, *we have* $\mathsf{dis}(f(w_1), f(w_2))_2 \leq t_2$.
- $\forall\, W_1$ *on* $\mathcal{M}_1$ *such that* $\mathbf{H}_\infty(W_1) \geq m_1$, *we have* $\mathbf{H}_\infty(f(W_1)) \geq m_2$.

The following lemma is immediate:

Lemma 2. *If f is (t_1, t_2, m_1, m_2)-biometric embedding of $\mathcal{M}_1$ into $\mathcal{M}_2$ and $(\mathsf{Gen}_1(\cdot), \mathsf{Rep}_1(\cdot,\cdot))$ is a $(\mathcal{M}_2, m_2, \ell, t_2, \epsilon)$-fuzzy extractor, then $(\mathsf{Gen}_1(f(\cdot)), \mathsf{Rep}_1(f(\cdot),\cdot))$ is a $(\mathcal{M}_1, m_1, \ell, t_1, \epsilon)$-fuzzy extractor.*

Notice that a similar result does not hold for secure sketches, unless f is injective (and efficiently invertible).

We will see the utility of this particular notion of embedding (as opposed to previously defined notions) in Section 6.

4 Constructions for Hamming Distance

In this section we consider constructions for the space $\mathcal{M} = \{0,1\}^n$ under the Hamming distance metric.

THE CODE-OFFSET CONSTRUCTION. Juels and Wattenberg [15] considered a notion of "fuzzy commitment." [5] Given a binary $[n, k, 2t+1]$ 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 random $X \leftarrow \{0,1\}^k$, set

$$\mathsf{SS}(W; X) = W \oplus C(X)\,.$$

(Note that if W is uniform, this secure sketch direcly yields a fuzzy extractor with $R = X$).

When the code C is linear, this is equivalent to revealing the syndrome of the input w, and so we do not need the randomness X. Namely, in this case we could have set $\mathsf{SS}(w) = \mathsf{syn}_C(w)$ (as mentioned in the introduction, this construction also appears implicitly in the information reconciliation literature, e.g. [3, 4, 7]: when Alice and Bob hold secret values which are very close in Hamming distance, one way to correct the differences with few bits of communication is for Alice to send to Bob the *syndrome* of her word w with respect to a good linear code.)

Since the syndrome of a k-dimensional linear code is $n-k$ bits long, it is clear that $\mathsf{SS}(w)$ leaks only $n-k$ bits about w. In fact, we show the same is true even for nonlinear codes.

Lemma 3. *For any $[n, k, 2t+1]$ code C and any m, SS above is a $(\mathcal{M}, m, m+k-n, t)$ secure sketch. It is efficient if the code C allows decoding errors in polynomial time.*

Proof. Let D be the decoding procedure of our code C. Since D can correct up to t errors, if $v = w \oplus C(x)$ and $\mathsf{dis}(w, w') \le t$, then $D(w' \oplus v) = x$. Thus, we can set $\mathsf{Rec}(w', v) = v \oplus C(D(w' \oplus v))$.

Let A be the joint variable (X, W). Together, these have min-entropy $m+k$ when $\mathbf{H}_\infty(W) = m$. Since $\mathsf{SS}(W) \in \{0,1\}^n$, we have $\tilde{\mathbf{H}}_\infty(W, X \mid \mathsf{SS}(W)) \ge m+k-n$. Now given $\mathsf{SS}(W)$, W and X determine each other uniquely, and so $\tilde{\mathbf{H}}_\infty(W \mid \mathsf{SS}(W)) \ge m+k-n$ as well. □

In the full version, we present some generic lower bounds on secure sketches and extractors. Let $A(n, d)$ denote the maximum number of codewords possible

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

in a code of distance d in $\{0,1\}^n$. Then the entropy loss of a secure sketch for the Hamming metric is at least $n - \log A(n, 2t+1)$, when the input is uniform (that is, when $m = n$). This means that the code-offset construction above is optimal for the case of uniform inputs. Of course, we do not know the exact value of $A(n, d)$, never mind of efficiently decodable codes which meet the bound, for most settings of n and d. Nonetheless, the code-offset scheme gets as close to optimality as is possible in coding.

Getting Fuzzy Extractors. As a warm-up, consider the case when W is uniform ($m = n$) and look at the code-offset sketch construction: $V = W \oplus C(X)$. Setting $R = X$, $P = V$ and $\mathsf{Rep}(W', V) = D(V \oplus W')$, we clearly get an $(\mathcal{M}, n, k, t, 0)$ fuzzy extractor, since V is truly random when W is random, and therefore independent of X. In fact, this is exactly the usage proposed by Juels-Wattenberg, 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. However, by using the construction in Lemma 1, we get

Lemma 4. *Given any $[n, k, 2t+1]$ code C and any m, ϵ, we can get an $(\mathcal{M}, m, \ell, t, \epsilon)$ fuzzy extractor, where $\ell = m+k-n-2\log(1/\epsilon)$. The recovery* Rep *is efficient if C allows decoding errors in polynomial time.*

5 Constructions for Set Difference

Consider the collection of all sets of a particular size s in a universe $\mathcal{U} = [n] = \{1, ..., n\}$. The distance between two sets A, B is the number of points in A that are not in B. Since A and B have the same size, the distance is half of the size of their symmetric difference: $\frac{1}{2}\mathsf{dis}(A, B) = |A \triangle B|$. If A and B are viewed as n-bit characteristic vectors over $[n]$, this metric is the same as the Hamming metric (scaled by $1/2$). Thus, the set difference metric can be viewed as a restriction of the binary Hamming metric to all the strings with exactly s nonzero components. However, one typically assumes that n is much larger than s, so that representing a set by n bits is much less efficient than, say writing down a list of elements, which requires $(s \log n)$ bits.

Large Versus Small Universes. 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. By 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. Thus, the large universe setting is exactly the setting in which the n-bit string representation of a set becomes too large to be usable. We consider the small-universe setting first, since it appears simpler (Section 5.1). The remaining subsections consider large universes.

5.1 Small Universes

When the universe size is polynomial in s, there are a number of natural constructions. Perhaps the most direct one, given previous work, is the construction of Juels and Sudan [16]. Unfortunately, that scheme achieves relatively poor parameters (see Section 5.2). We suggest two possible constructions. The first one represents sets as n-bit strings and uses the constructions of the previous section (with the caveat that Hamming distance is off by a factor of 2 from set difference).

The second construction goes directly through codes for set difference, also called "constant-weight" codes. A constant-weight code is a ordinary error-correcting code in $\{0,1\}^n$ in which all of the codewords have the same Hamming weight s. The set difference metric is transitive—the metric is invariant under permutations of the underlying universe $\mathcal{U}$, and for any two sets of the same size $A, B \subseteq \mathcal{U}$, there is a permutation of $\mathcal{U}$ that maps A to B. Thus, one can use the general scheme for secure sketches in transitive metrics (Section 3) to get a secure sketch for set difference with output length about $n \log n$.

The full version of the paper contains a more detailed comparison of the two constructions. Briefly: The second construction achieves better parameters since, according to currently proved bounds, it seems that constant-weight codes can be more dense than ordinary codes. On the other hand, explicit codes which highlight this difference are not known, and much more is known about efficient implementations of decoding for ordinary codes. In practice, the Hamming-based scheme is likely to be more useful.

5.2 Modifying the Construction of Juels and Sudan

We now turn to the large universe setting, where n is super-polynomial in s. Juels and Sudan [16] proposed a secure sketch for the set difference metric (called a "fuzzy vault" in that paper). They assume for simplicity that $n = |\mathcal{U}|$ is a prime power and work over the field $\mathcal{F} = GF(n)$. On input set A, the sketch they produce is a set of r pairs of points (x_i, y_i) in $\mathcal{F}$, with $s < r \leq n$. Of the x_i values, s are the elements of A, and their corresponding y_i value are evaluations of a random degree-$(s-2t-1)$ polynomial p at x_i; the remaining $r-s$ of the (x_i, y_i) values are chosen at random but not on p. The original analysis [16] does not extend to the case of a nonuniform password in a large universe. However, we give a simpler analysis which does cover that range of parameters. Their actual scheme, as well as our new analysis, can be found in the full version of the paper. We summarize here:

Lemma 5. *The entropy loss of the Juels-Sudan scheme is at most* $m - m' = 2t \log n + \log \binom{n}{r} - \log \binom{n-s}{r-s}$.

Their scheme requires storage $2r \log n$. 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 $2t \log n - \log \binom{r}{s} + \log \binom{n}{s}$, using the identity

$\binom{n}{r}\binom{r}{s} = \binom{n}{s}\binom{n-s}{r-s}$. Now the entropy of A is at most $\binom{n}{s}$, and so our lower bound on the remaining entropy is $(\log\binom{r}{s} - 2t\log n)$. To make this quantity large requires making r very large.

MODIFIED JS SKETCHES. We suggest a modification of the Juels-Sudan scheme with entropy loss at most $2t\log n$ and storage $s\log n$. Our scheme has the advantage of being even simpler to analyze. As before, we assume n 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) using only the coefficients of p'.

Algorithm 1 (Modified JS Secure Sketch). Input: a set $A \subseteq \mathcal{U}$.

1. Choose $p()$ at random from the set of polynomials of degree at most $k = s - 2t - 1$ over $\mathcal{F}$.
2. Let $p'()$ be the unique monic polynomial of degree exactly s such that $p'(x) = p(x)$ for all $x \in A$.
   (Write $p'(x) = x^s + \sum_{i=0}^{s-1} a_i x^i$. Solve for $a_0, ..., a_{s-1}$ using the s linear constraints $p'(x) = p(x)$, $x \in A$.)
3. Output the list of coefficients of $p'()$, that is $\mathsf{SS}(A) = (a_0, ..., a_{s-1})$.

First, observe that solving for $p'()$ in Step 2 is always possible, since the s constraints $\sum_{i=0}^{s-1} a_i x^i = p(x) - x^s$ are in fact linearly independent (this is just polynomial interpolation).

Second, this sketch scheme can tolerate t set difference errors. Suppose we are given a set $B \subseteq \mathcal{U}$ which agrees with A in at least $s - t$ positions. Given $p' = \mathsf{SS}(A)$, one can evaluate p' on all the points in the set B. The resulting vector agrees with p on at least $s-t$ positions, and using the decoding algorithm for Reed-Solomon codes, one can thus reconstruct p exactly (since $k = s-2t-1$). Finally, the set A can be recovered by finding the roots of the polynomial $p' - p$: since $p' - p$ is not identically zero and has degree exactly s, it can have at most s roots and so $p' - p$ is zero only on A.

We now turn to the entropy loss of the scheme. The sketching scheme invests $(s - 2t)\log n$ bits of randomness to choose the polynomial p. The number of possible outputs p' is n^s. If X is the invested randomness, then the (average) min-entropy (A, X) given $\mathsf{SS}(A)$ is at least $\tilde{\mathbf{H}}_\infty(A) - 2t\log n$. The randomness X can be recovered from A and $\mathsf{SS}(A)$, and so we have $\tilde{\mathbf{H}}_\infty(A \mid \mathsf{SS}(A)) \geq \tilde{\mathbf{H}}_\infty(A) - 2t\log n$. We have proved:

Lemma 6 (Analysis of Modified JS). *The entropy loss of the modified JS scheme is at most $2t\log n$. The scheme has storage $(s+1)\log n$ for sets of size s in $[n]$, and both the sketch generation $\mathsf{SS}()$ and the recovery procedure $\mathsf{Rec}()$ run in polynomial time.*

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 $2t\log n$ is essentially optimal for the uniform case $m = \log \binom{n}{s}$. Our lower bound (in the full version) shows that for a uniformly distributed input, the best possible entropy loss is $m - m' \geq \log \binom{n}{s} - \log A(n, s, 4t+1)$, where $A(n, s, d)$ is the maximum size of a code of constant weight s and minimum Hamming distance d. Using a bound of Agrell *et al* ([1], Theorem 12), the entropy loss is at least:

$$m - m' \geq \log \binom{n}{s} - \log A(n, s, 4t+1) \geq \log \binom{n - s + 2t}{2t}$$

When $n \geq s$, this last quantity is roughly $2t \log n$, as desired.

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

In this section, we show that code-offset construction 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.

Syndromes of Linear Codes. For a $[n, k, d]$ linear code C with parity check matrix H, recall that the syndrome of a word $w \in \{0, 1\}^n$ is $\mathsf{syn}(w) = Hw$. The syndrome has length $n - k$, and the code is exactly the set of words c such that $\mathsf{syn}(c) = 0^{n-k}$. The syndrome captures all the information necessary for decoding. That is, suppose a codeword c is sent through a channel and the word $w = c \oplus e$ is received. First, the syndrome of w is the syndrome of e: $\mathsf{syn}(w) = \mathsf{syn}(c) \oplus \mathsf{syn}(e) = 0 \oplus \mathsf{syn}(e) = \mathsf{syn}(e)$. Moreover, for any value u, there is at most one word e of weight less than $d/2$ such that $\mathsf{syn}(e) = u$ (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). Thus, knowing syndrome $\mathsf{syn}(w)$ is enough to determine the error pattern e if not too many errors occurred.

As mentioned before, we can reformulate the code-offset construction in terms of syndrome: $\mathsf{SS}(w) = \mathsf{syn}(w)$. The two schemes are equivalent: given $\mathsf{syn}(w)$ one can sample from $w \oplus C(X)$ by choosing a random string v with $\mathsf{syn}(v) = \mathsf{syn}(w)$; conversely, $\mathsf{syn}(w \oplus C(X)) = \mathsf{syn}(w)$. This reformulation gives us no special advantage when the universe is small: storing $w + C(X)$ is not a problem. However, it's a substantial improvement when $n \gg n - k$.

Syndrome Manipulation for Small-Weight Words. Suppose now that we have a small set $A \subseteq [n]$ of size s, where $n \gg s$. Let $x_A \in \{0, 1\}^n$ denote the characteristic vector of A. If we want to use $\mathsf{syn}(x_A)$ as the sketch of A, then we must choose a code with $n - k \leq \log \binom{n}{s} \approx s \log n$, since the sketch has entropy loss $(n - k)$ and the maximum entropy of A is $\log \binom{n}{s}$.

Binary BCH codes are a family of $[n, k, d]$ linear codes with $d = 4t + 1$ and $k = n - 2t \log n$ (assuming $n + 1$ is a power of 2) (see, e.g. [19]). These codes are optimal for $t \ll n$ by the Hamming bound, which implies that $k \leq n - \log \binom{n}{2t}$ [19]. Using the code-offset sketch with a BCH code C, we get entropy loss $n - k = 2t \log n$, just as we did for the modified Juels-Sudan scheme (recall that $d \geq 4t+1$ 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 $\mathsf{syn}(x_A) = Hx_A$ and, later, run a decoding algorithm to recover x_A. For BCH codes, this difficulty can be overcome. A word of small weight x can be described by listing the positions on which it is nonzero. We call this description the *support* of x and write $\mathsf{supp}(x)$ (that is $\mathsf{supp}(x_A) = A$)).

Lemma 7. *For a $[n, k, d]$ binary BCH code C one can compute:*

1. *$\mathsf{syn}(x)$, given $\mathsf{supp}(x)$, and*
2. *$\mathsf{supp}(x)$, given $\mathsf{syn}(x)$ (when x has weight at most $(d-1)/2$),*

in time polynomial in $|\mathsf{supp}(x)| = weight(x) \cdot \log(n)$ and $|\mathsf{syn}(x)| = n - k$.

The proof of Lemma 7 mainly requires a careful reworking of the standard BCH decoding algorithm. The details are presented in the full version of the paper. For now, we present the resulting sketching scheme for set difference. The algorithm works in the field $GF(2^m) = GF(n+1)$, and assumes a generator α for $GF(2^m)$ has been chosen ahead of time.

Algorithm 2 (BCH-based Secure Sketch). Input: a set $A \in [n]$ of size s, where $n = 2^m - 1$. (Here α is a generator for $GF(2^m)$, fixed ahead of time.)

1. Let $p(x) = \sum_{i \in A} x^i$.
2. Output $\mathsf{SS}(A) = (p(\alpha), p(\alpha^3), p(\alpha^5), ..., p(\alpha^{4t+1}))$ (computations in $GF(2^m)$).

Lemma 7 yields the algorithm $\mathsf{Rec}()$ which recovers A from $\mathsf{SS}(A)$ and any set which intersects A in at least $s - t$ points. However, the bound on entropy loss is easy to see: the output is $2t \log n$ bits long, and hence the entropy loss is at most $2t \log n$. We obtain:

Theorem 1. *The BCH scheme above is a $[m, m-2t \log n, t]$ secure sketch scheme for set difference with storage $2t \log n$. The algorithms SS and Rec both run in polynomial time.*

6 Constructions for Edit Distance

First we note that simply applying the same approach as we took for the transitive metric spaces before (the Hamming space and the set difference space for small universe sizes) does not work here, because the edit metric does not seem to be transitive. Indeed, it is unclear how to build a permutation π such that for any w' close to w, we also have $\pi(w')$ close to $x = \pi(w)$. For example, setting $\pi(y) = y \oplus (x \oplus w)$ is easily seen not to work with insertions and deletions. Similarly, if I is some sequence of insertions and deletions mapping w to x, it is not true that applying I to w' (which is close to w) will necessarily result in some x' close to x. In fact, then we could even get $\mathsf{dis}(w', x') = 2\mathsf{dis}(w, x) + \mathsf{dis}(w, w')$.

Perhaps one could try to simply embed the edit metric into the Hamming metric using known embeddings, such as conventionally used low-distorion embeddings, which provide that all distances are preserved up to some small "distortion" factor. However, there are no known nontrivial low-distortion embeddings from the edit metric to the Hamming metric. Moreover, it was recently proved by Andoni et al [2] that no such embedding can have distortion less than 3/2, and it was conjectured that a much stronger lower bound should hold.

Thus, as the previous approaches don't work, we turn to the embeddings we defined specifically for fuzzy extractors: biometric embeddings. 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 biometric embeddings is that they preserve some min-entropy: we do not want too many points to collide together, although collisions are allowed, even collisions of distant points. We will build a biometric embedding from the edit distance to the set difference.

A *c-shingle* [5], which is a length-c consecutive substring of a given string w. A *c-shingling* [5] of a string w of length n is the set (ignoring order or repetition) of all $(n-c+1)$ c-shingles of w. Thus, the range of the c-shingling operation consists of all nonempty subsets of size at most $n-c+1$ of $\{0,1\}^c$. To simplify our future computations, we will always arbitrarily pad the c-shingling of any string w to contain precisely n distinct shingles (say, by adding the first $n-|c\text{-shingling}|$ elements of $\{0,1\}^c$ not present in the given c-shingling). Thus, we can define a deterministic map $\mathsf{SH}_c(w)$ which maps w into n substrings of $\{0,1\}^c$, where we assume that $c \geq \log_2 n$. Let $\mathsf{Edit}(n)$ stand for the edit metric over $\{0,1\}^n$, and $\mathsf{SDif}(N,s)$ stand for the set difference metric over $[N]$ where the set sizes are s. We now show that c-shingling yields pretty good biometric embeddings for our purposes.

Lemma 8. *For any $c > \log_2 n$, SH_c is a $(t_1, t_2 = ct_1, m_1, m_2 = m_1 - \frac{n \log_2 n}{c})$-biometric embedding of $\mathsf{Edit}(n)$ into $\mathsf{SDif}(2^c, n)$.*

Proof. Assume $\mathsf{dis}(w_1, w_1')_{ed} \leq t_1$ and that I is the smallest set of $2t_1$ insertions and deletions which transforms w into w'. It is easy to see that each character deletion or insertion affects at most c shingles, and thus the symmetric difference between $\mathsf{SH}_c(w_1)$ and $\mathsf{SH}_c(w_2) \leq 2ct_1$, which implies that $\mathsf{dis}(\mathsf{SH}_c(w_1), \mathsf{SH}_c(w_2))_{sd} \leq ct_1$, as needed.

Now, assume w_1 is any string. Define $g_c(w_1)$ as follows. One computes $\mathsf{SH}_c(w_1)$, and stores n resulting shingles in lexicographic order $h_1 \dots h_n$. Next, one naturally partitions w_1 into n/c disjoint shingles of length c, call them $k_1 \dots k_{n/c}$. Next, for $1 \leq j \leq n/c$, one sets $p_c(j)$ to be the index $i \in \{1 \dots n\}$ such that $k_j = h_i$. Namely, it tells the index of the j-th disjoint shingle of w_1 in the ordered n-set $\mathsf{SH}_c(w_1)$. Finally, one sets $g_c(w_1) = (p_c(1) \dots p_c(n/c))$. Notice, the length of $g_c(w_1)$ is $\frac{n}{c} \cdot \log_2 n$, and also that w_1 can be completely recovered from $\mathsf{SH}_c(w_1)$ and $g_c(w_1)$.

Now, assume W_1 is any distribution of min-entropy at least m_1 on $\mathsf{Edit}(n)$. Since $g_c(W)$ has length $(n \log_2 n/c)$, its min-entropy is at most this much as well. But since min-entropy of W_1 drops to 0 when given $\mathsf{SH}_c(W_1)$ and $g_c(W_1)$, it means that the min-entropy of $\mathsf{SH}_c(W_1)$ must be at least $m_2 \geq m_1 - (n \log_2 n)/c$, as claimed.

We can now optimize the value c. By either Lemma 6 or Theorem 1, for arbitrary universe size (in our case 2^c) and distance threshold $t_2 = ct_1$, we can construct a secure sketch for the set difference metric with min-entropy loss $2t_2 \log_2(2^c) = 2t_1c^2$, which leaves us total min-entropy $m_2' = m_2 - 2t_1c^2 \geq m_1 - \frac{n \log n}{c} - 2t_1c^2$. Applying further Lemma 1, we can convert it into a fuzzy extractor over $\mathsf{SDif}(2^c, n)$ for the min-entropy level m_2 with error ϵ, which can extract at least $\ell = m_2' - 2\log\left(\frac{1}{\epsilon}\right) \geq m_1 - \frac{n \log n}{c} - 2t_1c^2 - 2\log\left(\frac{1}{\epsilon}\right)$ bits, while still correcting $t_2 = ct_1$ of errors in $\mathsf{SDif}(2^c, n)$. We can now apply Lemma 2 to get an $(\mathsf{Edit}(n), m_1, m_1 - \frac{n \log n}{c} - 2t_1c^2 - 2\log\left(\frac{1}{\epsilon}\right), t_1, \epsilon)$-fuzzy extractor. Let us now optimize for the value of $c \geq \log_2 n$. We can set $\frac{n \log n}{c} = 2t_1c^2$, which gives $c = (\frac{n \log n}{2t_1})^{1/3}$. We get $\ell = m_1 - (2t_1n^2 \log^2 n)^{1/3} - 2\log\left(\frac{1}{\epsilon}\right)$ and therefore

Theorem 2. *There is an efficient* $(\mathsf{Edit}(n), m_1, m_1 - (2t_1n^2 \log^2 n)^{1/3} - 2\log\left(\frac{1}{\epsilon}\right), t_1, \epsilon)$ *fuzzy extractor. Setting* $t_1 = m_1^3/(16n^2 \log^2 n)$*, we get an efficient* $(\mathsf{Edit}(n), m_1, \frac{m_1}{2} - 2\log\left(\frac{1}{\epsilon}\right), \frac{m_1^3}{16n^2 \log^2 n}, \epsilon)$ *fuzzy extractor. In particular, if* $m_1 = \Omega(n)$*, one can extract* $\Omega(n)$ *bits while tolerating* $\Omega(n/\log^2 n)$ *insertions and deletions.*

Acknowledgements

We thank Piotr Indyk for discussions about embeddings and for his help in the proof of Lemma 8. We are also thankful to Madhu Sudan for helpful discussions about the construction of [16] and the uses of error-correcting codes. Finally, we thank Rafi Ostrovsky for discussions in the initial phases of this work and Pim Tuyls for pointing out relevant previous work.

The work of the first author 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 second author was partly funded by the National Science Foundation under Grant No. CCR-0311485. The work of the third author was partly funded by US A.R.O. grant DAAD19-00-1-0177 and by a Microsoft Fellowship.

References

1. E. Agrell, A. Vardy, and K. Zeger. Upper bounds for constant-weight codes. *IEEE Transactions on Information Theory*, **46**(7), pp. 2373–2395, 2000.
2. A. Andoni, M. Deza, A. Gupta, P. Indyk, S. Raskhodnikova. Lower bounds for embedding edit distance into normed spaces. In *Proc. ACM Symp. on Discrete Algorithms, 2003*, pp. 523–526.
3. C. Bennett, G. Brassard, and J. Robert. Privacy Amplification by Public Discussion. *SIAM J. on Computing*, **17**(2), pp. 210–229, 1988.
4. C. Bennett, G. Brassard, C. Crépeau, and U. Maurer. Generalized Privacy Amplification. *IEEE Transactions on Information Theory*, **41**(6), pp. 1915-1923, 1995.
5. A. Broder. On the resemblence and containment of documents. In *Compression and Complexity of Sequences*, 1997.

6. A. E. Brouwer, J. B. Shearer, N. J. A. Sloane, and W. D. Smith, "A new table of constant weight codes," *IEEE Transactions on Information Theory*, **36**, p. 1334–1380, 1990.
7. C. Crépeau. Efficient Cryptographic Protocols Based on Noisy Channels. In *Advances in Cryptology — EUROCRYPT 1997*, pp. 306–317.
8. G. Davida, Y. Frankel, B. Matt. On enabling secure applications through off-line biometric identification. In *Proc. IEEE Symp. on Security and Privacy*, pp. 148–157, 1998.
9. Y.Z. Ding. Manuscript.
10. C. Ellison, C. Hall, R. Milbert, B. Schneier. Protecting Keys with Personal Entropy. *Future Generation Computer Systems*, **16**, pp. 311–318, 2000.
11. N. Frykholm. Passwords: Beyond the Terminal Interaction Model. *Master's Thesis*, Umea University.
12. N. Frykholm, A. Juels. Error-Tolerant Password Recovery. In *Proc. ACM Conf. Computer and Communications Security, 2001*, pp. 1–8.
13. V. Guruswami, M. Sudan. Improved Decoding of Reed-Solomon and Algebraic-Geometric Codes. In *Proc. 39th IEEE Symp. on Foundations of Computer Science*, 1998, pp. 28–39.
14. J. Håstad, R. Impagliazzo, L. Levin, M. Luby. A Pseudorandom generator from any one-way function. In *Proc. 21st ACM Symp. on Theory of Computing*, 1989.
15. A. Juels, M. Wattenberg. A Fuzzy Commitment Scheme. In *Proc. ACM Conf. Computer and Communications Security, 1999*, pp. 28–36.
16. A. Juels and M. Sudan. A Fuzzy Vault Scheme. In *IEEE International Symposium on Information Theory*, 2002.
17. J. Kelsey, B. Schneier, C. Hall, D. Wagner. Secure Applications of Low-Entropy Keys. In *Proc. of Information Security Workshop*, pp. 121–134, 1997.
18. J.-P. M. G. Linnartz, P. Tuyls. New Shielding Functions to Enhance Privacy and Prevent Misuse of Biometric Templates. In *AVBPA 2003*, p. 393–402.
19. J.H. van Lint. *Introduction to Coding Theory*. Springer-Verlag, 1992, 183 pp.
20. F. Monrose, M. Reiter, S. Wetzel. Password Hardening Based on Keystroke Dynamics. In *Proc. ACM Conf. Computer and Communications Security, 1999*, p. 73–82.
21. F. Monrose, M. Reiter, Q. Li, S. Wetzel. Cryptographic key generation from voice. In *Proc. IEEE Symp. on Security and Privacy*, 2001.
22. F. Monrose, M. Reiter, Q. Li, S. Wetzel. Using voice to generate cryptographic keys. In *Proc. of Odyssey 2001, The Speaker Verification Workshop*, 2001.
23. N. Nisan, A. Ta-Shma. Extracting Randomness: a survey and new constructions. In *JCSS*, **58**(1), pp. 148–173, 1999.
24. N. Nisan, D. Zuckerman. Randomness is Linear in Space. In *JCSS*, **52**(1), pp. 43–52, 1996.
25. J. Radhakrishnan and A. Ta-Shma. Tight bounds for depth-two superconcentrators. In *Proc. 38th IEEE Symp. on Foundations of Computer Science*, 1997, pp. 585–594.
26. R. Shaltiel. Recent developments in Explicit Constructions of Extractors. *Bulletin of the EATCS*, **77**, pp. 67–95, 2002.
27. V. Shoup. A Proposal for an ISO Standard for Public Key Encryption. Available at `http://eprint.iacr.org/2001/112`, 2001.
28. E. Verbitskiy, P. Tyls, D. Denteneer, J.-P. Linnartz. Reliable Biometric Authentication with Privacy Protection. In *Proc. 24th Benelux Symposium on Information theory*, 2003.