Detection of Algebraic Manipulation with Applications to Robust Secret Sharing and Fuzzy Extractors

February 1, 2008

Ronald Cramer[1,2], Yevgeniy Dodis[3], Serge Fehr[2], Carles Padró[4], and Daniel Wichs[3]

[1] Mathematical Institute, Leiden University, The Netherlands
[2] CWI Amsterdam, The Netherlands
{cramer,fehr}@cwi.nl
[3] New York University
{dodis,wichs}@cs.nyu.edu
[4] Universitat Politècnica de Catalunya, Barcelona, Spain
cpadro@ma4.upc.edu

Abstract. Consider an abstract storage device $\Sigma(\mathcal{G})$ that can hold a single element x from a fixed, publicly known finite group $\mathcal{G}$. Storage is private in the sense that an adversary does not have read access to $\Sigma(\mathcal{G})$ at all. However, $\Sigma(\mathcal{G})$ is non-robust in the sense that the adversary can modify its contents by adding some offset $\Delta \in \mathcal{G}$. Due to the privacy of the storage device, the value Δ can only depend on an adversary's *a priori* knowledge of x. We introduce a new primitive called an *algebraic manipulation detection* (AMD) code, which encodes a source s into a value x stored on $\Sigma(\mathcal{G})$ so that any tampering by an adversary will be detected, except with a small error probability δ. We give a nearly optimal construction of AMD codes, which can flexibly accommodate arbitrary choices for the length of the source s and security level δ. We use this construction in two applications:

- We show how to efficiently convert any linear secret sharing scheme into a *robust secret sharing scheme*, which ensures that no *unqualified subset* of players can modify their shares and cause the reconstruction of some value $s' \neq s$.
- We show how how to build nearly optimal *robust fuzzy extractors* for several natural metrics. Robust fuzzy extractors enable one to reliably extract and later recover random keys from noisy and non-uniform secrets, such as biometrics, by relying only on *non-robust public storage*. In the past, such constructions were known only in the random oracle model, or required the entropy rate of the secret to be greater than half. Our construction relies on a randomly chosen common reference string (CRS) available to all parties.

1 Introduction

We consider an abstract storage device $\Sigma(\mathcal{G})$ that can hold a single element x from a fixed, publicly known finite (additive) group $\mathcal{G}$. Storage is private in the sense that an adversary does not have read access to $\Sigma(\mathcal{G})$ at all. However, $\Sigma(\mathcal{G})$ allows tampering in the sense that an adversary may manipulate the stored value x by adding some offset $\Delta \in \mathcal{G}$ of his choice. As a result, $\Sigma(\mathcal{G})$ stores the element $x + \Delta \in \mathcal{G}$. Due to the privacy of the storage device, the value Δ can only depend on an adversary's *a priori* knowledge of x. For instance, one-time-pad encryption can be understood as such a storage device: it hides the message perfectly, but an adversary can add (bitwise-xor) a string to the message without being detected. Of course, by itself, this example is not very interesting, since it requires some *additional private and tamper-proof storage* for the one-time pad key. [1] However, in the two applications discussed below, no other private or tamper-proof storage is available and hence we will need to use $\Sigma(\mathcal{G})$ alone to achieve authenticity.

[1] For example, by using a slightly longer secret key containing a key to a one-time MAC in addition to the one-time-pad key, one can trivially add authentication to this application.

1.1 Linear Secret Sharing Schemes

In a *linear secret sharing scheme* (e.g. Shamir's secret sharing [26] and many others) a secret s is distributed among n players so that each player gets some algebraic *share* of the secret. Any *qualified* subset of the players can pool their shares together and recover s by means of a linear transformation over the appropriate domain while any *unqualified* subset gets no information about s. Unfortunately, the correctness of the recovery procedure is guaranteed only if all the shares are correct. In particular, if a qualified subset of the players pools their shares for reconstruction, but the honest players among them form an unqualified set, then the dishonest players (possibly just one!) can cause the reconstruction of a modified secret. Moreover, the difference between the correct secret s and the reconstructed secret s' is controlled by the corrupted players, due to the linearity of the scheme. Luckily, this is "all" that the corrupted players can do: (1) by the privacy of the secret sharing scheme, the noise introduced by the corrupted players can only depend on their prior knowledge of the secret and (2) by the linearity of the secret sharing scheme, for any attempted modification of their shares, the corrupted players must "know" the additive difference between s and s'. In essence, a linear secret sharing scheme of s can be viewed as storing s on our abstract device $\Sigma(\mathcal{G})$.

To deal with this problem, we introduce the notion of an *algebraic manipulation detection* (AMD) code. This is a probabilistic encoding of a source s from a given set $\mathcal{S}$ as an element of the group $\mathcal{G}$, with unique decodability. The security of the code ensures that, when the encoding is stored in $\Sigma(\mathcal{G})$, any manipulation of contents by an adversary will be detected except with a small error probability δ. The guarantee holds even if the adversary has full a priori knowledge of the source state s. No secret keys are required since we rely on the privacy of $\Sigma(\mathcal{G})$ instead.

Using an AMD code, we can turn any linear secret sharing scheme into a *robust secret sharing scheme* [28], which ensures that no unqualified subset of players can modify their shares and cause the reconstruction of some value $s' \neq s$. The transformation is very simple: apply the linear secret sharing scheme to the encoding of s rather than s itself.

In terms of parameters, we obtain robust secret sharing schemes which are nearly as efficient as their non-robust counterparts, since the overhead added by encoding a source will be very small. More precisely, to achieve security $2^{-\kappa}$, we build an AMD code where the length of the encoding of a u-bit value s is only $2\kappa + \mathcal{O}(\log(u/\kappa))$ bits longer than the length of s. This construction is close to optimal since we prove a lower bound of 2κ on the amount of overhead that an AMD encoding must add to the size of the source. As a concrete example, in order to robustly secret share a 1 megabyte message with security level $\delta = 2^{-128}$, our best construction adds fewer than 300 bits by encoding the message, whereas previous constructions (described below) add nearly 2 megabytes.

Relation to Prior Work on Secret Sharing. Although AMD codes were never formally defined in previous work, some constructions of AMD codes have appeared, mostly in connection with making secret sharing robust [20, 7, 21]. Although some of these constructions are essentially optimal, all of them are largely inflexible in that the error probability δ is dictated by the cardinality of the source space $\mathcal{S}$: $\delta \approx 1/|\mathcal{S}|$. In particular, this implies that when the cardinality of $\mathcal{S}$ is large, the known constructions may introduce significantly more overhead than what is needed to achieve a particular security threshold. In contrast, our constructions can accommodate arbitrary choices of security δ and message length u.

For example, Cabello, Padró and Sáez [7] (see also [23, 22]) proposed an elegant construction of a robust secret sharing scheme which implicitly relies on the following AMD code. For any finite field $\mathbb{F}$ of order q, the encoding of the secret $s \in \mathbb{F}$ is a triple $(s, x, x \cdot s)$, where $x \in_R \mathbb{F}$. This code achieves security $\delta = 1/q$ and optimal message overhead $2\log(q) = 2\log(1/\delta)$ for this value of δ. However, as already mentioned, it is far from optimal when we only desire a security level $\delta \gg 1/q$, making this construction inflexible for many applications. Similarly, Ogata and Kurosawa [20] proposed an inflexible construction of a *weakly* robust secret sharing scheme (the scheme is robust only if the shared secret is uniformly random) that implicitly defines what we call a *weak* AMD code. We describe this construction and argue its inflexibility in Appendix C.3. In Appendix D, we also show a natural generic transformation of weak AMD codes to (ordinary "strong") AMD codes, observing that such a transformation can never achieve the optimal overhead (nearly) achieved by our direct construction. AMD codes are also very useful for several other related applications. Indeed, in Section 3 we point their applications to robust information dispersal, secure private storage and anonymous message transmission.

In the context of robust secret sharing, the inflexibility issue mentioned above has recently been addressed in a paper by Obana and Araki [19], where a *flexible* robust secret sharing scheme (in fact, an AMD code in our terminology) was proposed and claimed to be "proven" secure. However, as we discuss in Appendix B, the proposed robust secret sharing scheme (respectively AMD code) is completely *insecure*.

1.2 Fuzzy Extractors

A less obvious example comes from the domain of *fuzzy extractors* [10]. A fuzzy extractor extracts a uniformly random key R from some non-uniform secret w (e.g., biometric data) in such a way that this key can be recovered from any w' sufficiently close to w in some appropriate metric space.[2] To accomplish this task, the fuzzy extractor also computes a public *helper string* P in addition to the extracted key R, and then recovers R using w' and P. Unfortunately, the original notion of a fuzzy extractor critically depends on the value of P being stored on a tamper-proof (though public) device. As observed by Boyen et al. [6, 5], this severely limits the usability of the concept. To address this problem, [6, 5] introduced a stronger notion of a *robust fuzzy extractor*, where any tampering of P will be detected by the user, even with an imperfect reading w' of w! Thus, P can be stored on a potentially untrusted server without the fear that a wrong key $\tilde{R} \neq R$ will be extracted.

Before describing the new and prior results on robust fuzzy extractors, let us give some intuition on how this setting is related to our abstract storage device. As we will show (extending the previous observation of [11]), for "appropriately designed" (non-robust) fuzzy extractors, the effect of modifying the helper string P into $\tilde{P}$ can be essentially subsumed by giving the attacker the ability to *control* the difference between the original key R extracted from w, and the "defective" key $\tilde{R}$ extracted from w' and $\tilde{P}$. Thus, on a very high level, storing the public helper P on a public and unprotected storage can be viewed as implicitly storing the extracted key R on a device $\Sigma(\mathcal{G})$ that ensures privacy but allows tampering.

Unfortunately, in this application one does not have the freedom of storing some encoding of R on $\Sigma(\{0,1\}^u)$, so AMD codes are not directly applicable. Instead, we introduce a related notion called a *(one-time) message authentication code with key manipulation security* (KMS-MAC). Abstractly, this authentication code is keyed by a random element of some finite group $\mathcal{G}$, and remains secure even if the key is stored in $\Sigma(\mathcal{G})$. The message and the authentication tag can be stored in insecure storage that is neither private nor tamper-proof. The adversary, who gets to see one valid message/tag pair and modify the key stored on $\Sigma(\mathcal{G})$, will be unable to produce an alternative message/tag pair that verifies under the modified key, except with some small error probability δ. We show how to construct KMS-MACs using appropriate AMD codes. Combined with our nearly optimal AMD construction, we get KMS-MACs that essentially achieve the same parameters as ordinary (one-time) MACs: to authenticate an u-bit message with substitution security $2^{-\kappa}$, one uses a key of size $2\kappa + \mathcal{O}(\log(u/\kappa))$ and a tag of size $\kappa + \mathcal{O}(\log(u/\kappa))$.

[2] For now and much of the paper, we will concentrate on the Hamming space over $\{0,1\}^n$, later pointing out how to extend our results to related metrics.

We use KMS-MACs to add robustness to fuzzy extractors. As we mentioned, the public helper P is stored on a public unprotected storage and we can think of the extracted key R as being stored in $\Sigma(\{0,1\}^u)$. Surprisingly, we can use the key R (which is derived from P) to authenticate P itself! The idea is to split the extracted key R into two parts R_{mac} and R_{out}. The "long" R_{out} will be the new extracted key, while the "short" R_{mac} will be sacrificed and used as the key to the KMS-MAC applied to the original helper string P (so that the new helper string will contain P and the tag). An adversary that replaces P with P' implicitly adds a known offset to R_{mac} but, by the security of the KMS-MAC, is then unable to compute a valid tag for P' under the modified key. As a result, for the first time, we obtain *robust* fuzzy extractors for the Hamming (and related) metrics, which do not rely on random oracles (or other computational assumptions) and achieve *nearly the same optimal parameters as their non-robust counterparts*. However, as we explain shortly, this result is obtained in the Common Reference String model. Indeed, a setup assumption is necessary as our result breaks the impossibility result of [12] for the plain model.

Relation to Prior Work on Fuzzy Extractors. In their original paper, Dodis et al. [10] gave several nearly optimal constructions for (non-robust) fuzzy extractors for the Hamming and several other metrics. Boyen et al. [5] gave a generic transformation which makes a fuzzy extractor robust *in the random oracle model*, without considerably sacrificing any of the parameters. Unfortunately, in the plain model Dodis et al. [11] showed that robustness can only be achieved if the initial secret w contains an entropy rate of at least one half (i.e. the entropy of the secret is at least half the length of the secret). In fact, this holds even if no errors are allowed [12] (i.e., $w = w'$). Moreover, even when the secret does meet this threshold, robustness is only achieved at a large cost in the length of the extracted random key, as compared to the optimal non-robust extractors for the same entropy threshold.

In this work we overcome this pessimistic state of affairs by building robust fuzzy extractors in the *Common Reference String* (CRS) model. The common reference string can be chosen once when the system is designed and can be hardwired/hardcoded into all hardware/software implementing the system. Moreover, the CRS can be published publicly and we allow the attacker to observe (but not modify) it.[3] Our CRS is a random bitstring - it has no trapdoors and we do not require any ability to "program" it. Since most users do not create their own hardware/software but instead assume that a third party implementation is correct, the assumption that this implementation also contains an honestly generated random string does not significantly increase the amount of trust required from users. We do assume that the probability distribution from which the secret w is chosen is independent of the CRS. This is a very natural assumption for biometrics and many other scenarios. However, it also means that our scheme is not applicable in the setting of exposure resilient cryptography (see [9]) where the attacker can learn some function of the secret after seeing the CRS.

What our result shows, however, is that this seemingly minor addition not only allows us to achieve robustness without additional restrictions on the entropy rate of the secret, but also to *nearly match the extracted key length of non-robust fuzzy extractor constructions* (or the robust fuzzy extractor constructions in the random oracle model [5]).

On a technical level, it is also interesting to compare our model and techniques with those of Dodis et al. [11], who built robust fuzzy extractors in the plain model (with the necessarily poor parameters mentioned above). The work of [11] could be viewed (in our language) as reducing the question of building robust fuzzy extractors to that of using the *the original secret* w stored in $\Sigma(\mathcal{G})$, for authentication purposes. In particular, the authors had to build a message authentication code (in fact, one secure against key manipulation

[3] We remark that assuming tamper-proof storage of the CRS, which can be shared by many users, is very different than assuming tamper-proof storage of a "user-specific" helper string P. Indeed, the former can be hardwired into the system, and the latter can not.

attacks) using the *non-uniform* string w as the key. Authentication codes keyed by non-uniform randomness imply non-trivial parameter degradation in the plain model [12] and all the (necessary) inefficiencies of [11] followed from this fact. In contrast, the addition of the CRS reduces the question of building robust fuzzy extractors to that of using uniformly random *extracted randomness* R, stored on $\Sigma(\mathcal{G})$, for authentication purposes (this implication is non-trivial and forms one of the contributions of this work). As a consequence, we can use much more efficient KMS-MACs relying on *uniformly random* secret keys and, therefore, obtain nearly optimal robust fuzzy extractors in the CRS model.

2 Algebraic Manipulation Detection Codes

Definition 1. *An* (S, G, δ)-algebraic manipulation detection code*, or* (S, G, δ)-AMD code *for short, is a* probabilistic encoding *map* $\mathcal{E} : \mathcal{S} \to \mathcal{G}$ *from a set* $\mathcal{S}$ *of size* S *into an (additive) group* $\mathcal{G}$ *of order* G*, together with a (deterministic)* decoding *function* $D : \mathcal{G} \to \mathcal{S} \cup \{\perp\}$ *such that* $D(\mathcal{E}(s)) = s$ *with probability 1 for any* $s \in \mathcal{S}$*. The security of an AMD code requires that for any* $s \in \mathcal{S}$*,* $\Delta \in \mathcal{G}$*,* $\Pr[D(\mathcal{E}(s) + \Delta) \notin \{s, \perp\}] \leq \delta$*.*

An AMD code is called systematic *if* $\mathcal{S}$ *is a group, and the encoding is of the form*

$$\mathcal{E} : \mathcal{S} \to \mathcal{S} \times \mathcal{G}_1 \times \mathcal{G}_2,\ s \mapsto (s, x, f(x, s))$$

for some function f *and* $x \in_R \mathcal{G}_1$*. The decoding function of a systematic AMD code is naturally given by* $D(s', x', \sigma') = s'$ *if* $\sigma' = f(x', s')$ *and* $\perp$ *otherwise.*

Intuitively, $\mathcal{E}(s)$ can safely be stored on a private storage device $\Sigma(\mathcal{G})$ so that an adversary who manipulates the stored value by adding an offset Δ, cannot cause it to decode to some $s' \neq s$. It is also possible to define a *weak* AMD code where security only holds for a *random* $s \in \mathcal{S}$ rather than an arbitrary one. We focus of regular (strong) AMD codes and mention some constructions and applications of weak AMD codes in the appendices.

From a practical perspective, it is typically not sufficient to have one particular code, but rather one would like to have a class of codes at hand such that for every choice u for the bit-length of the source s and for every choice κ of the security level, there exists a code that "fits" these parameters. This motivates the following definition:

Definition 2. *An AMD code* family *is a class of AMD codes such that for any* $\kappa, u \in \mathbb{N}$ *there exists an* (S, G, δ)*-AMD code in that class with* $S \geq 2^u$ *and* $\delta \leq 2^{-\kappa}$*.*

We point out that in this definition, the group $\mathcal{G}$ can be different for every AMD code in the family and is left unspecified. In our constructions the group $\mathcal{G}$ will often be the additive group of the vector space $\mathbb{F}^d$ for some field $\mathbb{F}$. Specifically, we will often focus on the field $\mathbb{F}_{2^d}$ (as an additive group, this is equivalent to $\mathbb{F}_2^d$) so addition (and subtraction) is just bitwise-xor of d bit long strings.

We would like the construction of an AMD code to be close to optimal in that G should not be much larger than S . We consider the *tag size* ϖ of a (S, G, δ)-AMD code defined as $\varpi = \log(G) - \log(S)$. Intuitively, this denotes the number of bits that the AMD code appends to the source. More generally we define the efficiency of an AMD code family as follows.

Definition 3. *The* effective tag size $\varpi^*(\kappa, u)$ *with respect to* $\kappa, u \in \mathbb{N}$ *of an AMD code family is defined as* $\varpi^*(\kappa, u) = \min\{\log(G)\} - u$ *where the minimum is over all* (S, G, δ)*-AMD codes in that class with* $S \geq 2^u$ *and* $\delta \leq 2^{-\kappa}$*.*

In Appendix A, we prove the following lower bound on the effective tag size of an AMD code family.

Theorem 1. *Any AMD code family has an affective tag size lower bounded by* $\varpi^*(\kappa, u) \geq 2\kappa - 2^{-u+1} \geq 2\kappa - 1$.

2.1 Optimal and Flexible Construction

We are now ready to present a construction of AMD codes which is both optimal and flexible. As noted in the introduction, a similar, but more complicated construction appeared in [11], though it was presented as part of a larger construction, and its properties were not stated explicitly as a stand-alone primitive. The two constructions were discovered concurrently and independently from each other.

Let $\mathbb{F}$ be a field of size q and characteristic p, and let d be any integer such that $d+2$ is not divisible by p. Define the function $\mathcal{E} : \mathbb{F}^d \to \mathbb{F}^d \times \mathbb{F} \times \mathbb{F}$ by $E(s) = (s, x, f(x,s))$ where

$$f(x,s) = x^{d+2} + \sum_{i=1}^{d} s_i x^i$$

Theorem 2. *The given construction is a systematic $(q^d, q^{d+2}, (d+1)/q)$-AMD code with tag size $\varpi = 2\log q$.*

Proof. We wish to show that for any $s \in \mathbb{F}$ and $\Delta \in \mathbb{F}^{d+2}$: $\Pr[D(\mathcal{E}(s)+\Delta) \notin \{s, \perp\}] \le \delta$. It is enough to show that for any $s' \neq s$ and any $\Delta_x, \Delta_f \in \mathbb{F}$: $\Pr[f(x,s)+\Delta_f = f(x+\Delta_x, s')] \le \delta$. Hence we consider the event

$$x^{d+2} + \sum_{i=1}^{d} s_i x^i + \Delta_f = (x+\Delta_x)^{d+2} + \sum_{i=1}^{d} s'_i (x+\Delta_x)^i \tag{1}$$

We rewrite the right hand side of (1) as $x^{d+2} + (d+2)\Delta_x x^{d+1} + \sum_{i=1}^{d} s'_i x^i + \Delta_x \cdot p(x)$, where $p(x)$ is some polynomial of degree at most d in x. Subtracting this term from both sides of equation (1), x^{d+2} cancels out and we get

$$-(d+2)\Delta_x x^{d+1} + \sum_{i=1}^{d} (s_i - s'_i) x^i - \Delta_x \cdot p(x) + \Delta_f = 0 \tag{2}$$

We claim that the left side of equation 2 is a *non-zero* polynomial of degree at most $d+1$. To see this, let us consider two cases:

1. If $\Delta_x \neq 0$, then the leading coefficient is $-(d+2)\Delta_x \neq 0$ (here we use the fact that $d+2$ is not divisible by the characteristic of the field).
2. If $\Delta_x = 0$, then (2) simplifies to $\sum_{i=1}^{d} (s_i - s'_i) x^i + \Delta_f = 0$, which is not identically zero since we assumed that $s \neq s'$.

This shows that (2) has at most $d+1$ solutions x. Let B be the set of such solutions so $|B| \le d+1$. Then

$$\Pr[D(\mathcal{E}(s)+\Delta) \notin \{s,\perp\}] = \Pr_{x \leftarrow \mathbb{F}}[x \in B] \le \frac{d+1}{q}$$

□

Notice, the elements of the range group $\mathcal{G} = \mathbb{F}^d \times \mathbb{F} \times \mathbb{F}$ can be conveniently viewed as elements of $\mathbb{Z}_p^t$, for some t (recall, p is the characteristic of $\mathbb{F}$). Thus, addition in $\mathcal{G}$ simply corresponds to element-wise addition modulo p. When $p = 2$, this simply becomes the XOR operation.

Quantifying the above construction over all fields $\mathbb{F}$ and all values of d (such that $d+2$ is not divisible by p), we get a very flexible AMD family. Indeed, we show that the effective tag size of the family is nearly optimal.

Corollary 1. *The effective tag size of the AMD code family is $\varpi^*(\kappa, u) \leq 2\kappa + 2\log(\frac{u}{\kappa} + 3) + 2$. Moreover, this can be achieved with the range group $\mathcal{G}$ being the group of bitstrings under the bitwise-xor operation.*[4]

Proof. For a given κ and u, choose d and q as follows: let d be the smallest positive *odd* integer such that $u \leq d(\kappa + \log(d+1))$, and choose $q = 2^{\lceil \kappa + \log(d+1) \rceil}$. Note that $d+2$ is not divisible by 2, which is the characteristic of $\mathbb{F}_q$. Furthermore $u \leq d\log(q)$, and thus we can restrict the source space $\mathbb{F}^d$, viewed as $\{0,1\}^{d\log(q)}$, to the subset $\mathcal{S} = \{0,1\}^u$ and the range $\mathbb{F}^d \times \mathbb{F} \times \mathbb{F}$ to the subgroup $\mathcal{G} = \mathcal{S} \times \mathbb{F} \times \mathbb{F}$. The resulting (S, G, δ)-AMD code fits κ and u in that $S \geq 2^u$ and $\delta = (d+1)/q \leq 2^{-\kappa}$. The effective tag size is given by:

$$\begin{aligned} \log(G) - u = \log(|\mathcal{S} \times \mathbb{F} \times \mathbb{F}|) - u = 2\log(q) &\leq 2\kappa + 2\log(d+1) + 2 \\ &\leq 2\kappa + 2\log(\tfrac{u}{\kappa} + 3) + 2\,. \end{aligned}$$

Thus $\varpi^*(\kappa, u) \leq 2\kappa + 2\log(\frac{u}{\kappa} + 3) + 2$. □

3 Application to Robust Secret Sharing

A *secret sharing scheme* is given by two probabilistic functions. The function Share maps a secret s from some group $\mathcal{G}$ to a vector $S = (S_1, \ldots, S_n)$ where the *shares* S_i are in some group $\mathcal{G}_i$. The function $\mathsf{Recover}$ takes as input a vector of shares $\tilde{S} = (\tilde{S}_1, \ldots, \tilde{S}_n)$ where $\tilde{S}_i \in \mathcal{G}_i \cup \{\perp\}$ and outputs $\tilde{s} \in \mathcal{G} \cup \{\perp\}$. A secret sharing schemes is defined over some *monotone access structure* which maps subsets $B \subseteq \{1, \ldots, n\}$ to a status: $\texttt{qualified}, \texttt{unqualified}, \perp$. The correctness property of such a scheme states that for any $s \in \mathcal{G}$ and any *qualified* set B, the following is true with probability 1. If $S \leftarrow \mathsf{Share}(s)$ and $\tilde{S}$ is defined to be $\tilde{S}_i = S_i$ for each $i \in B$ and $\tilde{S}_i = \perp$ for each $i \notin B$, then $\mathsf{Recover}(\tilde{S}) = s$. Similarly, the privacy of such a scheme states that for any *unqualified* subset A, the shares $\{S_i\}_{i \in A}$ reveal no information about the secret s (this is formalized using standard indistinguishability).

Thus, qualified sets of players can recover the secret from their pooled shares, while unqualified subsets learn no information about the secret. Sets of players which are neither qualified nor unqualified might not be able to recover the secret in full but might gain some partial information about its value.

A *linear* secret sharing scheme has the property that the $\mathsf{Recover}$ function is linear: given any $s \in \mathcal{G}$, any $S \in \mathsf{Share}(s)$, and any vector S' (possibly containing some $\perp$ symbols), we have $\mathsf{Recover}(S + S') = s + \mathsf{Recover}(S')$, where vector addition is defined element-wise and addition with $\perp$ is defined by $\perp + x = x + \perp = \perp$ for all x.

Examples of linear secret sharing schemes include Shamir's secret sharing scheme [26] where the access structure is simply a threshold on the number of players, or a scheme for a general access structure in [16].

We consider a setting where an honest dealer uses a secret sharing scheme to share some secret s among n players. Later, an outside entity called the *reconstructor* contacts some qualified subset B of the players, collects their shares and reconstructs the secret. The security of the scheme ensures that, as long as the set $A \subseteq B$ of players corrupted by an adversary is unqualified, the adversary gets no information about the shared secret. However, if the *honest* players $B \backslash A$ also form an unqualified subset, then the adversary can enforce the reconstruction of an incorrect secret by handing in incorrect shares. In fact, if the reconstructor contacts a *minimal* qualified subset of the players, then even a single corrupted player can cause the reconstruction of an incorrect secret. Robust secret sharing schemes (defined in [28, 4]) ensure that such attacks

[4] We can also imagine situations where the "base" field $\mathbb{F}'$ of some characteristic p is given to us, and our freedom is in choosing the extension field $\mathbb{F}$ and the appropriate value of d so that $\mathcal{S}$ can be embedded into $\mathbb{F}^d$. Under such restrictions, the effective tag size becomes roughly $2\kappa + 2\log(u) + O(\log p)$.

can't succeed: as long as the adversary corrupts only an unqualified subset of the players, the reconstructor will never recover a modified version of the secret.

Definition 4. *A secret sharing scheme is δ-robust if for any unbounded adversary $\mathcal{A}$ who corrupts an unqualified set of players $A \subseteq \{1,\ldots,n\}$ and any $s \in \mathcal{G}$, we have the following. Let $S \leftarrow \mathsf{Share}(s)$ and $\tilde{S}$ be a value such that, for each $1 \le i \le n$,*

$$\tilde{S}_i = \begin{cases} \mathcal{A}(i, s, \{S_i\}_{i\in A}) & \text{if } i \in A \\ S_i \text{ or } \perp & \text{if } i \notin A \end{cases}$$

Then $\Pr[\mathsf{Recover}(\tilde{S}) \notin \{s, \perp\}] \le \delta$.

We note that in a (non-robust) linear secret sharing scheme, when the adversary modifies shares by setting $\tilde{S}_i = S_i + \Delta_i$ then, by linearity of the scheme, the adversary also knows the difference $\Delta = \tilde{s} - s$ between the reconstructed secret $\tilde{s}$ and the shared secret s. This implies that we can think of s as being stored in an abstract storage device $\Sigma(\mathcal{G})$, which is private for an adversary who corrupts an unqualified subset of the players, yet is not-robust in that the adversary can specify additive offsets so that $\Sigma(\mathcal{G})$ stores $s + \Delta$. This immediately implies that we can turn any linear secret sharing scheme into an δ-robust secret sharing scheme using AMD codes.

Theorem 3. *Let* $(\mathsf{Share}, \mathsf{Recover})$ *denote a linear secret sharing scheme with domain $\mathcal{G}$ of order G, and let $(\mathcal{E}, D)$ be an (S, G, δ)-AMD code with range $\mathcal{G}$. Then the scheme* $(\mathsf{Share}^*, \mathsf{Recover}^*)$ *given by* $\mathsf{Share}^*(s) = \mathsf{Share}(\mathcal{E}(s))$, $\mathsf{Recover}^*(\tilde{S}) = D(\mathsf{Recover}(\tilde{S}))$ *is an δ-robust secret sharing scheme.*

Proof. Let $S = \mathsf{Share}^*(S)$ and let $\tilde{S}$ be a vector meeting the requirements of Def. 4. Let $S' = \tilde{S} - S$. The vector S' contains 0 for honest players, $\perp$ for absent players, and arbitrary values for dishonest players. We have:

$$\begin{aligned} \Pr[\mathsf{Recover}^*(\tilde{S}) \notin \{s, \perp\}] &= \Pr[D(\mathsf{Recover}(S) + \mathsf{Recover}(S')) \notin \{s, \perp\}] \\ &= \Pr[D(\mathcal{E}(s) + \Delta) \notin \{s, \perp\}] \end{aligned}$$

where the value $\Delta = \mathsf{Recover}(S')$ is determined by the adversarial strategy $\mathcal{A}$. By the privacy of the secret sharing scheme, it is only based on the adversary's a-priori knowledge of the shared secret and is otherwise independent of the value $\mathcal{E}(s)$. The conclusion then follows immediately from the definition of AMD codes. □

For Shamir secret sharing (and similar schemes), where the group $\mathcal{G}$ can be an arbitrary field of size $q \ge n$, we can use the optimal and flexible AMD code construction from Section 2.1. In doing so, each player's share would increase by roughly $2\log(1/\delta) + 2\log u$ bits (where u in the length of the message) as compared to the non-robust case.

ROBUST INFORMATION DISPERSAL. Systematic AMD codes have an additional benefit in that the encoding leaves the original value s intact. This could be beneficial in the scenario where players do not care about the privacy of s, but only about its authenticity. In other words, it is safe to use *information dispersal* on s or, alternatively, s can be stored in some public non-robust storage. Using a systematic AMD code which maps s to $(s, x, f(x, s))$, the players can just secret share the authentication information $(x, f(x, s))$ and use it later to authenticate s. As long as the corrupted players form an unqualified set, the authentication information $(x, f(x, s))$ remains private and hence an adversary who changes s to s' (and trivially knows the offset $\Delta_s = s - s'$) still cannot come up with an offset to $(x, f(x,s))$ so that it authenticates s' instead of s. The value s might be very large but the authentication information $(x, f(x,s))$ remains relatively small, and hence secret sharing only the authentication information (rather than the entire encoding) gives us significant gains in efficiency. Concretely, to authenticate an u-bit secret s, we only need to secret share roughly $2(\log(1/\delta) + \log u)$ bits.

SECURE AND PRIVATE STORAGE / SECURE MESSAGE TRANSMISSION. Consider again the problem of reconstructing a shared secret in the presence of faulty shares. However, now the goal is not only to prevent the reconstruction of an incorrect secret by detecting foul play, but to ensure that reconstruction always succeeds in producing the correct secret (except with small probability). In other words we do not want to allow the option of reconstructing $\perp$. We still assume the dealer to be honest and that reconstruction is towards one player. However, now we additionally assume that among the players participating in reconstruction, the honest players form a *qualified* set. The dishonest players are still assumed to form an *unqualified* set. This problem is known under the name (unconditional) *secure information dispersal* [24, 17] or non-interactive *secure message transmission* [14, 13]. There is a generic, though for large player sets computationally inefficient, construction based on a robust secret sharing [8]: for every qualified subset of the involved players, invoke the robust reconstruction until for one set of shares no foul play is detected and a secret is reconstructed. If the robust secret sharing scheme is $1/2^{\kappa+n}$-secure, then this procedure succeeds in producing the correct secret except with probability at most $1/2^{\kappa}$.

ANONYMOUS MESSAGE TRANSMISSION. In recent work [3], Broadbent and Tapp explicitly used the notion of AMD codes introduced in this paper (and our construction of them) in the setting of unconditionally secure multi-party protocols with a dishonest majority. Specifically, AMD codes allowed them to obtain robustness in their protocol for anonymous message transmission. This protocol, and with it the underlying AMD code, was then used in [2] as a building block to obtain a protocol for anonymous quantum communication.

4 Message Authentication Codes with Key Manipulation Security

As a notion related to AMD codes, we define message authentication codes which remain secure even if the adversary can manipulate the key. More precisely, we assume that (only) the key of the authentication code is stored on an abstract private device $\Sigma(\mathcal{G})$ to which the adversary has algebraic manipulation access, but the message and the *authentication tag* are stored publicly and the adversary can modify them at will. This is in contrast to AMD codes where the entire encoding of the message is stored in $\Sigma(\mathcal{G})$.

Definition 5. *An* (S, G, T, δ)-message authentication code with key manipulation security *(*KMS MAC*) is a function* $\mathsf{MAC} : \mathcal{S} \times \mathcal{G} \to \mathcal{T}$ *which maps a* source message *in a set* $\mathcal{S}$ *of size* S *to a* tag *in the set* $\mathcal{T}$ *of size* T *using a* key *from a group* $\mathcal{G}$ *of order* G*. We require that for any* $s \neq s' \in S$*, any* $\sigma, \sigma' \in \mathcal{T}$ *and any* $\Delta \in \mathcal{G}$

$$\Pr[\mathsf{MAC}(s', K + \Delta) = \sigma' \mid \mathsf{MAC}(s, K) = \sigma] \leq \delta$$

where the probability is taken over a uniformly random key $K \in_R \mathcal{G}$*.*

Intuitively, the adversary get some message/tag pair (s, σ). The adversary wins if he can produce an offset Δ and a message $s' \neq s$ along with a tag σ' such that the pair (s', σ') verifies correctly under the key $K + \Delta$. The above definition guarantees that such an attack succeeds with probability at most δ. In fact, the definition is slightly stronger than required, since we quantify over all possible tags σ of the message s (rather than just looking at a randomly generated one). However, since the above definition is achievable and simpler to state, we will consider this stronger notion only. We can also think of a KMS-MAC as a generalization of a standard message authentication code, which only guarantees security for $\varDelta = 0$.

As with AMD codes, we will consider the notion of a KMS-MAC family. For efficiency, we are interested in minimizing the tag size $\log(T)$ and the key size $\log(G)$. The following well known lower bounds on standard message authentication codes (e.g., see [27]) obviously also apply to the stronger notion of a KMS-MAC.

Lemma 1. *For any authentication code with security $\delta \leq 2^{-\kappa}$, the key size $\log(G)$ must be at least 2κ, and the tag size $\log(T)$ must be at least κ.*

We now give a construction of a KMS-MAC out of any systematic AMD code.

Theorem 4. *Let $\mathcal{E} : \mathcal{S} \to \mathcal{S} \times \mathcal{G}_1 \times \mathcal{G}_2, s \mapsto (s, x, f(x, s))$ be a systematic $(|\mathcal{S}|, |\mathcal{S}||\mathcal{G}_1||\mathcal{G}_2|, \delta)$-AMD code. Then the function $\mathsf{MAC} : \mathcal{S}\times(\mathcal{G}_1\times\mathcal{G}_2) \to \mathcal{G}_2$ yields a $(|\mathcal{S}|, |\mathcal{G}_1||\mathcal{G}_2|, |\mathcal{G}_2|, \delta)$-KMS-MAC:*

$$\mathsf{MAC}(s, (x_1, x_2)) = f(x_1, s) + x_2$$

Proof. Assume $K = (x_1, x_2) \in \mathcal{G}_1 \times \mathcal{G}_2$ is chosen uniformly at random, and consider arbitrary $\varDelta = (\varDelta_1, \varDelta_2) \in \mathcal{G}_1 \times \mathcal{G}_2$, $\sigma, \sigma' \in \mathcal{G}_2$, and $s, s' \in \mathcal{S}$, where $s \neq s'$.

The event $\mathsf{MAC}(s, K) = \sigma$ is the event $f(x_1, s)+x_2 = \sigma$, which is the same as $x_2 = -f(x_1, s)+\sigma$. Let us call this event E_1. Similarly, the event $\mathsf{MAC}(s', K+\varDelta) = \sigma'$ is the event $f(x_1+\varDelta_1, s')+(x_2+\varDelta_2) = \sigma'$, which is the same as $f(x_1 + \varDelta_1, s') = -x_2 + \sigma' - \varDelta_2$. Let us call this event E_2. Thus, we need to bound $\Pr[E_2 \mid E_1]$.

Let us denote $\varDelta_f = -\sigma + \sigma' - \varDelta_2$ and define an auxiliary event E_2' as $f(x_1 + \varDelta_1, s') = f(x_1, s) + \varDelta_f$. We claim that $\Pr[E_2 \mid E_1] = \Pr[E_2' \mid E_1]$. Indeed, if $x_2 = -f(x_1, s) + \sigma$, then

$$-x_2 + \sigma' - \varDelta_2 = -(-f(x_1, s) + \sigma) + \sigma' - \varDelta_2 = f(x_1, s) + (-\sigma + \sigma' - \varDelta_2) = f(x_1, s) + \varDelta_f$$

Finally, notice that E_2' and E_1 are *independent*. Indeed, since E_2' does not depend on x_2, and x_2 is chosen at random from $\mathcal{G}_2$, whether or not x_2 is equal to $-f(x_1, s) + \sigma$ does not affect any other events not involving x_2. Thus, $\Pr[E_2' \mid E_1] = \Pr[E_2']$. Therefore, we have

$$\Pr[\mathsf{MAC}(s', K + \varDelta) = \sigma' \mid \mathsf{MAC}(s, K) = \sigma] = \Pr[f(x_1 + \varDelta_1, s') = f(x_1, s) + \varDelta_f] \leq \delta$$

where the last inequality follows directly from the security of the AMD code, since $s \neq s'$. □

Using the systematic AMD code family constructed in Section 2.1, we get a nearly optimal KMS-MAC family. In particular, plugging in the systematic AMD code family from Theorem 2 and using the parameters obtained in Corollary 1, we get:

Corollary 2. *There is a KMS-MAC family such that, for any $\kappa, u \in \mathbb{N}$, the family contains an (S, G, T, δ)-KMS-MAC (with respect to XOR operation) with $\delta \leq 2^{-\kappa}$, $S \geq 2^u$ and*

$$\log(G) \leq 2\kappa + 2\log(u/\kappa + 3) + 2$$
$$\log(T) \leq \kappa + \log(u/\kappa + 3) + 1$$

5 Application to Robust Fuzzy Extractors

We start by reviewing the some basic definitions needed to define the notion of fuzzy extractors from [10].

MIN-ENTROPY. The *min-entropy* of a random variable X is $\mathbf{H}_\infty(X) = -\log(\max_x \Pr_X[x])$. Following [10], we define the (average) conditional min-entropy of X given Y as $\widetilde{\mathbf{H}}_\infty(X \mid Y) = -\log(\mathbf{E}_{y \leftarrow Y}(2^{-\mathbf{H}_\infty(X|Y=y)}))$ (here the expectation is taken over y for which $\Pr[Y = y]$ is nonzero). This definition is convenient for cryptographic purposes, because the probability that the adversary will predict X given Y is $2^{-\widetilde{\mathbf{H}}_\infty(X|Y)}$. Finally, we will use [10, Lemma 2.2], which states that $\widetilde{\mathbf{H}}_\infty(X \mid Y) \geq \mathbf{H}_\infty((X,Y)) - \lambda$, where 2^λ is the number of elements in Y.

SECURE SKETCHES. Let $\mathcal{M}$ be a metric space with distance function dis. Informally, a secure sketch enables recovery of a string $w \in \mathcal{M}$ from any "close" string $w' \in \mathcal{M}$ without leaking too much information about w.

Definition 6. *An (m, m', t)-*secure sketch *for a metric space $\mathcal{M}$ is a pair of efficient randomized procedures (*SS, Rec*) s.t.:*

1. *The sketching procedure* SS *on input $w \in \mathcal{M}$ returns a bit string $s \in \{0,1\}^*$. The recovery procedure* Rec *takes an element $w' \in \mathcal{M}$ and $s \in \{0,1\}^*$.*
2. Correctness: *If* $\mathsf{dis}(w, w') \leq t$ *then* $\mathsf{Rec}(w', \mathsf{SS}(w)) = w$.
3. Security: *For any distribution W over $\mathcal{M}$ with min-entropy m, the (average) min-entropy of W conditioned on s does not decrease very much. Specifically, if $\mathbf{H}_\infty(W) \geq m$ then $\widetilde{\mathbf{H}}_\infty(W \mid \mathsf{SS}(W)) \geq m'$.*

The quantity $m - m'$ is called the entropy loss *of the secure sketch.*

As already mentioned in Footnote 2, we will concentrate on the Hamming metric over $\{0,1\}^n$, later extending our results to several related metrics. For this metric we will make use of the *syndrome construction* from [10], which we review in Appendix E (this construction appeared as a component of protocols earlier, e.g., in [1]). For our current purposes, though, we only need to know that this construction is a *linear transformation* over $\mathbb{F}_2^n$.

STATISTICAL DISTANCE. Let X_1, X_2 be two probability distributions over some space S. Their *statistical distance* is $\mathbf{SD}(X_1, X_2) \stackrel{\text{def}}{=} \frac{1}{2}\sum_{s \in S} |\Pr_{X_1}[s] - \Pr_{X_2}[s]|$. If $\mathbf{SD}(X_1, X_2) \leq \varepsilon$, we say they are ε-close, and write $X_1 \approx_\varepsilon X_2$. Note that ε-close distributions cannot be distinguished with advantage better than ε even by a computationally unbounded adversary. We use the notation U_d to denote (fresh) uniform distribution over $\{0,1\}^d$.

RANDOMNESS EXTRACTORS FOR AVG. MIN ENTROPY. A randomness extractor, as defined in [18], extracts a uniformly random string from any secret with high enough entropy using some randomness as a seed. Here we include a slightly altered definition to ensure that we can extract randomness from any secret with high enough *average* min-entropy.

Definition 7. *A function* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^\ell$ *is called a (m, ℓ, ε)-*extractor *if for all random variables X and Y such that $X \in \{0,1\}^n$ and $\widetilde{\mathbf{H}}_\infty(X \mid Y) \geq m$, and $I \leftarrow U_d$, we have*

$$\mathbf{SD}\left(\ (Y, \mathsf{Ext}(X; I), I)\ ,\ (Y, U_\ell, U_d)\ \right) \leq \varepsilon$$

It was shown by [10, Lemma 2.4] that universal hash functions are good extractors in the above sense. In particular, the construction $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^\ell$, defined by $\mathsf{Ext}(x, i) \stackrel{\text{def}}{=} [x \cdot i]_1^\ell$ is a (m, ℓ, ε)-extractor for any $\ell \leq m - 2\log(1/\varepsilon)$. Here the multiplication $x \cdot i$ is performed in the field $\mathbb{F}_{2^n}$ and the notation $[z]_1^\ell$ denotes the first ℓ bits of z.

FUZZY EXTRACTORS. A fuzzy extractor extracts a uniformly random key from some secret w in such a way that the key can be recovered from any w' close to w. The notion was first defined in [10]. Here we alter the definition to allow for a public common reference string (CRS).

Definition 8. *An* $(m, \ell, t, \varepsilon)$-fuzzy extractor *for a metric space* $\mathcal{M}$ *is defined by randomized procedures* $(\mathsf{Init}, \mathsf{Gen}, \mathsf{Rep})$ *with the following properties:*

1. *The procedure* Init *takes no inputs and outputs a string* $\mathrm{CRS} \in \{0,1\}^*$.
2. *The generation procedure* Gen*, on input* $w \in \mathcal{M}, \mathrm{CRS} \in \{0,1\}^*$*, outputs an extracted string* $R \in \{0,1\}^\ell$ *and a helper string* $P \in \{0,1\}^*$*. The reproduction procedure* Rep *takes* $w' \in \mathcal{M}$ *and* $P, \mathrm{CRS} \in \{0,1\}^*$ *as inputs. It outputs* $\tilde{w} \in \mathcal{M} \cup \{\perp\}$.
3. Correctness: *If* $\mathsf{dis}(w, w') \leq t$ *and* $(R, P) \leftarrow \mathsf{Gen}(w, \mathrm{CRS})$*, then* $\mathsf{Rep}(w', P, \mathrm{CRS}) = R$.
4. Privacy: *For any distribution* W *with min-entropy* m *over the metric* $\mathcal{M}$ *, the string* R *is close to uniform even conditioned on the value of* P*. Formally, if* $\mathbf{H}_\infty(W) \geq m$ *and* $(R, P) \leftarrow \mathsf{Gen}(W, \mathrm{CRS})$*, then* $(R, P, \mathrm{CRS}) \approx_\varepsilon (U_\ell, P, \mathrm{CRS})$.

Composing an (m, m', t)-secure sketch with a (m', ℓ, ε)-extractor $\mathsf{Ext}\colon \mathcal{M} \times \{0,1\}^d \to \{0,1\}^\ell$ (as defined in Def. 7) yields a $(m, \ell, t, \varepsilon)$-fuzzy extractor [10]. The construction of [10] has an empty CRS and sets $P = (\mathsf{SS}(w), i)$ and $R = \mathsf{Ext}(w; i)$ for a random i. However, it is easy to see that the construction would remain secure if the extractor seed i was contained in the CRS and P was just $\mathsf{SS}(w)$. One advantage of such approach would be that the Gen and Rep algorithms are then deterministic which might make them easier to implement in hardware. Another advantage is that it would eventually allow us to overcome the impossibility barrier of robust fuzzy extractors (defined next) in the plain model.

5.1 Definition of Robust Fuzzy Extractor in CRS Model

Fuzzy extractors allow one to reveal P publicly without sacrificing the security of the extracted randomness R. However, there are no guarantees when an active attacker modifies P. To prevent such attacks, robust fuzzy extractors were defined and constructed in [5, 11]. Here we define robust fuzzy extractors in the CRS model.

For two (correlated) random variables W, W' over a metric space $\mathcal{M}$, we say $\mathsf{dis}(W, W') \leq t$ if the distance between W and W' is at most t with probability one. We call (W, W') a (t, m)-*correlated pair* if $\mathsf{dis}(W, W') \leq t$ and $\mathbf{H}_\infty(W) \geq m$. It will turn out that we can get more efficient constructions if we assume that the random variable $\Delta = W - W'$ indicating the errors between W and W' is independent of W (this was the only case considered by [5]). However, we do not want to make this assumption in general since it is often unlikely to hold. We define the family $\mathcal{F}^{all}_{t,m}$ to be the family of all (t, m)-correlated pairs (W, W') and the family $\mathcal{F}^{indep}_{t,m}$ to be the family of (t, m)-correlated pairs for which $\Delta = W - W'$ is independent of W.

Definition 9. *An* $(m, \ell, t, \varepsilon, \delta)$-robust fuzzy extractor *for a metric space* $\mathcal{M}$ *and a family* $\mathcal{F}$ *of* (t, m)-*correlated pairs is an* $(m, \ell, t, \varepsilon)$-*fuzzy extractor over* $\mathcal{M}$ *such that for all* $(W, W') \in \mathcal{F}$ *and all adversaries* $\mathcal{A}$

$$\Pr\left[\begin{array}{c} \mathsf{Rep}(\tilde{P}, w', \mathrm{CRS}) \neq \perp \\ \tilde{P} \neq P \end{array} \middle| \begin{array}{c} \mathrm{CRS} \leftarrow \mathsf{Init}(),\ \ (w, w') \leftarrow (W, W') \\ (P, R) \leftarrow \mathsf{Gen}(w, \mathrm{CRS}),\ \ \tilde{P} \leftarrow \mathcal{A}(P, R, \mathrm{CRS}) \end{array}\right] \leq \delta$$

We call the above notion post-application *robustness and it will serve as our main definition. We also consider a slightly weaker notion, called* pre-application *robustness where we do not give* R *to the adversary* $\mathcal{A}$.

The distinction between *pre*-application and *post*-application robustness was already made in [5, 11]. Intuitively, when a user Alice extracts a key using a robust fuzzy extractor, she may use this key for some purpose such that the adversary can (partially) learn the value of the key. The adversary can then mount an attack that modifies P based on this learned value. For post-application security, we insist that robustness is preserved even in this setting. For pre-application security, we assume that the adversary has no partial information about the value of the key.

5.2 Construction

We are now ready to construct robust fuzzy extractors in the CRS model. First, let us outline a general idea for the construction using an extractor Ext, a secure sketch $(\mathsf{SS}, \mathsf{Rec})$ and a one-time (information-theoretic) message authentication code MAC. A pictorial representation of the construction is shown in Figure 1 and pseudo-code is given below.

> $\mathsf{Init}()$ outputs a random seed i for the extractor Ext as a shared CRS.
> $\mathsf{Gen}(w, i)$ does the following:
> $R \leftarrow \mathsf{Ext}(w, i)$ which we parse as $R = (R_{mac}, R_{out})$.
> $s \leftarrow \mathsf{SS}(w)$, $\sigma \leftarrow \mathsf{MAC}(s, R_{mac})$, $P := (s, \sigma)$.
>   Output (P, R_{out}).
> $\mathsf{Rep}(w', \tilde{P}, i)$ does the following:
>   Parse $\tilde{P} = (\tilde{s}, \tilde{\sigma})$. Let $\tilde{w} \leftarrow \mathsf{Rec}(w', \tilde{s})$. If $d(\tilde{w}, w') > t$ then output $\perp$.
>   Using $\tilde{w}$ and i, compute $\tilde{R}$ and parse it as $\tilde{R}_{out}$, $\tilde{R}_{mac}$.
>   Verify $\tilde{\sigma} = \mathsf{MAC}(\tilde{s}, \tilde{R}_{mac})$. If equation holds output $\tilde{R}_{out}$, otherwise output $\perp$.

The idea is fairly intuitive. First, we extract randomness from w using the public extractor seed i. Then we use part of the extracted randomness R_{out} as the output, and the remaining part R_{mac} as the key for the one-time information-theoretic MAC to authenticate the secure sketch s of w.

However, in arguing robustness of the reconstruction phase, we notice that there is a problem. When an adversary modifies s to some value $\tilde{s}$ then this will force the user to incorrectly recover $\tilde{w} \neq w$, which in turn leads to the reconstruction of $\tilde{R} \neq R$ and $\tilde{R}_{mac} \neq R_{mac}$. So the key $\tilde{R}_{mac}$, which is used to verify the authenticity of s, will itself be modified when s is!

To break the circularity, we will need to us special linearity properties of the secure sketch and extractor constructions, which we specify in section 5.3. We will argue in that an adversary who modifies s to $\tilde{s}$ will know the offset Δ such that $\tilde{R}_{mac} = R_{mac} + \Delta$. Although $\tilde{R}_{mac}$ is derived from w', $\tilde{s}$ and the CRS, we can think of R_{mac} as being stored in an abstract device $\Sigma(\mathcal{G})$ which is private but only weakly robust in that the adversary can specify an additive offset by modifying s. We can then use a KMS-MAC to get security even when the key is stored on such a device. Hence, the adversary will not be able to come up with a valid pair $(\tilde{s}, \tilde{\sigma})$ where $\tilde{s} \neq s$.

5.3 Linearity of modifying P

In this section, we specify the properties of our secure sketch and extractor constructions to ensure that an adversary who knows $\Delta = w' - w$ and modifies s to $\tilde{s}$, will know the offset $R^{\tilde{\Delta}} = \tilde{R} - R$ between the original extracted key and the recovered key.

Secure Sketch Linearity Property: *Let* $(\mathsf{SS}, \mathsf{Rec})$ *be an* (m, m', t)*-secure-sketch and* w, w' *be values such that* $\mathsf{dis}(w, w') \leq t$. *Let* $\Delta = w' - w$ *and* $s = \mathsf{SS}(w)$. *For any* $\tilde{s}$, *let* $\tilde{w} := \mathsf{Rec}(w', \tilde{s})$ *and* $\tilde{\Delta} = \tilde{w} - w$. *Then,*

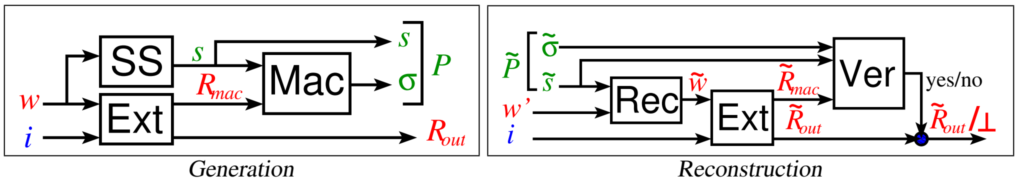

Fig. 1. Construction of Robust Fuzzy Extractor

we say that the secure sketch is linear if $\tilde{\Delta}$ is completely determined by Δ, s and $\tilde{s}$. Formally $\tilde{\Delta} = f(\Delta, s, \tilde{s})$ where f is a deterministic function.

Lemma 2. *The syndrome based construction of a secure sketch meets the above linearity property.*

This lemma follows easily from the properties of the syndrome construction and we give a proof in Appendix F. It was also implicitly used in [11].

Extractor Linearity Property: *The extractor* Ext *is linear if for any a, b and i, we have* $\mathsf{Ext}(a - b, i) = \mathsf{Ext}(a, i) - \mathsf{Ext}(b, i)$.

It is easy to see that the extractor defined by $\mathsf{Ext}(w, i) \stackrel{\text{def}}{=} [w \cdot i]_1^{\ell}$ has the required linearity property. We also notice that several other extractors (e.g., [29, 25]) with shorter seed lengths also satisfy this property. As it turns out, it is precisely this property of extractors, not useful in the plain model setting of [11], that would allow us to obtain the following key Lemma what we will use in the CRS model.

Lemma 3. *Assume a secure sketch* $(\mathsf{SS}, \mathsf{Rec})$ *and an extractor* Ext *meet the respective linearity properties above. Consider any $w, w', i, \tilde{s}$ and let $s = \mathsf{SS}(w)$, $R = \mathsf{Ext}(w, i)$, $\tilde{w} = \mathsf{Rec}(w', \tilde{s})$, $\tilde{R} = \mathsf{Ext}(\tilde{w}, i)$. Finally, denote $\Delta = w' - w$ and $R^{\tilde{\Delta}} = \tilde{R} - R$. Then, there is a deterministic function g such that $R^{\tilde{\Delta}} = g(\Delta, s, \tilde{s}, i)$. Namely, one can compute $R^{\tilde{\Delta}}$ by knowing only the difference Δ between w and w', the sketch s, the modified sketch $\tilde{s}$ and the public CRS i.*

Proof. Using Lemma 2, there is a deterministic function $f(\Delta, s, \tilde{s}) = \tilde{\Delta} = \tilde{w} - w$. If we let $g(\Delta, s, \tilde{s}, i) \stackrel{\text{def}}{=} \mathsf{Ext}(f(\Delta, s, \tilde{s}), i)$ then

$$\begin{aligned} g(\Delta, s, \tilde{s}, i) = \mathsf{Ext}(f(\Delta, s, \tilde{s}), i) = \mathsf{Ext}(\tilde{w} - w, i) &= \mathsf{Ext}(\tilde{w}, i) - \mathsf{Ext}(w, i) \\ &= \tilde{R} - R = R^{\tilde{\Delta}} \end{aligned}$$

□

5.4 Security of Construction and Parameters

We are now show that the construction outlined in Section 5.2 indeed satisfies the definition of a robust fuzzy extractor.

Let $(\mathsf{SS}, \mathsf{Rec})$ be a (m, m', t)-secure sketch satisfying the secure sketch linearity property and let u be an upper bound on the size of $\mathsf{SS}(w)$. Let MAC be a (S, G, T, δ)-KMS-MAC, such that $S \geq 2^u$. Assume that the keys come from a group $\mathcal{G} = \{0, 1\}^k$ under the XOR operation so that $G = 2^k$. Let $\mathcal{F}$ be a class of (t, m)-correlated variables (W, W') and let $\hat{m}$ be the largest value such that $\hat{m} \leq \widetilde{\mathbf{H}}_\infty(W | \mathsf{SS}(W), W - W')$ for any $(W, W') \in \mathcal{F}$. Lastly, let Ext be a $(\hat{m}, \ell, \varepsilon)$-strong randomness extractor satisfying the extractor linearity property and seeded by randomness i of length d.

Theorem 5. *When instantiated with the primitives* Ext, MAC *and* $(\mathsf{SS}, \mathsf{Rec})$, *our construction yields a* $(m, \ell - k, t, 2\varepsilon, \delta + \varepsilon)$*-robust-fuzzy extractor for the family* $\mathcal{F}$.

Proof. The correctness property of the fuzzy extractor is guaranteed by the correctness of the secure sketch. The privacy property follows from the security of the randomness extractor. Recall, that the adversary can observe i, s, σ. Since, by definition, $\hat{m} \leq \widetilde{\mathbf{H}}_\infty(W|\mathsf{SS}(W))$, the distribution (i, s, R_{mac}, R_{out}) can be distinguished from $(i, s, U_k, U_{\ell-k})$ with probability at most ε. In particular,

$$(i, s, R_{mac}, R_{out}) \approx_\varepsilon (i, s, U_k, U_{\ell-k}) \approx_\varepsilon (i, s, R_{mac}, U_{\ell-k})$$

and so $(i, s, R_{mac}, R_{out}) \approx_{2\varepsilon} (i, s, R_{mac}, U_{\ell-k})$ by the triangle inequality. An adversary given i, s, σ is weaker than an adversary given i, s, R_{mac} and even this latter adversary can distinguish R_{out} from $R_{\ell-k}$ with probability at most 2ε.

For robustness, consider any pair $(W, W') \in \mathcal{F}$ and any adversary $\mathcal{A}$ attacking the robustness of the scheme. Then

$$\begin{aligned}
\Pr[\mathcal{A} \text{ succeeds}] &= \Pr\left[\begin{array}{c|c} \mathsf{Rep}(\tilde{P}, w', \mathrm{CRS}) \neq \bot & \mathrm{CRS} \leftarrow \mathsf{Init}(),\ (w, w') \leftarrow (W, W') \\ \text{and } \tilde{P} \neq P & (P, R) \leftarrow \mathsf{Gen}(w, \mathrm{CRS}) \\ & \tilde{P} \leftarrow \mathcal{A}(\mathrm{CRS}, P, R) \end{array}\right] \\
&= \Pr\left[\begin{array}{c|c} & i \leftarrow U_d,\ (w, w') \leftarrow (W, W') \\ \mathsf{MAC}(\tilde{s}, \tilde{R}_{mac}) = \tilde{\sigma} & (R_{mac}, R_{out}) := \mathsf{Ext}(w, i) \\ & s := \mathsf{SS}(w),\ \sigma := \mathsf{MAC}(s, R_{mac}) \\ (\tilde{s}, \tilde{\sigma}) \neq (s, \sigma) & (\tilde{s}, \tilde{\sigma}) \leftarrow \mathcal{A}(i, s, \sigma, R_{out}) \\ & \tilde{w} := \mathsf{Rec}(w', \tilde{s}),\ (\tilde{R}_{mac}, \tilde{R}_{out}) := \mathsf{Ext}(\tilde{w}, i) \end{array}\right]
\end{aligned}$$

Now we use Lemma 3 which defines the deterministic function g such that

$$\Pr[\mathcal{A} \text{ succeeds}] = \Pr\left[\begin{array}{c|c} & i \leftarrow U_d,\ (w, w') \leftarrow (W, W') \\ \mathsf{MAC}(\tilde{s}, \tilde{R}_{mac}) = \tilde{\sigma} & (R_{mac}, R_{out}) := \mathsf{Ext}(w, i) \\ (\tilde{s}, \tilde{\sigma}) \neq (s, \sigma) & s := \mathsf{SS}(w),\ \sigma := \mathsf{MAC}(s, R_{mac}) \\ & (\tilde{s}, \tilde{\sigma}) \leftarrow \mathcal{A}(i, s, \sigma, R_{out}) \\ & \varDelta := w' - w,\ \tilde{R}_{mac} := R_{mac} + g(\varDelta, s, \tilde{s}, i) \end{array}\right]$$

On the right hand side of the inequality, the pair (w, w') and the value i determine the values $\varDelta, s, R_{mac}, R_{out}$. But the distributions $(\varDelta, s, i, R_{mac}, R_{out})$ and $(\varDelta, s, i, U_\ell)$ can be distinguished with probability at most ε, by the security of the extractor and the fact that $\hat{m} \leq \widetilde{\mathbf{H}}_\infty(W|\mathsf{SS}(W), \varDelta)$.

Hence we have:

$$\begin{aligned}
&\Pr[\mathcal{A} \text{ succeeds}] \\
&\leq \varepsilon + \Pr\left[\begin{array}{c|c} \mathsf{MAC}(\tilde{s}, \tilde{R}_{mac}) = \tilde{\sigma} & i \leftarrow U_d,\ R_{mac} \leftarrow U_k,\ (w, w') \leftarrow (W, W') \\ & s := \mathsf{SS}(w),\ \sigma := \mathsf{MAC}(s, R_{mac}) \\ (\tilde{s}, \tilde{\sigma}) \neq (s, \sigma) & (\tilde{s}, \tilde{\sigma}) \leftarrow \mathcal{A}(i, s, \sigma, U_{\ell-k}) \\ & \varDelta \leftarrow w' - w, \tilde{R}_{mac} := R_{mac} + g(\varDelta, s, \tilde{s}, i) \end{array}\right] \qquad (3) \\
&\leq \varepsilon \ + \max_{R^{\varDelta}_{mac}, \tilde{s} \neq s, \sigma, \tilde{\sigma}} \Pr\left[\begin{array}{c|c} & R_{mac} \leftarrow U_k \\ \mathsf{MAC}(\tilde{s}, \tilde{R}_{mac}) = \tilde{\sigma} & \sigma := \mathsf{MAC}(s, R_{mac}) \\ & \tilde{R}_{mac} := R_{mac} + R^{\varDelta}_{mac} \end{array}\right] \\
&\leq \varepsilon + \delta
\end{aligned}$$

Where the last inequality follows from the security of the KMS-MAC. □

The above theorem is stated with generality in mind. We now examine the parameters we get when plugging in the optimal implementation of a KMS-MAC and using the "multiplication" extractor $\mathsf{Ext}(x,i) \stackrel{\text{def}}{=} [x \cdot i]_1^v$.

Corollary 3. *Using given constructions of strong randomness extractors and KMS-MACs, we get a $(m, \ell, t, \varepsilon, \delta)$-robust fuzzy extractor for the family $\mathcal{F}$ and for any m, t, ε and $\delta \geq \varepsilon$. The extracted key length is*

$$\ell \approx \hat{m} - 2\log\left(\frac{2(u+3)}{\varepsilon(\delta-\varepsilon)}\right) - 2$$

Recall, that u is the length of the secure sketch, n is the length of the secret w, and $\hat{m} \leq \widetilde{\mathbf{H}}_\infty(W|\mathsf{SS}(W), W-W')$ for any $(W, W') \in \mathcal{F}$.

Moreover, for the family $\mathcal{F}^{all}_{(t,m)}$ of all (t,m) correlated pairs,

$$\hat{m} \geq m - u - t\left(\log\left(\frac{n}{t}\right) + \log e\right)$$

For the family $\mathcal{F}^{indep}_{(t,m)}$ of all (t,m)-correlated pairs for which $\Delta = W - W'$ and W are independent $\hat{m} = m' \geq m - u$.

Proof. The strong randomness extractor construction we looked at previously, extracts (R_{mac}, R_{out}) of length $\hat{m} - 2\log(1/\varepsilon')$ to achieve security ε'. We want $\varepsilon' = \varepsilon/2$. This implies $\ell \approx \hat{m} - 2\log(2/\varepsilon) - k$ where k is the size of R_{mac}. By the bounds on key-lengths of the KSM-MAC construction given in 2, if we want to get security $\delta - \varepsilon$ and authenticate messages of length u, we can use a key of length $k \leq 2\log(1/(\delta-\varepsilon)) + 2\log(u+3) + 2$ Putting these together we see

$$\ell \geq \hat{m} - 2[\log(2/\varepsilon) + \log(1/(\delta-\varepsilon)) + \log(u+3)] - 2 \geq \hat{m} - 2\log\left(\frac{2(u+3)}{\varepsilon(\delta-\varepsilon)}\right) - 2$$

This proves the first part of the corollary. To bound $\hat{m}$, we notice $\widetilde{\mathbf{H}}_\infty(W|\mathsf{SS}(W), W-W') \geq \mathbf{H}_\infty(W) - \lambda$ where 2^λ is the number of possible values of the pair $\mathsf{SS}(W), W-W'$. The number of possible values of $\mathsf{SS}(W)$ is 2^u, since u is a bound on the size of $\mathsf{SS}(W)$. The number of possible values of $\Delta = W' - W$ of a (t,m) correlated pair (W, W') is the volume of the ball of elements of length n that are at a distance t from each other. The log of this volume is derived in [11] and is $t\left(\log\left(\frac{n}{t}\right) + \log e\right)$. This gets us the first bound on $\hat{m}$. When Δ and W are independent then $\widetilde{\mathbf{H}}_\infty(W|\mathsf{SS}(W), W-W') = \widetilde{\mathbf{H}}_\infty(W|\mathsf{SS}(W)) = m' \geq \mathbf{H}_\infty(W) - u$ which derives the second bound. □

So far, all of our bounds are for post-application robustness. We now show that for pre-application robustness the bounds for the families $\mathcal{F}^{all}_{(t,m)}$ and $\mathcal{F}^{indep}_{(t,m)}$ are essentially equivalent. This is because, for pre-application robustness, the adversary does not get to see R_{out} when mounting a key-manipulation attack. Hence, for robustness, we no longer need to ensure that there is enough residual min entropy left over in w after the adversary sees Δ and s to extract R_{out} as well as R_{mac}.

Corollary 4. *For pre-application robustness only, we get a $(m, \ell, t, \varepsilon, \delta)$-robust fuzzy extractor for any (t,m)-correlated family $\mathcal{F}$ and for any m, t, ε and $\delta \geq \varepsilon$ with*

$$\ell \approx m' - 2\log\left(\frac{2(u+3)}{\varepsilon(\delta-\varepsilon)}\right) - 2$$

as long as

$$\hat{m} \geq 2\log\left(\frac{u+3}{\varepsilon(\delta-\varepsilon)}\right) - 2$$

Proof. The first condition on the size of the extracted key follows from Theorem 5 and bounds on the length k of the KMS-MAC key. In Theorem 5, inequality 3, for pre-application robustness the adversary does not get R_{out}. This means that inequality 3 holds as long as $(\mathit{\Delta}, s, i, R_{mac})$ and $(\mathit{\Delta}, s, i, U_k)$ can be distinguished with probability at most ε. Now we notice that the extractor $\mathsf{Ext}(w,i) \stackrel{\text{def}}{=} [w \cdot i]_1^v$ has the property that the first $\mathbf{H}_\infty(W) - 2\log(1/\varepsilon)$ bits of $\mathsf{Ext}(w,i)$ are ε close to random no matter how large v is. This means we only need R_{mac} to be indistinguishable in this case, and hence we have the weaker condition $k \leq \hat{m} - 2\log(1/\varepsilon)$ rather than $\ell \leq \hat{m} - 2\log(1/\varepsilon)$ in Theorem 5. Substituting the bounds on k we get

$$2\log((u+3)/(\delta-\varepsilon)) + 2 \leq \hat{m} - 2\log(1/\varepsilon)$$

which derives the condition stated in the corollary. This condition is very weak and likely to be satisfied in practice. Hence, for pre-application robustness, we can essentially ignore the fact that $\mathit{\Delta}$ and W might not be independent. □

COMPARISON WITH PREVIOUS CONSTRUCTIONS: Recall that the "non-robust" construction of [10] extracts $\ell \leq m' - 2\log\left(\frac{1}{\varepsilon}\right)$ bits. On the other hand, the robust construction of [11] requires:

$$\ell \leq \frac{1}{3}\left(2m - n - u - 2t\log\left(\frac{en}{t}\right) - 2\log\left(\frac{n}{\varepsilon^2\delta}\right)\right) - O(1)$$

The bounds achieved in this paper are significantly closer to the non-robust version. In essence we show that the price of robustness can be cheap if we allow random public system parameters.

5.5 Extension to Other Metrics

We note that the above construction can be extended for other metric spaces and secure sketches. For example, we can easily extend our discussion of the hamming distance over a binary alphabet to an alphabet of size q where $\mathbb{F}_q$ is a field. The secure sketch simply uses an error correcting code for $\mathbb{F}_q$ (possibly even allowing us to use the optimal Reed-Solomon codes if $q \geq n$). For the extractor we work over the field $\mathbb{F}_{q^n}$ and the truncation function $[x]_1^\ell$ is defined as truncating symbols of $\mathbb{F}$ (where elements of $\mathbb{F}_{q^n}$ are viewed as n dimensional vectors over $\mathbb{F}_q$) rather than bits.

Finally, we note that our construction extends to the set difference metric in exactly the same way as the construction of [11].

References

1. C. H. Bennett, G. Brassard, C. Crépeau, and M.-H. Skubiszewska. Practical quantum oblivious transfer. In J. Feigenbaum, editor, *Advances in Cryptology—CRYPTO '91*, volume 576 of *LNCS*, pages 351–366. Springer-Verlag, 1992, 11–15 Aug. 1991.
2. G. Brassard, A. Broadbent, Jo. Fitzsimons, S. Gambs, and A. Tapp. Anonymous quantum communication. Available at `http://arxiv.org/abs/0706.2356`. To be published in *ASIACRYPT '07*.
3. A. Broadbent, A. Tapp. Information-theoretic security without an honest majority. Available at `http://arxiv.org/abs/0706.2010`. To be published in *ASIACRYPT '07*.
4. C. Blundo and A. De Santis. Lower bounds for robust secret sharing schemes. *Information Processing Letters*, 63(6), 1997.
5. X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith. Secure remote authentication using biometric data. In R. Cramer, editor, *Advances in Cryptology—EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 147–163. Springer-Verlag, 2005.
6. X. Boyen. Reusable cryptographic fuzzy extractors. In *11th ACM Conference on Computer and Communication Security*. ACM, Oct. 25–29 2004.
7. S. Cabello, C. Padró, and G. Sáez. Secret sharing schemes with detection of cheaters for a general access structure. Designs, Codes and Cryptography 25 (2002) 175-188. Earlier version in Proceedings *12th International Symposium on Fundamentals of Computation Theory (FCT)*, volume 1233 of *Lecture Notes in Computer Science*. Springer, 1999.
8. R. Cramer, I. B. Damgård, and S. Fehr. On the cost of reconstructing a secret, or VSS with optimal reconstruction phase. In *Advances in Cryptology—CRYPTO '01*, volume 2139 of *Lecture Notes in Computer Science*. Springer, 2001.
9. Y. Dodis Exposure Resillient Cryptography. Ph.D. Thesis, MIT 2000
10. Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. Technical Report 2003/235, Cryptology ePrint archive, `http://eprint.iacr.org`, 2006. Previous version appeared at *EUROCRYPT 2004*.
11. Y. Dodis and J. Katz and L. Reyzin and A. Smith. Robust Fuzzy Extractors and Authenticated Key Agreement from Close Secrets. In *Advances in Cryptology—CRYPTO '06*, volume 4117 of *Lecture Notes in Computer Science*. Springer, 2006.
12. Y. Dodis and J. Spencer. On the (non-)universality of the one-time pad. In *43rd Annual Symposium on Foundations of Computer Science*, pages 376–385. IEEE, 2002.
13. Y. Desmedt and Y. Wang. Perfectly secure message transmission revisited. In *Advances in Cryptology—EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*. Springer, 1992.
14. D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *Journal of the ACM*, 40(1), 1993.
15. T. Johansson, G. Kabatianskii, and B. Smeets. On the relation between A-codes and codes correcting independent errors. In *Advances in Cryptology—EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*. Springer, 1993.
16. M. Karchmer and A. Wigderson. On span programs. In *8th Annual Conference on Structure in Complexity Theory (SCTC '93)*. IEEE, 1993.
17. H. Krawczyk. Distributed fingerprints and secure information dispersal. In *12th ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, 1993.
18. N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–53, 1996.
19. S. Obana and T. Araki. Almost Optimum Secret Sharing Schemes Secure Against Cheating for Arbitrary Secret Distribution. In *ASIACRYPT '06*, volume 4284 of *Lecture Notes in Computer Science*. Springer, 2006.
20. W. Ogata and K. Kurosawa. Optimum secret sharing scheme secure against cheating. In *Advances in Cryptology—EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*. Springer, 1996.
21. W. Ogata, K. Kurosawa, D.R. Stinson, and H. Saido. New combinatorial designs and their applications to authentication codes and secret sharing schemes. Discrete Mathematics 279 (2004), 383-405.
22. C. Padró, G. Sáez, J.L. Villar. Detection of cheaters in vector space secret sharing schemes. Designs, Codes and Cryptography 16 (1999) 75-85.
23. C. Padró. Robust vector space secret sharing schemes. Information Processing Letters 68 (1998) 107-111.
24. M. O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, 36(2), 1989.
25. R. Raz, O. Reingold, S. Vadhan. Extracting all the Randomness and Reducing the Error in Trevisan's Extractors. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing — STOC '99*. ACM Press 1999.
26. A. Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11), 1979.
27. G.J. Simmons. Authentication theory/Coding Theory. In *Advances in Cryptology—CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*. Springer, 1985.
28. M. Tompa and H. Woll. How to share a secret with cheaters. *Journal of Cryptology*, 1(3), 1988.
29. Luca Trevisan. Extractors and Pseudorandom Generators. *J. of the ACM*, 48(4):860-879, 2001.

A Lower Bounds

Theorem 6. *Any weak, respectively regular (strong), (S, G, δ)-AMD code satisfies*

$$G \geq \frac{S-1}{\delta} + 1 \qquad \textit{respectively} \qquad G \geq \frac{S-1}{\delta^2} + 1\,.$$

Let $(\mathcal{E}, D)$ be an (S, G, δ)-AMD code, with $\mathcal{E} : \mathcal{S} \to \mathcal{G}$ and $D : \mathcal{G} \to \mathcal{S} \times \{\perp\}$. For any $s \in \mathcal{S}$, consider the set $D^{-1}(s) = \{e \in \mathcal{G} : D(e) = s\}$. Clearly, $D^{-1}(s) \cap D^{-1}(s') = \emptyset$ and $|D^{-1}(s)| \geq 1$ for any $s \neq s'$.

Consider first the case where the AMD code is *weakly* secure. Let s be uniformly distributed over $\mathcal{S}$. Sample $\varDelta \neq 0$ at random from $\mathcal{G}$, independently of s. The probability that $D(\mathcal{E}(s) + \varDelta) \neq \{s, \perp\}$ is upper bounded by δ. This implies that

$$\delta \geq \Pr\left[\mathcal{E}(s) + \varDelta \in \textstyle\bigcup_{s' \neq s} D^{-1}(s')\right] = \frac{\left|\bigcup_{s' \neq s} D^{-1}(s')\right|}{G-1} \geq \frac{S-1}{G-1}$$

where the first inequality follows by considering $\varDelta$ fixed, and the first equality follows by considering s fixed, and realizing that if the (in)equality holds for any fixed value then it also holds for a random value.

Consider now the case where the AMD code is *strongly* secure. Then, for any $s \in \mathcal{S}$, it holds that $|D^{-1}(s)| \geq 1/\delta$. This follows from the fact that if one guesses $\mathcal{E}(s)$ correctly (knowing s) then it is easy to come up with a $\varDelta$ such that $D(\mathcal{E}(s) + \varDelta)) \notin \{s, \perp\}$. Similar to above, it hence follows that

$$\delta \geq \Pr\left[\mathcal{E}(s) + \varDelta \in \textstyle\bigcup_{s' \neq s} D^{-1}(s')\right] = \frac{\left|\bigcup_{s' \neq s} D^{-1}(s')\right|}{G-1} \geq \frac{(S-1)/\delta}{G-1}$$

which implies the claimed bound. Note that here the probability is taken over a random $\varDelta$ and over the randomness used by the encoding function $\mathcal{E}$ for a given s. □

Note that similar bounds were found in [20] for robust secret sharing schemes. This is no coincidence, since we show in the paper that AMD codes can be used to construct robust secret sharing schemes. The following bounds on the tag size now follow quite easily. It shows that it is unavoidable that the message grows by κ respectively 2κ bits if one wants to have weak respectively strong $2^{-\kappa}$-security.

Corollary 5. *The effective tag size of a weak, respectively strong, AMD code is lower bounded by*

$$\varpi^*(\kappa, u) \geq \kappa - 2^{-u+1} \geq \kappa - 1 \qquad \textit{respectively} \qquad \varpi^*(\kappa, u) \geq 2\kappa - 2^{-u+1} \geq 2\kappa - 1$$

Proof. For any weak (S, G, δ)-AMD code with with $S \geq 2^u$ and $\delta \leq 2^{-\kappa}$

$$\log(G) - u \geq \log(G) - \log(S) \geq \log\left(\frac{G-1}{S-1}\frac{S-1}{S}\right) = \log\left(\frac{G-1}{S-1}\right) + \log\left(1 - \frac{1}{S}\right) \geq \kappa - \frac{2}{S}$$

where the last inequality follows from Theorem 6 and the bound $\log(1-x) \geq -2x$ for $0 \leq x \leq \frac{1}{2}$.[5] Similarly, for a strongly secure AMD code, the argument proceeds analogously but the last inequality is replaced by $\log\left(\frac{G-1}{S-1}\right) + \log\left(1 - \frac{1}{S}\right) \geq 2\kappa - \frac{2}{S}$. □

[5] The bound follows from the fact that the two sides coincide when evaluated at $x = 0$ and at $x = \frac{1}{2}$, and that $-2x$ has constant slope whereas $\log(1-x)$ has strictly decreasing slope (as can be seen from its second derivative) i.e. makes a "right turn".

B An Insecure AMD code

Consider the systematic AMD code $\mathcal{E} : \mathbb{F}^d \to \mathbb{F}^d \times \mathbb{F} \times \mathbb{F}$, $s \to (s, x, f(x, s))$ with $x \in_R \mathbb{F}$, where $f(x, s) = s_1 x + \cdots + s_d x^d$. This AMD code, respectively the resulting robust secret sharing scheme, was proposed and "proven" to be secure in [19]. However, it is easy to see that this AMD code is *not* secure. This can easily be seen by observing that

$$f(x + \Delta_x, s) = \sum_{i=1}^{d} s_i (x + \Delta_x)^i = \sum_{i=1}^{d} s'_i x^i + \Delta_f = f(x, s') + \Delta_f$$

for some $s' = (s_1, \ldots, s_d)$ and some $\Delta_f \in \mathbb{F}$, where both s' and Δ_f can (efficiently) be computed when given s and Δ_x. Recall that when considering *strong* security, the adversary is assumed to know s. Hence, adding Δ_x to x, Δ_f to $f(x, s)$, and replacing s by s' allows the adversary to break the AMD code with probability 1. The "proof" given in [19] is very complicated, and thus it is difficult to point to what exactly was argued incorrectly. We note that this mistake was later noted and fixed by the authors independently of our work. However, we feel that the error nicely highlights the advantage of the abstract notion of an AMD code: it allows for a much simpler (in the above case we may even say *trivial*) analysis than, for instance, when considering fully-fletched robust secret sharing schemes.

C The Combinatorics of AMD Codes

C.1 Weakly Secure AMD Codes

Let $\mathcal{G}$ be a group of finite order G.

Definition 10. *A subset $V \subseteq \mathcal{G}$ of size S is a (S, G, t)*-bounded difference set *if the list of differences $v_i - v_j$, where $v_i, v_j \in V$, contains every non-zero element of $\mathcal{G}$* at most *t times.*

Note that the standard notion of a difference set requires the list of differences to contain every non-zero element *exactly* t times. We call an AMD code $(\mathcal{E}, D)$ *deterministic* if the (in general probabilistic) mapping $\mathcal{E}$ is deterministic. The following equivalence holds.

Theorem 7. *If $V \subset \mathcal{G}$ is a (S, G, t)-bounded difference set then the AMD-code*

$$\mathcal{E} : V \to \mathcal{G},\ s \mapsto s\,, \qquad \textit{and} \qquad D(s) = \begin{cases} s & \textit{if } s \in V \\ \perp & \textit{otherwise} \end{cases}$$

is a (deterministic) weakly secure (S, G, δ)-AMD code with $\delta = t/S$. And, vice versa, for an arbitrary deterministic *weak (m, n, δ)-AMD code $(\mathcal{E}, D)$, the subset $V = \mathcal{E}(\mathcal{S}) = \{\mathcal{E}(s) : s \in \mathcal{S}\} \subset \mathcal{G}$ is a (S, G, t)-bounded difference set with $t = \delta S$.*

Proof. It is clear that $(\mathcal{E}, D)$ as constructed is a weak (S, G, δ)-AMD code. It remains to argue the value of δ. By the property of V, for every non-zero $\Delta \in \mathcal{G}$, there exist at most t elements $s \in V$ such that $s + \Delta \in V$. For a uniformly distributes $s \in V$, and for Δ chosen independent of s, this means that $s + \Delta \in V$ holds at most with probability t/S. The other implication is argued similarly. □

C.2 Strongly Secure AMD Codes

Let $\mathcal{G}$ be a finite group of order G. Let $\mathcal{S}$ be a finite set of cardinality S. For simplicity write $\mathcal{S} = \{1, \ldots, S\}$. Let $V_1, \ldots, V_S$ be disjoint non-empty subsets of $\mathcal{G}$.

Definition 11. *We call* $(\mathcal{G}, V_1, \ldots, V_S)$ *a* differential structure.

The parameters of interest related to a differential structure are as follows. For any i we write t_i for the maximal overlap between any translation of V_i and the union of the other V_j's:

$$t_i = \max_{\Delta \in \mathcal{G}} \left| (V_i + \Delta) \cap \bigcup_{j \neq i} V_j \right| .$$

For a given differential structure $(\mathcal{G}, V_1, \ldots, V_S)$ consider the following AMD code.

$$\mathcal{E} : \{1, \ldots, S\} \to \mathcal{G},\ s \mapsto \tilde{s}$$

with

$$\tilde{s} \in_R V_s ,$$

i.e., $\tilde{s}$ is chosen with uniform distribution on V_s and independently of anything else, and

$$D(\tilde{s}) = \begin{cases} s & \text{if } \exists\, s : \tilde{s} \in V_s \\ \perp & \text{otherwise} . \end{cases}$$

This AMD code is *with uniform selection* in that for every $s \in \mathcal{S}$, the encoding $\mathcal{E}(s)$ is uniformly distributed over $D^{-1}(s) = \{e \in \mathcal{G} : D(e) = s\}$. All natural AMD codes we are aware of are with uniform selection.

Theorem 8. *If* $(\mathcal{G}, V_1, \ldots, V_S)$ *is a differential structure with parameters* $t_1, \ldots, t_S$*, then the above code* $(\mathcal{E}, D)$ *is a (strong)* (S, G, δ)*-AMD code (with uniform selection) where* $\delta = \max_i t_i/|V_i|$*. And, vice versa, for any* (S, G, δ)*-AMD code* with uniform selection*, the sets* $V_s = D^{-1}(s)$ *for* $s \in \mathcal{S}$ *form a differential structure where* $t_s \leq \delta |V_s|$.

Proof. Let s be an arbitrary fixed source. Let $\tilde{s}$ be its probabilistic encoding, uniformly distributed in V_s, and let Δ be the difference added to $\tilde{s}$ by the adversary, independent of $\tilde{s}$. Then, $\tilde{s} + \Delta$ is uniformly distributed in $V_s + \Delta$, and thus the probability that it lies in a V_j with $j \neq s$ is at most $t_s/|V_s|$. The other implication is argued similarly. □

An AMD code is *systematic* if the source set $\mathcal{S}$ is a group and the encoding is of the form

$$\mathcal{E} : \mathcal{S} \to \mathcal{S} \times \mathcal{G}_1 \times \mathcal{G}_2,\ s \mapsto (s, x, f(x, s))$$

for some function f, and where $x \in_R \mathcal{G}_1$. All our new constructions are systematic, and thus in particular with uniform selection. The decoding function of a systematic AMD code is naturally given by

$$D(s, x, e) = \begin{cases} s & \text{if } e = f(x, s) \\ \perp & \text{otherwise} \end{cases}$$

and we usually leave it implicit. The following lemma is trivial.

Lemma 4. *For a systematic AMD code, the underlying differential structure* $(\mathcal{G}, V_1, \ldots, V_S)$*, we have* $t_i = \max_{\substack{\Delta\in\mathcal{G} \\ j\neq i}} |(V_i+\Delta) \cap V_j|$ *for* $i = 1 \ldots S$*.*

Our results above can be viewed as supporting the view that combinatorics that is "ugly and non-smooth or non-symmetric" from a combinatorics point of view may sometimes lead to "stronger cryptography". Indeed, by requiring only certain relevant bounds on the parameters of a combinatorial construct with cryptographic relevance (like the *bounded* compared to the ordinary "*strict*" notion of a difference set), a much wider class of mathematical approaches to its construction may become available. Note that there are other areas in cryptography that have seen this phenomenon as well, e.g., authentication codes.

C.3 Relation to Earlier Work

Our combinatorial approach must be discussed with respect to earlier work by Ogata and Kurosawa [20] and Ogata, Kurosawa, Stinson and Saido [21]. In [20] the idea of using the classical notion of planar difference sets is introduced, and applications to (in our terminology) weakly secure AMD codes are given. The construction is based on the following AMD code. Let q be a prime so that $p = q^2 + q + 1$ is a prime as well, and let $B \subset \{0, \ldots, p-1\}$ be a *planar difference set* of size $q+1$. This means that the $(q+1)q = p-1$ pairwise differences modulo p of the elements in B are exactly the numbers $1, \ldots, p-1$. It is known that such a difference set exists (see e.g. [20] and the references therein). Then, $\mathcal{E} : B \to \mathbb{Z}_p$, $s \mapsto s$ is a weak $(q+1, q^2+q+1, 1/(q+1))$-AMD code. The tag size equals $\varpi = \log(q^2+q+1) - \log(q+1)$, which lies between $\log(q)$ and $\log(q+1)$. See also [21] for a more general approach. As before, the error probability is determined by the source space and hence the approach is not flexible.

Motivated by this, the above approach is extended in [21] to using *external difference families* (EDF), as introduced there. A (G, c, λ) S-EDF consists consists of a group $\mathcal{G}$ of order G and S disjoint non-empty subsets $V_1, \ldots, V_S$, each of size c, such that every non-zero element of $\mathcal{G}$ occurs *exactly* λ *times* as the difference between some v_i and some v_j where v_i and v_j come from different sets V_i and V_j, respectively. This abstract notion of an EDF (with $\lambda = 1$) leads to a weakly secure AMD code with a minimal tag size for a source space of size S and with $\delta = \frac{1}{cS}$. However, no general construction has been proposed to design EDF's, and thus it is not clear how fruitful this approach is, and in particular how good it is with respect to the *effective* tag size, i.e., when u and κ are given and a weakly secure (S, G, δ)-AMD code needs to be found with $m \geq 2^u$ and $\delta \leq 2^{-\kappa}$. Furthermore, we feel that the case that is more important for practice is the case where the size of the source space is *larger* than the inverse of the allowed error probability.

As to *strongly* secure AMD codes, with this notion of a (G, c, λ) m-EDF one could at best guarantee an error of at most $\frac{\lambda}{c}$, since it seems that one cannot rule out that there is a $\Delta \in \mathcal{G}$ and a V_i such that the intersection between $V_i + \Delta$ and some other V_j has cardinality λ.

In conclusion, our notion of differential structures, though somewhat related to external difference families, captures exactly the case of strongly secure AMD codes and it also paves the way for a wider class of mathematical constructions due to its relaxed conditions.

D From *Weak* AMD Codes to *Strong* AMD Codes

We show how to construct a strong AMD code from any weak AMD code and a (standard) message authentication code MAC. Consider a systematic[6] message authentication code $A : \mathcal{S} \times \mathcal{K} \to \mathcal{T}$ where we may

[6] The restriction to *systematic* codes is not crucial, but it allows to simplify the exposition.

assume, without loss of generality, that $\mathcal{S}$ and $\mathcal{T}$ are groups (e.g. sets of bitstrings of a given length with xor). In the standard setting, such a code is used to authenticate a source $s \in \mathcal{S}$ by appending the tag $\sigma = A(k, s)$ with a randomly sampled secret key $k \in \mathcal{K}$ (known to sender and receiver); integrity of a (possibly modified) pair $(\tilde{s}, \tilde{\sigma})$ is then checked by verifying if $\tilde{\sigma} = A(k, \tilde{s})$ holds. Let p_S be the success probability of the substitution attack, i.e., the maximum over all $s \neq s' \in \mathcal{S}$ of the probability of successfully substituting the authenticated s by s'.[7] Furthermore, let $\mathcal{E}' : \mathcal{S}' \to \mathcal{G}'$ be a *weakly* secure (S', G', δ')-AMD code with $\mathcal{S}' = \mathcal{K}$. Consider the following AMD code.

$$\mathcal{E} : \mathcal{S} \to \mathcal{S} \times \mathcal{G}' \times \mathcal{T},\ s \mapsto \big(s, E'(k), A(k, s)\big)\,.$$

for $k \in_R \mathcal{K}$. The decoding function D is obvious: $D(s, e', \sigma)$ outputs s if and only if $D'(e') \neq \perp$ and $\sigma = A(D'(e'), s)$.

Theorem 9. *The code $\mathcal{E}$ is a (S, G, δ)-AMD code with $S = |\mathcal{S}|$, $G = |\mathcal{S}||\mathcal{G}'||\mathcal{T}|$ and $\delta = \delta' + p_S$. If the underlying AMD code $\mathcal{E}'$ is systematic, then $\delta = \max\{\delta', p_S\}$.*

Proof. Obviously, the sizes of the domain and range of $\mathcal{E}$ are as claimed. It remains to determine δ. Fix an arbitrary $s \in \mathcal{S}$, and an arbitrary translation $\Delta = (\Delta_s, \Delta_{e'}, \Delta_\sigma) \in \mathcal{S} \times \mathcal{G}' \times \mathcal{T}$ with $\Delta_s \neq 0$. Let $e = (s, e', \sigma) = \mathcal{E}(s) = (s, \mathcal{E}'(k), A(k, s))$ for a random k. By assumption on $\mathcal{E}'$, the probability that $D'(e' + \Delta_{e'}) \notin \{k, \perp\}$ is at most δ'. Furthermore, by assumption on the authentication code, the probability that $\sigma + \Delta_\sigma = A(k, s + \Delta_s)$ is at most p_S. It follows that $D(e) = s + \Delta_s$ with probability at most $\delta' + p_S$.

In case of a systematic $\mathcal{E}'$, the encoding e' has k as first component, and we can make a case distinction of whether the corresponding first component Δ_k of $\Delta_{e'}$ is zero or not: if $\Delta_k \neq 0$ then $D'(e') = \perp$ except with probability δ', and if $\Delta_k = 0$ then $\sigma + \Delta_\sigma \neq A(k, s + \Delta_s)$ except with probability p_S. □

We now show that this approach is still doomed to give a sub-optimal AMD code with an effective tag size separated from the lower bound by essentially 2κ.

Proposition 1. *For any strongly secure AMD code obtained via Theorem 9, the effective tag size satisfies $\varpi^*(\kappa, u) \geq 4\kappa - 2^{-2\kappa+1}$.*

Proof (of Proposition 1). In order to achieve an error probability $\delta \leq 2^{-\kappa}$, by Lemma 1, the tag σ must be of bit-size at least κ and the key k of at least 2κ. But then, by Corollary 5, the elements in $\mathcal{G}'$ must be of bit-size at least $3\kappa - 2^{-2\kappa+1}$ (namely 2κ bits for the source k plus $\kappa - 2^{-2\kappa+1}$ for the tag size of $\mathcal{E}'$). This adds up to the claimed bound. □

E Syndrome Based Construction of Secure Sketch

For completeness, we review the secure sketch construction below.

Recall that an efficiently decodable $[n, k, 2t+1]$-error-correcting (binary) code C over $\{0,1\}^n$ consists of 2^k codewords $C = \{z \mid Hz = 0\}$, where H is the $(n-k) \times n$ *parity check matrix* of C (addition and multiplication over $GF(2)$). Namely, H defines $(n-k)$ linear constraints which are satisfied precisely by the codewords in C. Moreover, H is chosen in such a way that the Hamming distance between any two distinct codewords $z_1, z_2 \in C$ is at least $2t+1$ (recall, the Hamming distance between $a, b \in \{0,1\}^n$ is the

[7] We would like to point out that there is some ambiguity in how p_S may be precisely defined, with regard to the attacker's control over the source s to be substituted and over the source s' with which he substitutes s. The definition used here, which controls the *worst case*, is necessary for our application.

number of symbols i such that $a_i \neq b_i$). This means, in principle, that any codeword $z \in C$ can be recovered from any "corrupted" string z' within Hamming distance at most t from z. In an efficiently decodable code C, this procedure of recovering z from z' can be done efficiently.

As it turns out, for our purposes we will only need to know the following well known fact about such efficiently decodable $[n, k, 2t+1]$-codes: if $z \in C$ and $\mathsf{dis}(z, z') \leq t$, then there is an efficient procedure Decode that can determine the "error vector" $z' - z$ from the $(n-k)$-bit quantity Hz'. This quantity Hz' is also called the *syndrome of* z' and denoted $\mathsf{syn}(z')$.

Coming back to the syndrome construction of the secure sketches from [10], the sketch $s = \mathsf{SS}(w)$ of $w \in \{0,1\}^n$ consists of the k-bit syndrome of w with respect to some (efficiently decodable) $[n, n-k, 2t+1]$-error-correcting code C: $\mathsf{SS}(w) = \mathsf{syn}(w) = s$. Notice, s is a (deterministic) *linear function* of w, and that the entropy loss of this construction is at most $|s| = n-k$. To see the correctness of this cosntruction, we notice that the recovery function Rec of w from the sketch s and any w' of Hamming distance at most t from w is computed as follows:

$$\mathsf{Rec}(w', s) = w' - \mathsf{Decode}(\mathsf{syn}(w') - s)$$

We should also note that this construction extends to the set difference metric through sublinear-time encoding and decoding [10] .

F Proof of Lemma 2

Recall that the secure sketch for hamming distance is given by two function syn, Decode

$\mathsf{SS}(w) = \mathsf{syn}(w) = s$
$\mathsf{Rec}(w', s) = w' - \mathsf{Decode}(\mathsf{syn}(w') - s)$

and that syn is linear. Hence

$$\begin{aligned}\tilde{\Delta} &= \tilde{w} - w = \mathsf{Rec}(w', \tilde{s}) - w \\ &= w' - \mathsf{Decode}(\mathsf{syn}(w') - \tilde{s}) - w \\ &= \Delta - \mathsf{Decode}(\mathsf{syn}(w + \Delta) - \tilde{s}) \\ &= \Delta - \mathsf{Decode}(s + \mathsf{syn}(\Delta) - \tilde{s}) \\ &= f(\Delta, s, \tilde{s})\end{aligned}$$

where f is deterministic.