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

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 $\Delta$ 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. 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. 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 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 $\Delta$ 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.

## 1.1 Linear Secret Sharing Schemes

In a *linear secret sharing scheme* (e.g. Shamir's secret sharing [24] 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 $\delta$. 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* [26], 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

[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.

longer than the length of $s$. This construction is close to optimal since we prove a lower bound of $2\kappa$ 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 [19,6,20]. Although some of these constructions are essentially optimal, all of them are largely inflexible in that the error probability $\delta$ 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 $\delta$ and message length $u$.

For example, Cabello, Padró and Sáez [6] (see also [22,21]) 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 $\delta$. 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.

In the context of robust secret sharing, the inflexibility issue mentioned above has recently been addressed in a paper by Obana and Araki [18], where a *flexible* robust secret sharing scheme (in fact, an AMD code in our terminology) was proposed and claimed to be "proven" secure. However, in the full version of this paper [8], we give an attack on their construction showing it to be 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 $P$. In their original paper, Dodis et al. [10] constructed fuzzy extractors for the Hamming and several other metrics. 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. [5,4], this severely limits the usability of the concept. To address this problem, [5,4] 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.

[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.

**Prior Work and Our Result.** All of the prior work on robust fuzzy extractors uses some form of a *message authentication code (MAC)* keyed by $w$ to authenticate the public parameters $P$. Such codes are difficult to construct since $w$ is not a uniformly random secret, and the authentication information needs to be verifiable using an imperfect reading $w'$ of $w$.

Nevertheless, Boyen et al. [4] gave a generic transformation which makes a fuzzy extractor robust *in the random oracle model*, without considerably sacrificing any of the parameters. In the plain model, Dodis et al. [11] showed how to achieve robustness 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). The work of [12] shows that this requirement is necessary for information theoretic security in the plain model, even if no errors are allowed (i.e., $w = w'$). Moreover, when the secret does meet this entropy rate threshold, robustness in [11] 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 take a difference approach and use a portion of the *extracted randomness* $R$ to authenticate the public parameters $P$. Of course, using a MAC naively is insecure since the adversary who modifies $P$ to $\tilde{P}$ will cause the extraction of some $\tilde{R} \neq R$ and we cannot guarantee that the adversary is unable to produce an authentication tag for $\tilde{P}$ under the key $\tilde{R}$.

We overcome this difficulty by carefully analyzing the effects of modifying the public helper $P$ on the extracted randomness $R$. We construct fuzzy extractors with a *special linearity property* so that any modification of $P$ into $\tilde{P}$ can be essentially subsumed by giving the attacker the ability to *control* the difference $\Delta$ between the original key $R$ extracted from $w, P$ and the "defective" key $\tilde{R} = R + \Delta$ extracted from $w', \tilde{P}$. 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 our abstract storage device $\Sigma(\mathcal{G})$.

In this application one does not have the freedom of storing some *encoding* of $R$ on $\Sigma(\mathcal{G})$, 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 $\Sigma(\mathcal{G})$ so that an adversary can tamper with it by adding an offset $\Delta$. We show how to construct KMS-MACs using appropriate AMD codes.[3] Using a KMS-MAC, we can turn any fuzzy extractor with the above mentioned special linearity property into a robust fuzzy extractor with essentially the same parameters and no restrictions on the entropy rate of the secret $w$. However, this is (necessarily) done in the *Common Reference String (CRS)* model, as we explain below.

COMMON REFERENCE STRING MODEL. Unfortunately, the impossibility result of [12] guarantees that fuzzy extractors with the special linearity property cannot be constructed in the plain model since they imply robust fuzzy extractors for secrets with

[3] The idea of a KMS-MAC is implicitly used in [11] with a construction that is indeed quite similar to ours. However, the construction there is more complicated since the key is not guaranteed to be uniformly random.

entropy rate below a half. We overcome this pessimistic state of affairs by building such fuzzy extractors (and hence corresponding 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.[4] 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 [4]).

## 2 Algebraic Manipulation Detection Codes

**Definition 1.** *An* $(S, G, \delta)$-algebraic manipulation detection code*, or* $(S, G, \delta)$-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* $s \in \mathcal{S}$*,* $\Delta \in \mathcal{G}$*,* $\Pr[D(\mathcal{E}(s) + \Delta) \notin \{s, \perp\}] \le \delta$*.*

*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 $\Delta$, 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 full version of this work [8].

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 $u$

[4] 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.

for the bit-length of the source $s$ and for every choice $\kappa$ 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 $(S, G, \delta)$-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* $\varpi$ of a $(S, G, \delta)$-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 $(S, G, \delta)$-AMD codes in that class with $S \geq 2^u$ and $\delta \leq 2^{-\kappa}$.*

In the full version of this work [8], 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 $\Delta \in \mathbb{F}^{d+2}$: $\Pr[D(\mathcal{E}(s) + \Delta) \notin \{s, \perp\}] \leq \delta$. It is enough to show that for any $s' \neq s$ and any $\Delta_x, \Delta_f \in \mathbb{F}$: $\Pr[f(x, s) + \Delta_f = f(x + \Delta_x, s')] \leq \delta$. 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 $|B| \leq d+1$. Then

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

$\square$

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.*[5]

We prove the above corollary in the full version of our work [8].

## 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 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

[5] 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 $2\kappa + 2\log(u) + O(\log p)$.

is defined over some *monotone access structure* which maps subsets $B \subseteq \{1,\ldots,n\}$ to a status: `qualified`, `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 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 [24] where the access structure is simply a threshold on the number of players, or a scheme for a general access structure in [15].

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 [26,3]) ensure that such attacks 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 $\delta$-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 $\delta$-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* $(S, G, \delta)$*-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* $\delta$*-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 $q \geq n$, 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 $s$.Even when the value $s$ is large, the authentication information $(x, f(x, s))$ remains relatively small. 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. In some applications we want to make sure that, as long as the honest players form a *qualified* set and the dishonest players form an *unqualified* set, the correct secret will *always* be reconstructed (we do not allow the option of reconstructing $\perp$). This problem is known under the name (unconditional) *secure information dispersal* [23,16] 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 [7]: 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 [2], 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 [1] 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* $(S, G, T, \delta)$-message authentication code with key manipulation security *(*KMS MAC*) is a function* $\mathsf{MAC} : \mathcal{S} \times \mathcal{G} \rightarrow \mathcal{T}$ *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 $(s, \sigma)$. The adversary wins if he can produce an offset $\Delta$ and a message $s' \neq s$ along with a tag $\sigma'$ such that the pair $(s', \sigma')$ verifies correctly under the key $K + \Delta$. The above definition guarantees that such an attack succeeds with probability at most $\delta$. In fact, the definition is slightly stronger than required, since we quantify over all possible tags $\sigma$ 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 $\Delta = 0$.

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 [25]) 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* $2\kappa$*, and the tag size* $\log(T)$ *must be at least* $\kappa$*.*

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

**Theorem 4.** *Let* $\mathcal{E} : \mathcal{S} \rightarrow \mathcal{S} \times \mathcal{G}_1 \times \mathcal{G}_2, s \mapsto (s, x, f(x, s))$ *be a systematic* $(|\mathcal{S}|, |\mathcal{S}||\mathcal{G}_1||\mathcal{G}_2|, \delta)$*-AMD code. Then the function* $\mathsf{MAC} : \mathcal{S} \times (\mathcal{G}_1 \times \mathcal{G}_2) \rightarrow \mathcal{G}_2$ *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 $\Delta = (\Delta_1, \Delta_2) \in \mathcal{G}_1 \times \mathcal{G}_2$, $\sigma, \sigma' \in \mathcal{G}_2$, 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 $\mathsf{MAC}(s', K + \Delta) = \sigma'$ is the event $f(x_1 + \Delta_1, s') + (x_2 + \Delta_2) = \sigma'$, which is the same as $f(x_1 + \Delta_1, s') = -x_2 + \sigma' - \Delta_2$. Let us call this event $E_2$. Thus, we need to bound $\Pr[E_2 \mid E_1]$.

Let us denote $\Delta_f = -\sigma + \sigma' - \Delta_2$ and define an auxiliary event $E_2'$ as $f(x_1 + \Delta_1, s') = f(x_1, s) + \Delta_f$. 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

$$-x_2+\sigma'-\Delta_2 = -(-f(x_1,s)+\sigma)+\sigma'-\Delta_2 = f(x_1,s)+(-\sigma+\sigma'-\Delta_2) = f(x_1,s)+\Delta_f$$

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

$$\Pr[\mathsf{MAC}(s', K+\Delta) = \sigma' \mid \mathsf{MAC}(s, K)=\sigma] = \Pr[f(x_1+\Delta_1, s')=f(x_1, s)+\Delta_f] \leq \delta$$

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 $(S, G, T, \delta)$-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 $2^\lambda$ is the number of elements in $Y$.

SECURE SKETCHES. Let $\mathcal{M}$ be a metric space with distance function $\mathsf{dis}$. 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 (*$\mathsf{SS}, \mathsf{Rec}$*) s.t.:*

1. *The sketching procedure* $\mathsf{SS}$ *on input* $w \in \mathcal{M}$ *returns a bit string* $s \in \{0,1\}^*$*. The recovery procedure* $\mathsf{Rec}$ *takes an element* $w' \in \mathcal{M}$ *and* $s \in \{0,1\}^*$*.*
2. Correctness: *If* $\mathsf{dis}(w, w') \le t$ *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* $\mathbf{H}_\infty(W) \ge m$ *then* $\widetilde{\mathbf{H}}_\infty(W \mid \mathsf{SS}(W)) \ge m'$*.*

*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]. 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 $\mathbf{SD}\,(X_1, X_2) \overset{\text{def}}{=} \frac{1}{2}\sum_{s\in S} |\Pr_{X_1}[s] - \Pr_{X_2}[s]|$. If $\mathbf{SD}\,(X_1, X_2) \le \varepsilon$, we say they are $\varepsilon$-close, and write $X_1 \approx_\varepsilon X_2$. Note that $\varepsilon$-close distributions cannot be distinguished with advantage better than $\varepsilon$ 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 [17], 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* $(m, \ell, \varepsilon)$-extractor *if for all random variables* $X$ *and* $Y$ *such that* $X \in \{0,1\}^n$ *and* $\widetilde{\mathbf{H}}_\infty(X \mid Y) \ge m$*, and* $I \leftarrow U_d$*, we have* $\mathbf{SD}\,(\,(Y, \mathsf{Ext}(X; I), I)\,,\,(Y, U_\ell, U_d)\,) \le \varepsilon$*.*

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 $\mathsf{Ext}(x, i) \overset{\text{def}}{=} [x \cdot i]_1^\ell$ is a $(m, \ell, \varepsilon)$-extractor for any $\ell \le m - 2\log(1/\varepsilon)$. Here the multiplication $x \cdot i$ is performed in the field $\mathbb{F}_{2^n}$ and the notation $[z]_1^\ell$ denotes the first $\ell$ 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* $\mathsf{Init}$ *takes no inputs and outputs a string* $\mathrm{CRS} \in \{0,1\}^*$.
2. *The generation procedure* $\mathsf{Gen}$*, 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* $\mathsf{Rep}$ *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 $(m', \ell, \varepsilon)$-extractor $\mathsf{Ext}\colon \mathcal{M} \times \{0,1\}^d \rightarrow \{0,1\}^\ell$ (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 [4,11]. 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 [4]). 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* $\mathcal{A}$

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

*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 [4,11]. 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 $\mathsf{Ext}$, a secure sketch $(\mathsf{SS}, \mathsf{Rec})$ and a one-time (information-theoretic) message authentication code $\mathsf{MAC}$. 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 $\mathsf{Ext}$ 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 use a special linearity property of the fuzzy extractor. Namely, we want yo make sure that an adversary who modifies $s$ to $\tilde{s}$ will know the offset $R^{\tilde{\Delta}}_{mac} = \tilde{R}_{mac} - R_{mac}$. We formalize this as follows.

FUZZY EXTRACTOR LINEARITY PROPERTY: *For any $w, w', i$, let $\Delta = w' - w$, $s = \mathsf{SS}(w)$, $R = \mathsf{Ext}(w, i)$. For any $\tilde{s}$, let $\tilde{w} = \mathsf{Rec}(w', \tilde{s})$ and $\tilde{R} = \mathsf{Ext}(\tilde{w}, i)$. Then, there is a deterministic function $g$ such that $R^{\tilde{\Delta}} = \tilde{R} - R = g(\Delta, s, \tilde{s}, i)$.*

It is easy to show that, using the syndrome based construction of a secure sketch and the extractor $\mathsf{Ext}(x, i) \overset{\text{def}}{=} [x \cdot i]_1^{\ell}$, the resulting fuzzy extractor satisfies the above linearity property. In the full version of this paper, we give a more general treatment

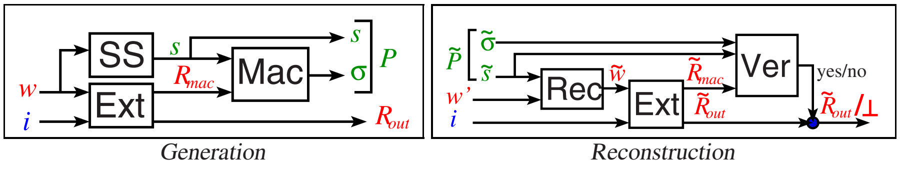

**Fig. 1.** Construction of Robust Fuzzy Extractor

showing that many other natural secure sketch and extractor constructions produce a fuzzy extractor with the above property.

Given a fuzzy extractor with the above linearity property, we can 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 which remains secure 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 $\tilde{s} \neq s$. We formalize this intuition in the next section.

## 5.3 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 and let $u$ be an upper bound on the size of $\mathsf{SS}(w)$. Let $\mathsf{MAC}$ be a $(S, G, T, \delta)$-KMS-MAC, such that $S \geq 2^u$. Assume that the keys for MAC 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 $(W, W') \in \mathcal{F}$. Let Ext be a $(\hat{m}, \ell, \varepsilon)$-strong randomness extractor seeded by randomness $i$ of length $d$. Lastly, assume that our secure sketch and randomness extractor produce a fuzzy extractor which satisfies the above defined fuzzy extractor linearity property.

**Theorem 5.** *When instantiated with the primitives* Ext, MAC *and* (SS, 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 $i, s, \sigma$. 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 $\varepsilon$. 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 $i, s, \sigma$ 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 $2\varepsilon$.

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

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

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

Now we use the fuzzy extractor linearity property which defines the deterministic function $g$ such that

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

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

Hence we have:

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

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 $\mathsf{Ext}(x, i) \overset{\text{def}}{=} [x \cdot i]_1^v$. Recall, we let $u$ denote the length of the secure sketch and $n$ denotes the length of the secret $w$. We define $m' = \widetilde{\mathbf{H}}_\infty(W|\mathsf{SS}(W)) \geq m - u$ to be the residual min entropy "left" in $w$ after seing $s$. Using Theorem 5 and some simple manipulation, we finally get the following concrete corollary.

**Corollary 3.** *Let $m, t, \varepsilon$ and $\delta \geq \varepsilon$ be chosen arbitrarily. Let $\rho = 2\log\left(\frac{2(u+3)}{\varepsilon(\delta-\varepsilon)}\right) + 2$. Let $v = t\left(\log\left(\frac{n}{t}\right) + \log e\right)$ be the upper bound on the volume of the Hamming ball of radius $t$. We construct an $(m, \ell, t, \varepsilon, \delta)$-robust fuzzy where the extracted key length $\ell$ is given by:*

- *For the family $\mathcal{F}^{all}_{(t,m)}$ and post-application robustness $\ell = m' - v - \rho$.*
- *For the family $\mathcal{F}^{all}_{(t,m)}$ and pre-application robustness $\ell = m' - \rho$ as long as $m' - v \geq \rho$.*
- *For the family $\mathcal{F}^{indep}_{(t,m)}$ and both pre/post-application robustness $\ell = m' - \rho$.*

The corollary is proven by bounding the value $\widetilde{\mathbf{H}}_\infty(W|\mathsf{SS}(W), W - W')$ for the two families $\mathcal{F}^{indep}_{(t,m)}$ and $\mathcal{F}^{all}_{(t,m)}$. We give a detailed proof in the full version of this work [8].

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.

## 5.4 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. In addition, our construction extends to the set difference metric in exactly the same way as the construction of [11].

## References

1. Brassard, G., Broadbent, A., Fitzsimons, J., Gambs, S., Tapp, A.: Anonymous quantum communication. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, Springer, Heidelberg (2007)
2. Broadbent, A., Tapp, A.: Information-theoretic security without an honest majority. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, Springer, Heidelberg (2007)
3. Blundo, C., De Santis, A.: Lower bounds for robust secret sharing schemes. Information Processing Letters 63(6) (1997)
4. Boyen, X., Dodis, Y., Katz, J., Ostrovsky, R., Smith, A.: Secure remote authentication using biometric data. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 147–163. Springer, Heidelberg (2005)
5. Boyen, X.: Reusable cryptographic fuzzy extractors. In: 11th ACM Conference on Computer and Communication Security, ACM Press, New York (2004)
6. Cabello, S., Padró, C., Sáez, G.: Secret sharing schemes with detection of cheaters for a general access structure. Designs, Codes and Cryptography 25, 175–188 (2002); In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 175–188. Springer, Heidelberg (1997)
7. Cramer, R., Damgård, I.B., Fehr, S.: On the cost of reconstructing a secret, or VSS with optimal reconstruction phase. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, Springer, Heidelberg (2001)

8. Cramer, R., Dodis, Y., Fehr, S., Padró, C. Wichs, D.: Detection of Algebraic Manipulation with Applications to Robust Secret Sharing and Fuzzy Extractors. Technical Reports 2008/030, Cryptology ePrint archive, `http://eprint.iacr.org/2008/030`
9. Dodis, Y.: Exposure Resillient Cryptography. Ph.D. Thesis, MIT (2000)
10. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. Technical Report 2003/235, Cryptology ePrint archive, Previous version appeared at EUROCRYPT 2004, `http://eprint.iacr.org/2003/235`
11. Dodis, Y., Katz, J., Reyzin, L., Smith, A.: Robust Fuzzy Extractors and Authenticated Key Agreement from Close Secrets. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, Springer, Heidelberg (2006)
12. Dodis, Y., Spencer, J.: On the (non-)universality of the one-time pad. In: 43rd Annual Symposium on Foundations of Computer Science, pp. 376–385. IEEE, Los Alamitos (2002)
13. Desmedt, Y., Wang, Y.: Perfectly secure message transmission revisited. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, Springer, Heidelberg (1993)
14. Dolev, D., Dwork, C., Waarts, O., Yung, M.: Perfectly secure message transmission. Journal of the ACM 40(1) (1993)
15. Karchmer, M., Wigderson, A.: On span programs. In: 8th Annual Conference on Structure in Complexity Theory (SCTC 1993), IEEE, Los Alamitos (1993)
16. Krawczyk, H.: Distributed fingerprints and secure information dispersal. In: 12th ACM Symposium on Principles of Distributed Computing (PODC), ACM Press, New York (1993)
17. Nisan, N., Zuckerman, D.: Randomness is linear in space. Journal of Computer and System Sciences 52(1), 43–53 (1996)
18. Obana, S., Araki, T.: Almost Optimum Secret Sharing Schemes Secure Against Cheating for Arbitrary Secret Distribution. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, Springer, Heidelberg (2006)
19. Ogata, W., Kurosawa, K.: Optimum secret sharing scheme secure against cheating. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, Springer, Heidelberg (1996)
20. Ogata, W., Kurosawa, K., Stinson, D.R., Saido, H.: New combinatorial designs and their applications to authentication codes and secret sharing schemes. Discrete Mathematics 279, 383–405 (2004)
21. Padró, C., Sáez, G., Villar, J.L.: Detection of cheaters in vector space secret sharing schemes. Designs, Codes and Cryptography 16, 75–85 (1999)
22. Padró, C.: Robust vector space secret sharing schemes. Information Processing Letters 68, 107–111 (1998)
23. Rabin, M.O.: Efficient dispersal of information for security, load balancing, and fault tolerance. Journal of the ACM 36(2) (1989)
24. Shamir, A.: How to share a secret. Communications of the Association for Computing Machinery 22(11) (1979)
25. Simmons, G.J.: Authentication theory/Coding Theory. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, Springer, Heidelberg (1985)
26. Tompa, M., Woll, H.: How to share a secret with cheaters. Journal of Cryptology 1(3) (1988)