Symmetric-key Corruption Detection : When XOR-MACs Meet Combinatorial Group Testing*

Kazuhiko Minematsu and Norifumi Kamiya

NEC Corporation, Japan
k-minematsu@ah.jp.nec.com kamiya@bc.jp.nec.com

Abstract. We study a class of MACs, which we call corruption detectable MAC, that is able to not only check the integrity of the whole message, but also detect a part of the message that is corrupted. It can be seen as an application of the classical Combinatorial Group Testing (CGT) to message authentication. However, previous work on this application has inherent limitation in communication. We present a novel approach to combine CGT and a class of linear MACs (XOR-MAC) that enables to break this limit. Our proposal, XOR-GTM, has a significantly smaller communication cost than any of the previous ones, keeping the same corruption detection capability. Our numerical examples for storage application show a reduction of communication by a factor of around 15 to 70 compared with previous schemes. XOR-GTM is parallelizable and is as efficient as standard MACs. We prove that XOR-GTM is provably secure under the standard pseudorandomness assumptions.

Keywords: MAC, Corruption Detection, Combinatorial Group Testing, XOR-MAC.

1 Introduction

MAC and corruption detection. Message Authentication Code (MAC) is a symmetric-key cryptographic function for ensuring the message authenticity. In a typical MAC protocol, the sender of message M computes a fixed, short tag T as a function of MAC : $T = \mathrm{MAC}(K, M)$ for the secret key K. On receiving the tuple (M', T') from the sender, which may be corrupted by the adversary, the receiver checks if the tuple is correct by performing tag computation $\widehat{T} = \mathrm{MAC}(K, M')$ using the shared K and comparing T' and $\widehat{T}$.

Standard MAC functions, such as HMAC or CMAC, are quite efficient, and their securities have been extensively studied. However, the naïve application described above only tells us whether M' has been corrupted or not, and nothing about how M' is corrupted, i.e. the locations of the corruptions. Suppose a message can be divided into m parts, say entries in a database or sectors in a HDD. If we take a MAC tag for each part, the number of tags is m, and the scheme gives us full information on the corrupted parts. However, it does not scale as the amount of tag information grows linearly with m. We want to reduce the number of tags without losing the detection ability too much. This need is fundamental and naturally arises in many applications, such as storage integrity protection and low-power wireless communications. In particular, the size of trusted storage for maintaining integrity of a large, untrusted storage has been an important issue in the storage security research [43,44] and the commercial solutions such as Tripwire. In a broader sense, where hash or digital signature may be used as well as MACs, similar problems occur at verification of downloaded software, malware classification, and digital forensics etc. The underlying problem is how to minimize the number of tags keeping the sufficient ability to detect the corrupted parts. We call a MAC scheme for this purpose *a corruption detectable MAC*.

Combinatorial group testing and its applications. It is known that the corruption detectable MAC can be seen as an application of Combinatorial Group Testing (CGT) [28], a method invented by Dorfman in WWII [20] to effectively find blood samples infected by syphilis. It has been studied for long years, mainly from the viewpoint of theoretical computer science and applications to biology, such as DNA library screening [41]. Recently, CGT has received significant attentions from many other areas, such as [38,51,23,31].

* A preliminary version of this paper appears in the proceedings of ESORICS 2019. This is the full version.

When CGT is applied to our problem, the sender has a message M consisting of m data items to be authenticated. We assume at most d items of M may be corrupted when it is sent to the receiver. What the sender can do is to take t MAC tags for certain sub-sequences of M for some $t \leq m$, corresponding to *group testing* in CGT. How tags are computed is completely specified by a $t \times m$ binary matrix $\mathbf{H}$, called test matrix. The trivial scheme described earlier uses the identity matrix and thus is inefficient. However, it is known that if $\mathbf{H}$ has a property called d-disjunct[1], we can detect $d' \leq d$ corrupted items. Since d-disjunct matrix is theoretically possible with $O(d^2 \log m)$ rows, the use of d-disjunct matrix can significantly reduce the number of tags if d is much smaller than m.

Despite the simple and natural problem, corruption detectable MACs have received surprisingly less attention to date. To our knowledge, Crescenzo et al. [17,19] and Goodrich et al. [28] are the earliest work on this direction. Corruption-localizing hash function was proposed at ESORICS 2009 [18] and subsequently studied [18,16,12]. Minematsu [40] at ESORICS 2015 studied the computational aspects of corruption detectable MACs. Fang et al. showed an application to storage integrity check [27]. Hirose and Shikata [30] applied CGT to aggregate MACs, a related area of corruption detectable MACs.

Limitation of previous schemes. The most of previous corruption detectable MACs and its variants used existing d-disjunct matrices for their test matrix. We call it DirectGTM for the direct use of disjunct matrix in a Group-Test-based MAC. Construction of d-disjunct matrix has been extensively studied (see Section 3.2), however, finding one with optimally small number of rows for given m and d is still a hard combinatorial problem even for tiny d. Besides, if d is $O(\sqrt{m/\log m})$ it is impossible to build a non-trivial d-disjunct matrix. Consequently, the communication efficiency of DirectGTM is inherently bounded by the current knowledge of constructions of small-row disjunct matrix. The aforementioned work by Minematsu [40] (hereafter Min15) showed that the computation cost of DirectGTM can be reduced to almost the single MAC function if we employ a MAC function similar to XOR-MAC [6] or PMAC [10]. However, the communication cost (i.e. the number of tags) is not improved.

Beyond DirectGTM. We break the above limitation by presenting a new class of corruption detectable MACs. The proposed scheme has a structure very similar to Min15 for tag generation, where a subsequence of message specified by a row of test matrix ($\mathbf{H}$) is processed by a hash-then-encrypt style of MAC function. However, the verification is conceptually different from Min15 in that we use the *decryption* of tags instead of themselves for verification. This seemingly tiny change will bring a ultimate difference from Min15, since it allows us to use any *linear combination* of (decrypted) tags for a verification of a new subsequence that is not specified by $\mathbf{H}$! Surprisingly, this suggests that $\mathbf{H}$ is not necessarily d-disjunct, but only the *row span* of $\mathbf{H}$ over GF(2) is required to be d-disjunct. Therefore, the communication efficiency is determined not by the number of rows but the *rank* of d-disjunct matrix. We define the appropriate security notions for such schemes, which we call XOR-GTM, and show XOR-GTM is provably secure using the standard symmetric-key primitives, such as pseudorandom function (PRF) and tweakable pseudorandom permutation (TPRP).

Efficient instantiations. Efficient instantiations of XOR-GTM are not easy, since despite the numerous studies on small-row d-disjunct matrices, their GF(2)-rank has rarely been studied. Even worse, as far as we studied, the state-of-the-art constructions tend to have a high rank, implying only a marginal gain from (ideal) DirectGTM. Instead, somewhat surprisingly, what we find useful are some classes of d-disjunct matrices that are near-square, which are *almost useless* for CGT in general. The matrices we found are not new : one of them is a modified Hadamard matrix and the other is a point-line incidence matrix defined over finite geometry. Both are classical and have been extensively studied for decades. However, when we instantiate XOR-GTM with them (more precisely, the bases of the row vectors of these matrices are used as the test matrix), we could achieve what is impossible with *any* instantiation of DirectGTM including Min15. In more detail, by using Hadamard matrix, DirectGTM can detect $d = 2$ corruptions with $\log_2(m+1)+1$ tags, which is better than any DirectGTM scheme with 2-disjunct test matrix by a factor of 3 to 5. Moreover, with point-line incidence matrices, we can detect corruptions of $d = O(\sqrt{m})$ parts with $t = O(\sqrt{m})$ tags. This exhibits a very strong advantage over the existing schemes. In our numerical examples for storage application (Section 7.2), XOR-GTM needs fewer tags than the trivial scheme by a factor of roughly 15 to 70, while any instantiation of DirectGTM with d-disjunct test matrix has almost no gain.

[1] The minimum condition is weaker (d-separable or $\overline{d}$-separable), however this does not guarantee an efficient detection.

As well as Min15, XOR-GTM is parallelizable and incremental [4]. The computation of XOR-GTM is very efficient at it is essentially the same as taking a single MAC tag for the message. For our instantiations, the underlying matrices are highly structured, which is useful for efficient implementation. This gives another great advantage in comparison to the naïve use of a d-disjunct matrix since the small-row disjunct matrix is typically very complex.

Further related work. In the context of cryptography in general, CGT has been used for various related applications such as [52,11,15,47,1]. Although the interaction between CGT and cryptography has not received much attentions thus far, we believe it is very promising.

2 Preliminaries

2.1 Basic Notations

Let $\{0,1\}^\infty$ be the set of all binary strings, including the empty string ε. We write the bit length of $X \in \{0,1\}^\infty$ by $|X|$. Here $|\varepsilon| = 0$. We define $\{0,1\}^* \stackrel{\text{def}}{=} \{0,1\}^\infty \setminus \{\varepsilon\}$. For a binary string X, its hamming weight is denoted by $\mathrm{Hw}(X)$. We define a set of m-tuples of non-empty strings as $\{0,1\}^{*m} \stackrel{\text{def}}{=} (\{0,1\}^*)^m$, and define $\{0,1\}^{*\leq m} \stackrel{\text{def}}{=} \bigcup_{i=1}^{m} \{0,1\}^{*i}$. We write $\{0,1\}^{\infty m} \stackrel{\text{def}}{=} (\{0,1\}^\infty)^m$ to denote the set of m-tuples of possibly empty strings. Here, (ε, v) and (v, ε) for $v \neq \varepsilon$ are distinct elements of $\{0,1\}^{\infty 2}$. A uniform sampling over a set $\mathcal{X}$ is written as $X \xleftarrow{\$} \mathcal{X}$. The base of logarithm is 2 unless otherwise written.

For positive integer n, let[2] $[\![n]\!] = \{1, \ldots, n\}$ and $(\!|n|\!) = \{0, \ldots, n-1\}$. For a finite set $\mathcal{X}$, $2^{\mathcal{X}}$ denotes its power set. For any set $\mathcal{X}$, let $\mathsf{cmp} : \mathcal{X} \times \mathcal{X} \to \{0,1\}$ be the comparison function (i.e. $\mathsf{cmp}(X,Y) = 0$ if $X = Y$ and $\mathsf{cmp}(X,Y) = 1$ if $X \neq Y$). For $M = (M[1], \ldots, M[m])$ and $M' = (M'[1], \ldots, M'[m])$ in $\{0,1\}^{*m}$, we define vector comparison $\mathsf{vdiff}(M, M')$ and index difference $\mathsf{diff}(M, M')$ as

$$\mathsf{vdiff}(M, M') = (\mathsf{cmp}(M[1], M'[1]), \ldots, \mathsf{cmp}(M[m], M'[m])),$$
$$\mathsf{diff}(M, M') = \{i \in [\![m]\!] : M[i] \neq M'[i]\}.$$

For $X = (X[1], \ldots, X[m]) \in \{0,1\}^m$, let $M \ominus X \in \{0,1\}^{*m'}$ be a vector obtained by subtracting $M[i]$s for all i s.t. $X[i] = 0$, where $m' = \mathrm{Hw}(X)$. For example, if $m = 4$ and $X = (1,0,1,0)$, $M \ominus X = (M[1], M[3])$.

Disjunct matrix. Let $\mathbf{M}$ be an $n \times m$ binary matrix. We write $\mathbf{M}_{i,*}$ to denote the i-th row, and $\mathbf{M}_{*,j}$ to denote the j-th column, and $\mathbf{M}_{i,j}$ to denote the entry at i-th row and j-th column. For simplicity we may abbreviate $\mathbf{M}_{i,*}$ to $\mathbf{M}_i$. The rows and columns of $\mathbf{M}$ are interchangeably seen as sets, e.g. $\mathbf{M}_i = \{j \in [\![m]\!] : \mathbf{M}_{i,j} = 1\}$, and $a \in \mathbf{M}_i$ means $\mathbf{M}_{i,a} = 1$.

For $X, Y \in \{0,1\}^n$, $X \vee Y$ denotes the bitwise Boolean sum (logical OR) of X and Y. We say $\mathbf{M}$ is d-disjunct [21] if, for any $\mathcal{S} \subseteq [\![m]\!]$ and $|\mathcal{S}| \in [\![d]\!]$, $\mathbf{M}_{*,j} \not\subseteq \bigvee_{h \in \mathcal{S}} \mathbf{M}_{*,h}$ holds for any $j \notin \mathcal{S}$. That is, a sum of any distinct $i \leq d$ columns of $\mathbf{M}$ does not cover any other column. An $m \times m$ identity matrix is trivially m-disjunct. A d-disjunct matrix is also said to have *disjunctness* of d. The most important property of d-disjunct matrix is the number of rows for any given d and m (See Sect. 3.2). For convention, we say "ideal" or "optimal" d-disjunct matrix to mean one achieves the smallest number of rows for a fixed d and m.

2.2 Cryptographic Functions

A keyed function with key space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$ is a function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$. We may write $F_K(X)$ for $F(K, X)$, and if $\mathcal{X} = \mathcal{M} \times [\![\ell]\!]$ for some positive integer ℓ, write $F_K^i(M)$ to denote $F_K(M, i)$ for $(M, i) \in \mathcal{X}$. A tweakable block cipher (TBC) [36] $E : \mathcal{K} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X}$ is a keyed permutation over $\mathcal{X}$ with additional tweak input in $\mathcal{T}$, i.e., $E(K, T, *)$ for any $(K, T) \in \mathcal{K} \times \mathcal{T}$ is a permutation over $\mathcal{X}$. We may write $E_K^T(X)$ instead of $E(K, T, X)$ or $E_K(T, X)$. The decryption of X with tweak T is written as $E_K^{-1}(T, X)$ or $E_K^{T,-1}(X)$.

[2] It is customary to use $[n]$ but we want to avoid confusion, say with $M[i]$.

A uniform random function (URF) $\mathsf{R} : \mathcal{X} \to \mathcal{Y}$ is a keyed function with uniform key distribution over all functions from $\mathcal{X}$ to $\mathcal{Y}$. Uniform random permutation (URP) $\mathsf{P} : \mathcal{X} \to \mathcal{X}$ is defined analogously. A tweakable uniform random permutation (TURP) with tweak space $\mathcal{T}$ and message space $\mathcal{X}$ is denoted by $\widetilde{\mathsf{P}} : \mathcal{T} \times \mathcal{X} \to \mathcal{X}$, which is a set of independent URPs indexed by $\mathcal{T}$. Decryption of $\widetilde{\mathsf{P}}$ is denoted by $\widetilde{\mathsf{P}}^{-1}$. Let $\mathcal{A}$ be an adversary who (possibly adaptively) queries to oracle $\mathcal{O}$ and outputs a bit as a final decision. The advantage of $\mathcal{A}$ in distinguishing two oracles, $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ and $F' : \mathcal{K}' \times \mathcal{X} \to \mathcal{Y}$, is defined as

$$\mathbf{Adv}^{\mathsf{ind}}_{F_K, F'_{K'}}(\mathcal{A}) \stackrel{\text{def}}{=} \left| \Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{F_K} \Rightarrow 1] - \Pr[K' \xleftarrow{\$} \mathcal{K}' : \mathcal{A}^{F'_{K'}} \Rightarrow 1] \right|,$$

where $\mathcal{A}^{F_K} \Rightarrow 1$ denotes the probability that $\mathcal{A}$'s final decision is 1 when the oracle is F_K. We let $\mathbf{Adv}^{\mathsf{ind}}_{F_K, \mathsf{R}}(\mathcal{A}) \stackrel{\text{def}}{=} \mathbf{Adv}^{\mathsf{prf}}_{F_K}(\mathcal{A})$ where $\mathsf{R} : \mathcal{X} \to \mathcal{Y}$ is a URF, called PRF-advantage of F_K (for $\mathcal{A}$). We say F_K is a PRF when $\mathbf{Adv}^{\mathsf{prf}}_{F_K}(\mathcal{A})$ is sufficiently small for all practical adversaries, where the definitions of "sufficiently small" and "practical" may depend on the context [3]. For a TBC $E : \mathcal{K} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X}$, we define Tweakable PRP (TPRP) advantage as $\mathbf{Adv}^{\mathsf{tprp}}_{E_K}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{ind}}_{E_K, \widetilde{\mathsf{P}}}(\mathcal{A})$ for $\mathcal{A}$ using chosen-plaintext (encryption) queries.

3 Previous Corruption Detectable MACs

3.1 DirectGTM

Given a $t \times m$ binary test matrix $\mathbf{H}$, the basic form of corruption detectable MAC is described as follows. For message $M \in \{0,1\}^{*m}$, the sender first computes

$$T[i] = \mathsf{MAC}_K(M \ominus \mathbf{H}_i, i)$$

for all $i \in [\![t]\!]$, using a MAC function $\mathsf{MAC} : \mathcal{K} \times \{0,1\}^{*\leq m} \times [\![t]\!] \to \{0,1\}^n$. The output is the tag vector $T = (T[1], \ldots, T[t]) \in (\{0,1\}^n)^t$. The verifier receives (M', T'), which may be a corrupted version of (M, T), and computes $\widehat{T} = (\widehat{T}[1], \ldots, \widehat{T}[t])$, where $\widehat{T}[i] = \mathsf{MAC}_K(M' \ominus \mathbf{H}_i, i)$, and compares T' and $\widehat{T}$ to obtain $\mathsf{vdiff}(T', \widehat{T})$.

Then, the verifier tries to detect the corrupted items, by subtracting $\mathbf{H}_i$ (as a set) from $[\![m]\!]$ for all $i \in [\![t]\!]$ such that $\mathsf{cmp}(T'[i], \widehat{T}[i]) = 0$, i.e., the tags are matched. The remaining set indicates the indexes of the corrupted items. In the context of CGT, the above procedure is called *naïve decoding* [21]. The following is a well-known fact from the property of d-disjunct matrix.

Proposition 1. *Suppose* $\mathbf{H}$ *is* d*-disjunct and* $\mathsf{diff}(M, M') \leq d$ *and* $T' = T$*. Then, the naïve decoding correctly detects all the corrupted items if* $\mathsf{cmp}(T'[i], \widehat{T}[i]) = \mathsf{cmp}(M \ominus \mathbf{H}_i, M' \ominus \mathbf{H}_i)$ *for all* $i \in [\![t]\!]$*.*

This holds since a d-disjunct matrix implies that any negative (uncorrupted) item is included in at least one test that does not contain any positive (corrupted) item and thus evicted. We call a corruption detectable MAC of the above form DirectGTM. While there are some differences, the previous corruption detectable schemes are all classified as DirectGTM.

Min15 [40] studied the computation overhead of DirectGTM from the standard MAC. He showed that one can reduce the computation cost almost as low as the standard MAC independent of the test matrix, by employing a deterministic MAC similar to XOR-MAC [6] or PMAC [46]. Nevertheless, Min15 is still a DirectGTM in that it needs d-disjunct $\mathbf{H}$ to detect d corruptions.

Hirose and Shikata [30] showed an application of CGT to aggregate MAC, which is a MAC scheme for aggregation of MAC tags computed by independent nodes. The original proposal by Katz and Lindell [34] takes a sum (XOR) of all tags for aggregation, hence it is not possible to detect the corrupted tags (or nodes). Instead, the scheme of [30] aggregates the tags following the test matrix $\mathbf{H}$, and yields t tags. This can be interpreted as a form of DirectGTM: if $\mathbf{H}$ is d-disjunct, we can detect up to d corrupted tags (nodes).

3.2 Constructions of Disjunct Matrix

The construction of disjunct matrix has been extensively studied from the viewpoint of designs and codes. Classical examples are Macula [37] and Kautz and Singleton [35]. The Du-Hwang book [21] describes a number of known constructions. Eppstein et al. proposed Chinese Reminder Sieve (CRS) [24]. Thierry-Mieg proposed Shifted Transversal Design (STD) [50] for biological applications. Constructions based on constant weight code are described at [2]. As CGT receives attentions from various fields, more schemes, both adaptive and non-adaptive, are expected.

Let $t_{\min}(d, m)$ be the minimum number of rows for a d-disjunct matrix of m columns. It is known that $t_{\min}(d, m) = \Theta(d^2 \log m)$ [21]. The seminal work by Porat and Rothschild [45] showed an order-optimal and deterministic construction of d-disjunct matrix, however it needs a large (though polynomial) search for matrix generation. Besides, the optimality *including the constant* is not known. Only the case $d = 1$ has been solved: 1-disjunct matrix implies that no column is contained in another column. Such a matrix is called a completely separating system, and has about $\log m$ rows [21]. If we relax the condition to 1-separable, the concrete construction of $\lceil \log m \rceil$ rows is easy as it is equivalent to that the all columns are distinct. However, even for the case $d = 2$, the optimal construction remains open for decades.

Lower bounds. A lower bound of

$$t_{\min}(d, m) \geq \min\left\{ \binom{d+2}{2}, m \right\} \tag{1}$$

was shown by Dýachkov and Rykov [22], attributed to Bassalygo. An improved bound has been shown by Shangguan and Ge [48]:

$$t_{\min}(d, m) \geq \min\left\{ \frac{d^2(15+\sqrt{33})}{24}, m \right\}. \tag{2}$$

Moreover, there is a conjectured lower bound by Erdös et al. [26]

$$t_{\min}(d, m) \geq \min\left\{ (d+1)^2, m \right\}, \tag{3}$$

which was later shown to be correct for $d \leq 5$ (see [48]).

4 Our Proposal

4.1 Breaking the Barrier of DirectGTM

In the previous DirectGTMs, the choice of test matrix was independent of the choice of MAC_K, and a d-disjunct matrix or its variant was suggested to be used as $\mathbf{H}$. Thus, the communication cost of DirectGTM is reduced only by finding a small-row d-disjunct matrix. Unfortunately, this is a hard problem even for tiny d and even impossible when d is close to $\sqrt{m/\log m}$, as shown in the previous section. This limits the practical usefulness of DirectGTM.

We break this barrier by exploiting a certain linearity in the MAC computation. Suppose we have a Min15 scheme with t tests (tags). There is an intermediate vector $S = (S[1], \ldots, S[t])$, where $S[i]$ is a keyed hash value of the subsequence of M specified by $\mathbf{H}_i$, and an encryption of $S[i]$ yields the i-th tag $T[i]$. We observe that checking at $T[i]$ is equivalent to checking at $S[i]$, and even more, any linear combination of $S[i]$s will yield another test, i.e., a verification of a new subsequence of M.

For example, if $T[1]$ is a tag for $(M[1], M[2])$ and $T[2]$ is a tag for $(M[2], M[3])$, our scheme allows to use $S[1] \oplus S[2]$ as a test for $(M[1], M[3])$, since the verifier can computes $S[1]$ and $S[2]$ by decrypting the sent tags and see if $S[1] \oplus S[2]$ match with $\widehat{S}[1] \oplus \widehat{S}[2]$ computed from M. Hence, without explicitly sending a tag for $(M[1], M[3])$, we could perform three tests with two tags. In other words, when the authenticator takes MAC tags based on test matrix $\mathbf{H}$, the verifier can use (any sub-matrix of) the row span of $\mathbf{H}$ as a *virtual* test matrix. This can bring significantly richer information to the verifier without increasing the communication.

4.2 Syntax

We first describe the syntax for corruption detectable MAC. Let positive integers m, t, n be a parameter. Typically they are fixed, but the model can be easily extended to the case of variable parameters. A corruption detectable MAC consists of four algorithms. The key generation $\mathsf{KG} : \mathbb{N} \to \mathcal{K}$ takes a security parameter $p \in \mathbb{N}$ and returns a key $K \in \mathcal{K}$. The key K is shared by the legitimate parties (authenticator and verifier).

The tagging function $\mathsf{Tag} : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ takes message $M \in \mathcal{M}$ and key $K \in \mathcal{K}$ to return a tag vector $T \in \mathcal{T}$, where $\mathcal{M} = \{0,1\}^{*m}$ and $\mathcal{T} = (\{0,1\}^n)^t$. Message and tag vector are written as $M = (M[1], \ldots, M[m]) \in \mathcal{M}$ and $T = (T[1], \ldots, T[t]) \in \mathcal{T}$, where $M[i]$ is called a message item (or item for short), and $T[i]$ is called a tag string (or tag for short). The verification function $\mathsf{Ver} : \mathcal{K} \times \mathcal{M} \times \mathcal{T} \to \mathcal{D}$ with $\mathcal{D} = \{0,1\}$ is for verification : $\mathsf{Ver}_K(M', T') = 0$ denotes the tuple (M', T') is authenticated (no corruption), while $\mathsf{Ver}_K(M', T') = 1$ denotes the tuple is not authenticated, thus an authentication failure. Finally, the detection function $\mathsf{Det} : \mathcal{K} \times \mathcal{M} \times \mathcal{T} \to 2^{[\![m]\!]}$ takes key K and the possibly corrupted tuple (M', T') to return the derived index set of corrupted message items $\mathcal{P} \in 2^{[\![m]\!]}$. For example, when $\{1,3\} \leftarrow \mathsf{Det}_K(M, T)$ it means $M'[1]$ and $M'[3]$ are considered to be corrupted.

In addition, we define the string-wise verification function $\mathsf{SVer} : \mathcal{K} \times \mathcal{M} \times \mathcal{T} \to \mathcal{D}^t$ that takes K, M' and T' to return $B \in \mathcal{D}^t$ which gives more information on the verification failure, where $\mathsf{Ver}_K(M', T') = 0$ indicates $B = (B[1], \ldots, B[t]) = (0, \ldots, 0)$ and $\mathsf{Ver}_K(M', T') = 1$ indicates $B \neq (0, \ldots, 0)$, thus potentially giving about t-bit information on verification failure rather than the unitary error event. The precise meaning of $B[i]$ will depend on the scheme. While SVer may have practical relevance, we use it to simplify security analysis. This syntax will be used to define our security notions at Section 5.

4.3 XOR-GTM

We present our corruption detectable MAC, XOR-GTM. The name comes from the similarity to XOR-MAC [6], though, XOR-MAC is a plain stateful MAC.

Parameters. XOR-GTM is a deterministic MAC over $\mathcal{M} = \{0,1\}^{*m}$ for a fixed, positive integer m. It has two parameters, $t \times m$ binary test matrix $\mathbf{H}$ and its *extension rule* R. Here, R specifies the linear combinations of rows of $\mathbf{H}$, and is defined as $R = (R_1, \ldots, R_v)$, where $R_i \subseteq [\![t]\!]$, for some $t \leq v$. We define $\mathbf{H}^R$ as a $v \times m$ *extended test matrix* obtained by taking linear combinations of $\mathbf{H}$ rows specified by R, that is,

$$\mathbf{H}^R_i = \bigoplus_{j \in R_i} \mathbf{H}_j, \text{ for all } i \in [\![v]\!].$$

For simplicity, we assume $R_i = \{i\}$ for $i \in [\![t]\!]$, hence $\mathbf{H}^R$ is a $v \times m$ matrix obtained by adding $v - t$ rows to $\mathbf{H}$. Similarly for $X = (X[1], \ldots, X[t]) \in (\{0,1\}^n)^t$, we define $X^R = (X^R[1], \ldots, X^R[v]) \in (\{0,1\}^n)^v$ as

$$X^R[i] = \bigoplus_{j \in R_i} X[j], \text{ for all } i \in [\![v]\!], \tag{4}$$

which is an expansion of X by R.

To avoid trivial attacks by the adversary and apparently redundant tests, we require the following soundness conditions for $\mathbf{H}$ and R.

Definition 1. *A pair of* $\mathbf{H}$ *and* R *(or equivalently,* $\mathbf{H}^R$*) is said to be sound if all rows of* $\mathbf{H}^R$ *are distinct and there is an all-one row.*

For simplicity we assume $\mathbf{H}_1$ is all-one whenever it is sound. The cryptographic components of XOR-GTM are PRF $F : \mathcal{K} \times [\![m]\!] \times \{0,1\}^* \to \{0,1\}^n$ and TBC $G : \mathcal{K}' \times [\![t]\!] \times \{0,1\}^n \to \{0,1\}^n$ for tweak space $[\![t]\!]$. The procedures of XOR-GTM are as follows.

Tag computation. For message $M \in \{0,1\}^{*m}$, we define $\mathsf{XOR\text{-}GTM}[F_K].\mathrm{hash}(M) = (S[1], \ldots, S[t])$, where

$$S[i] = \bigoplus_{j \in \mathbf{H}_i} F^j_K(M[j]).$$

Tag computation procedure ($\mathsf{XOR\text{-}GTM}[F_K, G_{K'}].\mathrm{tag}(M)$) first performs the above and compute

$$T[i] = G^i_{K'}(S[i])$$

for all $i \in [\![t]\!]$ and outputs $T = (T[1], \ldots, T[t])$, which is called a tag vector.

Verification and corruption detection. The verification of tuple (M', T') ($\mathsf{XOR\text{-}GTM}[F_K, G_{K'}].\mathrm{verify}(M', T')$) first computes $\widehat{T} = \mathsf{XOR\text{-}GTM}[F_K, G_{K'}].\mathrm{tag}(M')$, and checks if $\widehat{T} = T'$. In fact, as we assumed $\mathbf{H}_1$ is all-one, checking the first components will suffice. If they do not match, the receiver tries to detect corruptions by computing $S' = (S'[1], \ldots, S'[t])$ where $S'[i] = G^{i,-1}_{K'}(T'[i])$, and $\widehat{S} = \mathsf{XOR\text{-}GTM}[F_K].\mathrm{hash}(M')$, and expanding $\widehat{S}$ and S' to $\widehat{S}^R$ and $(S')^R$, using (4). The receiver then performs the naïve decoder with $\mathbf{H}^R$. That is, for all $i \in [\![v]\!]$ such that $\widehat{S}^R[i] = (S')^R[i]$, the receiver removes all the elements of $\mathbf{H}^R_i$ (as a set) from $[\![m]\!]$, and outputs the remaining set as the indexes of the corrupted items. We define this procedure as $\mathsf{XOR\text{-}GTM}[F_K, G_{K'}].\mathrm{detect}(M', T')$. See Figure 1 for the pseudocodes.

Relationship to Min15. When $v = t$, $\mathbf{H}^R = \mathbf{H}$ and $T^R = T$ hold and tag computation, verification and detection are exactly the same as Min15, except the fact that we explicitly require the invertibility of G while Min15 does not. The equivalence of verification holds because $T[i] = \widehat{T}[i]$ is equivalent to $G^{i,-1}_{K'}(T[i]) = G^{i,-1}_{K'}(\widehat{T}[i])$.

Example 1. Suppose $m = 4, t = 3$ and $v = 6$ with following $\mathbf{H}$ and $\mathbf{H}^R$. Here, $R = (\{1\}, \{2\}, \{3\}, \{1,2\}, \{2,3\}, \{1,2,3\})$.

$$\mathbf{H} = \begin{pmatrix} 1&1&0&0 \\ 0&1&1&0 \\ 0&0&1&1 \end{pmatrix}, \quad \mathbf{H}^R = \begin{pmatrix} 1&1&0&0 \\ 0&1&1&0 \\ 0&0&1&1 \\ 1&0&1&0 \\ 0&1&0&1 \\ 1&0&0&1 \end{pmatrix}.$$

For message $M \in \{0,1\}^{*3}$, $\mathsf{XOR\text{-}GTM}$ generates $T = (T[1], T[2], T[3])$ where

$$\begin{aligned} T[1] &= G^1_{K'}(S[1]), \quad S[1] = F^1_K(M[1]) \oplus F^2_K(M[2]) \\ T[2] &= G^2_{K'}(S[2]), \quad S[2] = F^2_K(M[2]) \oplus F^3_K(M[3])) \\ T[3] &= G^3_{K'}(S[3]), \quad S[3] = F^3_K(M[3]) \oplus F^4_K(M[4])). \end{aligned}$$

The linear combinations of hash values (S) specified by R are:

$$S[4] = S[1] \oplus S[2] = F^1_K(M[1]) \oplus F^3_K(M[3]) \tag{5}$$

$$S[5] = S[2] \oplus S[3] = F^2_K(M[2]) \oplus F^4_K(M[4]) \tag{6}$$

$$S[6] = S[1] \oplus S[2] \oplus S[3] = F^1_K(M[1]) \oplus F^4_K(M[4]). \tag{7}$$

On receiving (M', T'), the (basic) verification is done by comparing the received $T' = (T'[1], T'[2], T'[3])$ with $\widehat{T} = \mathsf{XOR\text{-}GTM}[F_K, G_{K'}](M')$. To detect the corruptions, we decrypt T' by $G_{K'}$ to obtain $S' = (S'[1], S'[2], S'[3])$ and compute $\widehat{S} = (\widehat{S}[1], \widehat{S}[2], \widehat{S}[3])$ from M', and expand S' and $\widehat{S}$ as (5)(6)(7) to obtain $(S')^R$ and $\widehat{S}^R$.

In fact, $\mathbf{H}^R$ comes from Macula [37] with parameter $(n, k, d) = (4, 3, 2)$, hence is 2-disjunct. Thus, this example detects up to 2 corruptions among 4 items using 3 tags, which is impossible with $\mathsf{DirectGTM}$ as there is no 3×4 2-disjunct matrix from (3). We note that this example is just to understand the idea : $\mathbf{H}^R$ is not sound as it lacks all-one row (hence not secure), and adding all-one row will make it useless.

Efficient computation. Since $\mathbf{H}_i$ may intersect with other $\mathbf{H}_j$, a straightforward tag computation will bring lots of redundant F calls. However, the computation of T can be done by m calls of F_K and t calls of $G_{K'}$ as well as Min15. See Figure 1. A nice thing is that this feature is independent of the contents of $\mathbf{H}$. Usually $m \gg t$ (as this is why we use CGT!), hence, $\mathsf{XOR\text{-}GTM}$ is roughly as efficient as standard MACs applied to the whole message. An experimental AES-based implementation in Min15 also supports this claim.

Algorithm
$\mathsf{XOR\text{-}GTM}[F_K, G_{K'}].\mathrm{tag}(M)$

1. $S \leftarrow \mathsf{XOR\text{-}GTM}[F_K].\mathrm{hash}(M)$
2. **for** $i = 1$ **to** t **do**
3. $\quad T[i] \leftarrow G^i_{K'}(S[i])$
4. $T \leftarrow (T[1], \ldots, T[t])$
5. **return** T

Algorithm
$\mathsf{XOR\text{-}GTM}[F_K].\mathrm{hash}(M)$

1. **for** $i = 1$ **to** t **do**
2. $\quad S[i] \leftarrow 0^n$
3. **for** $j = 1$ **to** m **do**
4. $\quad Z \leftarrow F^j_K(M[j])$
5. $\quad$ **for** $i = 1$ **to** t **do**
6. $\quad\quad$ **if** $\mathbf{H}_{i,j} = 1$
7. $\quad\quad\quad$ **then** $S[i] \leftarrow S[i] \oplus Z$
8. $S \leftarrow (S[1], \ldots, S[t])$
9. **return** S

Algorithm
$\mathsf{XOR\text{-}GTM}[F_K, G_{K'}].\mathrm{verify}(M', T')$

1. $\widehat{T} \leftarrow \mathsf{XOR\text{-}GTM}[F_K, G_{K'}].\mathrm{tag}(M')$
2. **if** $\widehat{T} = T'$ **return** $\top$
3. **else return** $\bot$

Algorithm
$\mathsf{XOR\text{-}GTM}[F_K, G_{K'}].\mathrm{verify\text{-}S}(M', T')$

1. **for** $i = 1$ **to** t **do**
2. $\quad S' \leftarrow G^{i,-1}_{K'}(T'[i])$
3. $\widehat{S} \leftarrow \mathsf{XOR\text{-}GTM}[F_K].\mathrm{hash}(M')$
4. **for** $i = 1$ **to** v **do**
5. $\quad \widehat{S}^R[i] \leftarrow \bigoplus_{j \in R_i} \widehat{S}[j]$
6. $\quad (S')^R[i] \leftarrow \bigoplus_{j \in R_i} S'[j]$
7. **for** $i = 1$ **to** v **do**
8. $\quad$ **if** $\widehat{S}^R[i] = (S')^R[i]$ **then** $B[i] \leftarrow \top$
9. $\quad$ **else** $B[i] \leftarrow \bot$
10. $B \leftarrow (B[1], B[2], \ldots, B[v])$
11. **return** B

Algorithm
$\mathsf{XOR\text{-}GTM}[F_K, G_{K'}].\mathrm{detect}(M', T')$
$//R_i = \{i\}$ for $i \in [\![t]\!]$

1. $\mathcal{P} \leftarrow [\![m]\!]$
2. **for** $i = 1$ **to** t **do**
3. $\quad S' \leftarrow G^{i,-1}_{K'}(T'[i])$
4. $\widehat{S} \leftarrow \mathsf{XOR\text{-}GTM}[F_K].\mathrm{hash}(M')$
5. **for** $i = 1$ **to** v **do**
6. $\quad \widehat{S}^R[i] \leftarrow \bigoplus_{j \in R_i} \widehat{S}[j]$
7. $\quad (S')^R[i] \leftarrow \bigoplus_{j \in R_i} S'[j]$
8. **for** $i = 1$ **to** v **do**
9. $\quad$ **if** $\widehat{S}^R[i] = (S')^R[i]$
10. $\quad\quad$ **then** $\mathcal{P} \leftarrow \mathcal{P} \setminus \mathbf{H}^R_i$
11. **return** $\mathcal{P}$

Fig. 1: XOR-GTM using $t \times m$ test matrix $\mathbf{H}$ and extension rule R with v elements.

5 Security Analysis

We show XOR-GTM is a secure MAC under the standard unforgeability notion [7,9], and more importantly, it is hard to forge the detection procedure, if F_K and $G_{K'}$ are PRF and tweakable PRP.

5.1 Security Notions

The first is Tag Vector Unforgeability (TVUF), which is essentially the same as the standard unforgeability of deterministic MACs. The second is Tag String Unforgeability (TSUF), a stronger notion of unforgeability. The third one is Decoder Unforgeability (DUF), which represents the hardness of fooling the naïve decoder to detect corruptions. To define them, we introduce several oracles. Following the syntax defined at Section 4.2, we consider a corruption detectable MAC, denoted by MAC_K, as a tuple $(\mathsf{KG}, \mathsf{Tag}, \mathsf{Ver}, \mathsf{SVer}, \mathsf{Det})$. We assume the key $K \in \mathcal{K}$ has been generated by $\mathsf{KG}(p)$ in advance, for a security parameter p.

Definition 2. *(Oracles) A tagging oracle $\mathcal{O}_T$ accepts M and returns $T = \mathsf{Tag}_K(M)$. The tag vector verification oracle $\mathcal{O}_V$, or simply the verification oracle, accepts $(M', T') \in \mathcal{M} \times \mathcal{T}$ and returns $\mathsf{Ver}_K(M', T')$. The tag string verification oracle $\mathcal{O}_{SV}$ accepts (M', T') and returns $\mathsf{SVer}_K(M', T')$. The detection oracle $\mathcal{O}_D$ accepts $(M', T') \in \{0,1\}^{*m} \times \mathcal{T}^t$ and returns $\mathsf{Det}_K(M', T')$.*

A query to $\mathcal{O}_T$ is called a tagging query and written as T-query. Queries to other oracles are called analogously.

Definition 3. *(TVUF) Let $\mathcal{A}_1$ be an adversary who (possibly adaptively) queries to $\mathcal{O}_T$ and $\mathcal{O}_V$. We say $\mathcal{A}_1$ forges if it receives 0 from $\mathcal{O}_V$ by querying (M', T') without making a tagging query M'. The advantage of $\mathcal{A}_1$ is defined as*

$$\mathbf{Adv}^{\mathsf{tvuf}}_{\mathsf{MAC}_K}(\mathcal{A}_1) \stackrel{\mathrm{def}}{=} \Pr[\mathcal{A}_1^{\mathcal{O}_T, \mathcal{O}_V} \text{ forges }].$$

Definition 4. *(TSUF) Let $\mathcal{A}_2$ be an adversary who queries to $\mathcal{O}_T$ and $\mathcal{O}_{SV}$. We say $\mathcal{A}_2$ forges if it receives $B = (B[1], \ldots, B[t])$ from $\mathcal{O}_{SV}$ indicating a* non-trivial tag string forgery. *That is, for an SV-query (M', T') and the corresponding response B from $\mathcal{O}_{SV}$, there exists $i \in [\![t]\!]$ such that $B[i] = 0$ and $(M' \ominus \mathbf{H}_i, T'[i]) \neq (M \ominus \mathbf{H}_i, T[i])$ holds for any (M, T) obtained from a previous T-query and its response. The advantage of $\mathcal{A}_2$ is defined as*

$$\mathbf{Adv}^{\mathsf{tsuf}}_{\mathsf{MAC}_K}(\mathcal{A}_2) \stackrel{\mathrm{def}}{=} \Pr[\mathcal{A}_2^{\mathcal{O}_T, \mathcal{O}_{SV}} \text{ forges }],$$

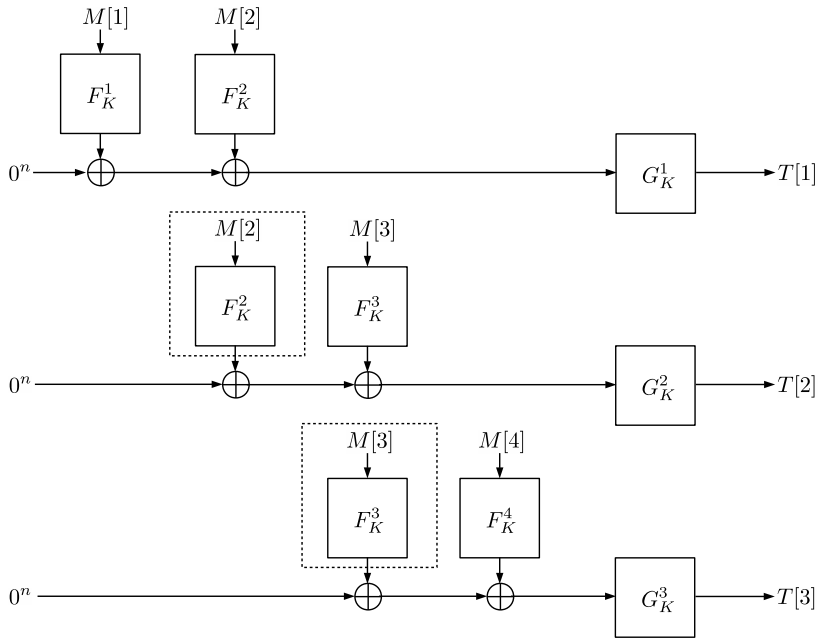

Fig. 2: XOR-GTM for $m = 4$ and $t = 3$ (we omit the first all-one row). The invocations of F in a dotted box can be avoided by caching.

Definition 5. *(DUF) Let $\mathcal{A}_3$ be an adversary who queries to $\mathcal{O}_T$ and $\mathcal{O}_D$. We assume $\mathcal{A}_3$ is d-*corrupting*, i.e. any D-query (M', T') satisfies (1) $T' = T$ holds for a previous T-query (M, T) (where M is called target message) and (2) $1 \le |\mathsf{diff}(M', M)| \le d$. We say $\mathcal{A}_3$ forges if $\mathcal{O}_D$ fails, that is, it returns $\mathcal{P} \neq \mathsf{diff}(M', M)$. We define*

$$\mathbf{Adv}^{\mathsf{duf}(d)}_{\mathsf{MAC}_K}(\mathcal{A}_3) \stackrel{\text{def}}{=} \Pr[\mathcal{A}_3^{\mathcal{O}_T, \mathcal{O}_D} \text{ forges }].$$

The security against tag vector forgery is measured by TVUF advantage, and we say $\mathsf{MAC}_{\mathbf{K}}$ is secure against tag vector forgery if $\mathbf{Adv}^{\mathsf{tvuf}}_{\mathsf{MAC}_{\mathbf{K}}}(\mathcal{A}_1)$ is sufficiently small for all practical adversaries. The security against tag string forgery and decoder forgery are defined similarly.

These notions are the same as Min15 except TSUF which is slightly different.

Notes on DUF. When two distinct M_i and M_j yield the same T, Definition 5 tells that the adversary wins if one of M_i or M_j be the target. This may not reflect the practical situation and it is also possible to modify the definition so that the target is always unique (say by querying a tuple (M, M') and the oracle computes $T' = T$ from M). In fact, a non-trivial tag collision breaks our scheme as it tells some non-trivial information on the outputs of F_K, and we count it as one of bad events in our provable security analysis (see Section 5.2). Thus both definitions have no significant difference in practice. See Section 5.4 for other discussions.

5.2 Provable Security Bounds

$\mathsf{XOR\text{-}GTM}[F_K, G_{K'}]$ is defined by the algorithms of Figure 1 (where KG is trivially defined and omitted): $\mathsf{Tag}_{\mathbf{K}} = \mathsf{XOR\text{-}GTM}[F_K, G_{K'}]$.tag, and $\mathsf{Ver}_{\mathbf{K}} = \mathsf{XOR\text{-}GTM}[F_K, G_{K'}]$.verify, and $\mathsf{SVer}_{\mathbf{K}} = \mathsf{XOR\text{-}GTM}[F_K, G_{K'}]$.verify-S, and $\mathsf{Det}_{\mathbf{K}} = \mathsf{XOR\text{-}GTM}[F_K, G_{K'}]$.detect, where $\mathbf{K} = (K, K')$.

We show the security bounds for $\mathsf{XOR\text{-}GTM}[F_K, G_{K'}]$ assuming $t \times m$ H and R (consisting of v elements) are sound and $\mathbf{H}^R$ is d-disjunct.

Theorem 1. *(TVUF security of XOR-GTM) For any $\mathcal{A}_1$ using q_t T-queries and q_v V-queries with time complexity τ, we have*

$$\mathbf{Adv}^{\mathsf{tvuf}}_{\mathsf{XOR\text{-}GTM}[F_K,G_{K'}]}(\mathcal{A}_1) \le \mathbf{Adv}^{\mathsf{prf}}_{F_K}(\mathcal{A}_F) + \mathbf{Adv}^{\mathsf{tprp}}_{G_{K'}}(\mathcal{A}_G) + \frac{tq^2+q_v}{2^n},$$

where $q = q_t + q_v$, and for some $\mathcal{A}_F$ using mq queries and $\tau' = O(\tau)$ time, and $\mathcal{A}_G$ using tq queries and $\tau'' = O(\tau)$ time.

Proof. The proof is mostly the same as [40]. We prove the security of $\mathsf{XOR\text{-}GTM}[\mathsf{R}_F, \widetilde{\mathsf{P}}_G]$ and combine it with a standard information-theoretic to computational conversion (e.g. see [3]).

Let R_G be the URF having the same domain as G. We consider indistinguishability of $\mathsf{XOR\text{-}GTM}[\mathsf{R}_F, \mathsf{R}_G]$ from the ideal primitive $\mathsf{R}_{\mathsf{XOR\text{-}GTM}}$, which is a set of t independent URFs, $\mathsf{R}^{(i)} : \{0,1\}^* \to \mathcal{M}$ for $i \in [\![t]\!]$. For query M, $\mathsf{R}_{\mathsf{XOR\text{-}GTM}}(M) = (\mathsf{R}^{(1)}(M(1)), \mathsf{R}^{(2)}(M(2)), \ldots, \mathsf{R}^{(t)}(M(t)))$, where $M(i)$s are determined by the same $\mathbf{H}$ as XOR-GTM. Let $\mathcal{A}$ be an adversary using q queries for distinguishing $\mathsf{XOR\text{-}GTM}[\mathsf{R}_F, \mathsf{R}_G]$ from $\mathsf{R}_{\mathsf{XOR\text{-}GTM}}$. The advantage is defined as

$$\mathbf{Adv}^{\mathsf{prf'}}_{\mathsf{XOR\text{-}GTM}[\mathsf{R}_F,\mathsf{R}_G]}(\mathcal{A}) \stackrel{\text{def}}{=} \mathbf{Adv}^{\mathsf{ind}}_{\mathsf{XOR\text{-}GTM}[\mathsf{R}_F,\mathsf{R}_G],\mathsf{R}_{\mathsf{XOR\text{-}GTM}}}(\mathcal{A}). \tag{8}$$

Let $S_i[j]$ be the j-th hash value in the i-th query M_i:

$$S_i[j] = \bigoplus_{k \in \mathbf{H}^R_j} \mathsf{R}_{F,j}(M_i[j]).$$

We also assume $\{S_i[j]\}_{i\in[\![q_t]\!],j\in[\![t]\!]}$ is also generated at $\mathsf{R}_{\mathsf{XOR\text{-}GTM}}$ following the same rule as above, involving dummy R_Fs. The generated $S_i[j]$s are always discarded.

Eq. (8) is bounded by slightly extending the security proof of PMAC1 (a TBC-based parallel MAC) [46]. In more detail, let $\mathsf{E}_{\mathrm{tvuf}}$ be the event that $S_i[k] = S_j[k]$ for some $1 \le i < j \le q$, $k \in [\![t]\!]$, $M_i(k) \ne M_j(k)$ (otherwise $S_i[k] = S_j[k]$ trivially holds). Given $\overline{\mathsf{E}_{\mathrm{tvuf}}}$, the outputs of $\mathsf{XOR\text{-}GTM}[\mathsf{R}_F, \mathsf{R}_G]$ is completely random except the trivial collision caused by the identical inputs, i.e. $M_i(k) = M_j(k)$, thus the output distribution is identical to that of $\mathsf{R}_{\mathsf{XOR\text{-}GTM}}$. This implies that

$$\mathbf{Adv}^{\mathsf{ind}}_{\mathsf{XOR\text{-}GTM}[\mathsf{R}_F,\mathsf{R}_G],\mathsf{R}_{\mathsf{XOR\text{-}GTM}}}(\mathcal{A}) \le \Pr_{\mathcal{A}^{\mathsf{R}_{\mathsf{XOR\text{-}GTM}}}}[\mathsf{E}_{\mathrm{tvuf}}]. \tag{9}$$

Without loss of generality we can consider $\mathcal{A}$ to be deterministic (no internal random coin), and because the output of $\mathsf{R}_{\mathsf{XOR\text{-}GTM}}$ does not tell anything about $S_i[j]$, any adaptive choice of queries does not help, and we can limit our focus to non-adaptive queries to invoke $\mathsf{E}_{\mathrm{tvuf}}$. The right hand side of Eq. (9) is thus bounded as

$$
\begin{aligned}
&\max_{M_1,\ldots,M_q} \Pr[\exists (i,j) \in [\![q]\!]^2, k \in [\![t]\!], i \neq j, M_i(k) \neq M_j(k) : S_i[k] = S_j[k]] \\
&\leq \max_{M_1,\ldots,M_q} \sum_{\substack{(i,j)\in[\![q]\!]^2, k\in[\![t]\!], i\neq j: \\ M_i(k)\neq M_j(k)}} \Pr[S_i[k] = S_j[k]] \\
&\leq \max_{M_1,\ldots,M_q} \sum_{\substack{(i,j)\in[\![q]\!]^2, k\in[\![t]\!], i\neq j: \\ M_i(k)\neq M_j(k)}} \Pr\left[\bigoplus_{h\in\mathbf{H}_k} \mathsf{R}_{F,h}(M_i[h]) = \bigoplus_{h\in\mathbf{H}_k} \mathsf{R}_{F,h}(M_j[h])\right] \\
&\leq \max_{M_1,\ldots,M_q} \sum_{\substack{(i,j)\in[\![q]\!]^2, k\in[\![t]\!], i\neq j: \\ M_i(k)\neq M_j(k)}} \Pr\left[\bigoplus_{\substack{h\in\mathbf{H}_k: \\ M_i[h]\neq M_j[h]}} \mathsf{R}_{F,h}(M_i[h]) \oplus \mathsf{R}_{F,h}(M_j[h]) = 0^n\right] \\
&\leq \max_{M_1,\ldots,M_q} \sum_{\substack{(i,j)\in[\![q]\!]^2, k\in[\![t]\!], i\neq j: \\ M_i(k)\neq M_j(k)}} \frac{1}{2^n} \\
&\leq t \cdot \binom{q}{2} \cdot \frac{1}{2^n} \qquad (10)
\end{aligned}
$$

as $M_i(k) \neq M_j(k)$ implies at least one $h \in \mathbf{H}_k^R$ such that $M_i[h] \neq M_j[h]$. Note that we do not have to count collision of type $S_i[k] = S_j[h]$ for $k \neq h$ thanks to the input difference in final R_G, i.e. $(k, S_i[k]) \neq (h, S_j[k])$. From Eq. (10) we have

$$\mathbf{Adv}^{\mathsf{prf'}}_{\mathsf{XOR\text{-}GTM}[\mathsf{R}_F,\mathsf{R}_G]}(\mathcal{A}) \leq \frac{tq^2}{2^{n+1}}.$$

We also have $\mathbf{Adv}^{\mathsf{ind}}_{\widetilde{\mathsf{P}}_G,\mathsf{R}_G}(\mathcal{A}) \leq tq^2/2^{n+1}$, thus a hybrid argument shows

$$\mathbf{Adv}^{\mathsf{prf'}}_{\mathsf{XOR\text{-}GTM}[\mathsf{R}_F,\widetilde{\mathsf{P}}_G]}(\mathcal{A}) \leq 2\frac{tq^2}{2^{n+1}} \leq \frac{tq^2}{2^n}.$$

Then we have a standard conversion from the definition of indistinguishability:

$$
\begin{aligned}
\mathbf{Adv}^{\mathsf{tvuf}}_{\mathsf{XOR\text{-}GTM}[\mathsf{R}_F,\widetilde{\mathsf{P}}_G]}(\mathcal{A}) &\leq \mathbf{Adv}^{\mathsf{ind}}_{\mathsf{XOR\text{-}GTM}[\mathsf{R}_F,\widetilde{\mathsf{P}}_G],\mathsf{R}_{\mathbf{XOR\text{-}GTM}}}(\mathcal{A}) + \mathbf{Adv}^{\mathsf{tvuf}}_{\mathsf{R}_{\mathbf{XOR\text{-}GTM}}}(\mathcal{A}) \\
&\leq \frac{tq^2}{2^n} + \frac{q_v}{2^n},
\end{aligned}
$$

where $\mathbf{Adv}^{\mathsf{tvuf}}_{\mathsf{R}_{\mathbf{XOR\text{-}GTM}}}(\mathcal{A}) \leq q_v/2^n$ is simply obtained by the fact that each component $\mathsf{R}^{(i)}$ is independent and $M(1) = M$ (from the fact that $\mathbf{H}_1$ is all-one), implying that the first tag $T[1]$ is independent and random for all q queries.

Additional note on the necessity of all-one row. One may think of removing the condition of all-one row in $\mathbf{H}$ and instead just requiring $\mathbf{H}^R$ to contain all-one row. However, this will not work as $v - t$ bottom rows of $\mathbf{H}^R$ do not contribute to verification. Moreover, even if we include these rows for check, in the same manner to detection query in DUF, the simple attack described at [40] still works. What is important for TVUF notion is to include the sum of all $F_i(M[i])$ encrypted by G taking *unique* tweak.

Theorem 2. *(TSUF security of XOR-GTM) For any $\mathcal{A}_2$ using q T-queries and q_v SV-queries with time complexity τ, we have*

$$\mathbf{Adv}^{\mathsf{tsuf}}_{\mathsf{XOR\text{-}GTM}[F_K,G_{K'}]}(\mathcal{A}_2) \le \mathbf{Adv}^{\mathsf{prf}}_{F_K}(\mathcal{A}_F) + \mathbf{Adv}^{\mathsf{tprp}}_{G_{K'}}(\mathcal{A}_G) + \frac{tq^2 + tq_v}{2^n},$$

where $q = q_t + q_v$, and for some $\mathcal{A}_F$ using mq queries and $\tau' = O(\tau)$ time, and $\mathcal{A}_G$ using tq queries and $\tau'' = O(\tau)$ time.

Proof. The proof is mostly the same as Theorem 1 by changing $\mathbf{Adv}^{\mathsf{tvuf}}$ to $\mathbf{Adv}^{\mathsf{tsuf}}$. We just evaluate $\mathbf{Adv}^{\mathsf{tsuf}}_{\mathsf{R}_{\mathsf{XOR\text{-}GTM}}}(\mathcal{A}) \le tq_v/2^n$ as the success condition for the adversary is to find $M'(j)$ for some $j \in [\![t]\!]$ such that $M'(j) \neq M_i(j)$ for all $i \in [\![q_t]\!]$. This implies that the bound is increased by t.

Theorem 3. *(DUF security of XOR-GTM) Let $\mathbf{H}^R$ be sound and d-disjunct. For any d-corrupting $\mathcal{A}_3$ using q_t T-queries and q_d D-queries with time complexity τ, we have*

$$\mathbf{Adv}^{\mathsf{duf}(d)}_{\mathsf{XOR\text{-}GTM}[F_K,G_{K'},\mathbf{H},R]}(\mathcal{A}_3) \le \mathbf{Adv}^{\mathsf{prf}}_{F_K}(\mathcal{A}_F) + \mathbf{Adv}^{\mathsf{prf}}_{G_{K'}}(\mathcal{A}_G) + \frac{vq^2 + vq_d}{2^n},$$

where $q = q_t + q_d$, holds for some $\mathcal{A}_F$ using $m(q + q_d)$ queries and $\tau' = O(\tau)$ time, and $\mathcal{A}_G$ using $v(q_t + q_d)$ queries and $\tau'' = O(\tau)$ time.

The security proof of DUF bound is described in the next section.

5.3 DUF Proof

We first consider a variant of DUF oracle, DUF′ oracle denoted by $\mathcal{O}_{D'}$, which takes the same input as $\mathcal{O}_D$ but returns the raw decoder input. That is, when $\mathcal{O}_{D'}$ takes (M', T'), $M' = (M'[1], \ldots, M'[m])$ and $T' = (T'[1], \ldots, T'[t])$, it returns $\widehat{B} = (\widehat{B}[1], \ldots, \widehat{B}[v])$ with $\widehat{B}[i] = \mathsf{cmp}(S'[i], \widehat{S}[i])$ and

$$\begin{aligned} S'[i] &= \widetilde{\mathsf{P}}^{-1}_{G,i}(T'[i]), \text{ for } i \in [\![t]\!] \\ S'[j] &= \bigoplus_{k \in R[j]} S'[k], \text{ for } j \in \{t+1, \ldots, v\} \\ \widehat{S}[i] &= \bigoplus_{j \in \mathbf{H}^R_i} \mathsf{R}_{F,j}(M'[j]). \end{aligned} \tag{11}$$

A query to $\mathcal{O}_{D'}$ will be called a D'-query. Let $\mathcal{P}$ be the output of naïve decoder taking $\widehat{B}$. An adversary of DUF′ game is said to win (forge) if $\mathcal{P} \neq \mathsf{diff}(M, M')$.

Since the naïve decoder is a public function of $\widehat{B}$, the adversary $\mathcal{A}'$ in DUF′ game can always simulate the adversary $\mathcal{A}$ in the original DUF game, using the same numbers of T- and D/D'-queries as $\mathcal{A}$. Hence we have

$$\mathbf{Adv}^{\mathsf{duf}(d)}_{\mathsf{XOR\text{-}GTM}}(\mathcal{A}) \le \mathbf{Adv}^{\mathsf{duf'}(d)}_{\mathsf{XOR\text{-}GTM}}(\mathcal{A}'), \tag{12}$$

where the latter term denotes the advantage under DUF′ game, which we want to bound.

Suppose a D'-query (M', T') is made after a T-query-response pair $(M, T = T')$ is obtained and $|\mathsf{diff}(M, M')| \in [\![d]\!]$ (i.e. the target message is M).

Let $B = (B[1], \ldots, B[v])$ with $B[i] = \mathsf{cmp}(M(i), M'(i))$, where $M(i) = M \ominus \mathbf{H}^R_i$ and $M'(i) = M' \ominus \mathbf{H}^R_i$ for convention. In other words, B tells the existence of differences between the subsequence of message items in M and M' specified by the v rows of $\mathbf{H}^R$. We observe that, if $\widehat{B} = B$ the adversary never wins as B represents the message difference without noise and the naïve decoder taking B will always be correct from the property of d-disjunctness of $\mathbf{H}^R$. Also B is computable by the adversary. This implies that the responses of D'-queries are essentially useless as the response that do not lead to win (forge) are always known in advance. With the same reasoning as [5], this shows that return values of D'-queries are useless for choosing

subsequent queries, hence the optimal adversary is considered to ignore the responses of D'-queries. Since responses are ignored, they are safely assumed to be made after the all T-queries.

From the above observation, the adversary needs to see $\widehat{B} \neq B$ to win. Since $B[i] = 0$ and $\widehat{B}[i] = 1$ cannot occur (the case $M(i) = M'(i)$ but $S[i] \neq S'[i]$ which is clearly impossible), we must have at least one $i \in [\![v]\!]$ such that $B[i] = 1$ and $\widehat{B}[i] = 0$.

Let $\mathsf{DirectGTM}[\mathsf{R}_F, \widetilde{\mathsf{P}}_G]$ use $v \times m$ test matrix $\mathbf{H}' = \mathbf{H}^R$ for $\mathbf{H}$ and R specified by $\mathsf{XOR\text{-}GTM}[\mathsf{R}_F, \widetilde{\mathsf{P}}_G]$. $\widetilde{\mathsf{P}}_G$ is assumed to have tweak space $[\![v]\!]$, by attaching $(v - t)$ independent URPs.

Let us consider TSUF security of $\mathsf{DirectGTM}[\mathsf{R}_F, \widetilde{\mathsf{P}}_G]$, where we bound the adversary to query $\mathcal{O}_{SV}$ that is allowed for DUF game. Thus any SV-query (M', T') has some previous T-query $(M, T = T')$. Since $v \times m$ $\mathbf{H}'$ is the test matrix for $\mathsf{DirectGTM}[\mathsf{R}_F, \widetilde{\mathsf{P}}_G]$, the expansion rule is simply $R' = (R'[1], \ldots, R'[v])$ with $R'[i] = \{i\}$. The oracle $\mathcal{O}_{SV}$ compares $T'[i]$ with $\widehat{T}[i] = \widetilde{\mathsf{P}}_G(i, \widehat{S}[i])$ to compute $\widehat{B}$ for all $i \in [\![v]\!]$. Since each tweak of $\widetilde{\mathsf{P}}_G$ implements a (keyed) permutation, comparing $T'[i]$ and $\widehat{T}[i]$ is equivalent to comparing $S'[i] = \widetilde{\mathsf{P}}_G^{-1}(i, T'[i])$ and $\widehat{S}[i]$ for any $i \in [\![v]\!]$.

Furthermore, this is also equivalent to decrypt only the first t tags out of v tags, obtain $S'[1], \ldots, S'[t]$, and apply Eq. (11) to derive $S'[t+1], \ldots, S'[v]$. Note that this equivalence holds because T' in the detection query is always a correct tag vector for M for which appeared at a T-query, as we restricted earlier. This means that the oracle $\mathcal{O}_{SV}$ ignores the last $v - t$ tags while the output $\widehat{B}$ is the same as the original TSUF game (but again this holds since the restriction above).

Now, we observe that the aforementioned game for $\mathsf{DirectGTM}[\mathsf{R}_F, \widetilde{\mathsf{P}}_G]$ (with a restricted adversary) is exactly the same as DUF game of $\mathsf{XOR\text{-}GTM}[\mathsf{R}_F, \widetilde{\mathsf{P}}_G]$ except that T-query returns $v - t$ additional tags. Since the computation rule of the first t tags are the same for $\mathsf{DirectGTM}[\mathsf{R}_F, \widetilde{\mathsf{P}}_G]$ and $\mathsf{XOR\text{-}GTM}[\mathsf{R}_F, \widetilde{\mathsf{P}}_G]$, this information only to increase the advantage. Therefore, we have

$$\mathbf{Adv}^{\mathsf{duf}'(d)}_{\mathsf{XOR\text{-}GTM}[\mathsf{R}_F, \widetilde{\mathsf{P}}_G]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{tsuf}}_{\mathsf{DirectGTM}[\mathsf{R}_F, \widetilde{\mathsf{P}}_G]}(\mathcal{A}') \tag{13}$$

for some $\mathcal{A}'$ using q_t T-queries and q_d SV-queries. Combining Eq. (12) and Theorem 2 with Eq. (13), we complete the proof.

5.4 Discussions on Decoder Unforgeability

As well as previous work [28,40], we assume that only the message is corrupted for defining DUF, which is more restrictive than the standard attack model for MACs, and is close to the security model of keyless hash functions. Note that when an i-th tag string is corrupted but the corresponding data (in our terminology, $M \ominus \mathbf{H}_i$) is intact, the verifier cannot decide whether the data is corrupted, or only the tag is corrupted, i.e., a false positive in the test outcome. Generally, the tags must be intact in order to prevent such false positives. Conversely, if the adversary has no limitation on tag corruption, there is no mean to exclude false positives for any corruption detectable MAC schemes including Min15. The avoidance of tag-only corruption is practical for some use cases. In a storage integrity protection system, MACs are applied to a large storage and the tags are usually stored in a small, trusted place (e.g. a secure hardware or an isolated server).

Meanwhile, it is also possible to extend our notions to capture the limited amount of tag corruption. This will require to extend the notion of disjunctness as studied in the context of CGT [41,14,50,42]. See also Section 3.5 of Min15.

6 Instantiations of XOR-GTM

6.1 Finding Useful Matrices

XOR-GTM suggests that, a d-disjunct matrix of low rank is desirable instead of small number of rows. However, the rank was rarely studied in the most of existing disjunct matrix constructions. Moreover, the state-of-the-art constructions tend to have a high rank. For example, we investigated the rank of matrices

from CRS [24] and STD [50] used by Min15. As far as we tried, the rank was around $0.95t$ to $0.90t$ for the matrix of t rows, hence only up to 10% reduction in communication. This phenomenon is more or less expected, as these matrices are designed to have a small number rows and not to have a small rank. Both are quite different goals.

In the following, we show several low-rank d-disjunct matrix constructions which can be used as $\mathbf{H}^R$ in XOR-GTM. The corresponding $\mathbf{H}$ is obtained as a basis matrix (i.e. a matrix obtained by the basis of row vectors of $\mathbf{H}^R$), and R is determined accordingly. Interestingly, all matrices are near-square, thus not the choice for the common applications of CGT. However, they achieve a smaller communication cost than any of DirectGTM.

6.2 Macula's Construction

Macula [37] proposed the following simple construction for d-disjunct matrix. Let (n, k, d) be a list of positive integers with $n > k > d$. Let $\left(\binom{n}{j}\right)$ be the family of subsets of $[\![n]\!]$ having cardinality j. Let $\phi_d : [\![\binom{n}{d}]\!] \to \left(\binom{n}{d}\right)$ and $\phi_k : [\![\binom{n}{k}]\!] \to \left(\binom{n}{k}\right)$ be the one-to-one mappings. Let $\mathbf{M}_{\mathrm{macula}}(n, k, d)$ be the $\binom{n}{d} \times \binom{n}{k}$ matrix defined as

$$\mathbf{M}_{\mathrm{macula}}(n, k, d)_{i,j} = \begin{cases} 1 & \text{if } \phi_d(i) \subset \phi_k(j) \\ 0 & \text{otherwise.} \end{cases}$$

It is shown that $\mathbf{M}_{\mathrm{macula}}(n, k, d)$ is d-disjunct [37]. Its column weight is $\binom{k}{d}$, and the row weight is $\binom{n-d}{k-d}$. When $\binom{n}{d} > \binom{n}{k}$, this construction has more rows than columns. However, the GF(2)-rank of $\mathbf{M}_{\mathrm{macula}}(n, k, d)$ can be (much) smaller than the number of columns.

Example 2. Let $(n, k, d) = (5, 3, 2)$. Then $\mathbf{M}_{\mathrm{macula}}(n, k, d)$ is

$$\begin{pmatrix} 1&1&1&0&0&0&0&0&0&0 \\ 1&0&0&1&1&0&0&0&0&0 \\ 0&1&0&1&0&1&0&0&0&0 \\ 0&0&1&0&1&1&0&0&0&0 \\ 1&0&0&0&0&0&1&1&0&0 \\ 0&1&0&0&0&0&1&0&1&0 \\ 0&0&1&0&0&0&0&1&1&0 \\ 0&0&0&1&0&0&1&0&0&1 \\ 0&0&0&0&1&0&0&1&0&1 \\ 0&0&0&0&0&1&0&0&1&1 \end{pmatrix}.$$

This 10×10 matrix has rank 6. Hence, it enables a non-trivial XOR-GTM of $(m, t, d) = (10, 7, 2)$ including additional all-one row for soundness. while there is no meaningful DirectGTM with 2-disjuct matrix, as lower bound of 2-disjunct exceeds 10 from (22).

We have more examples, such as 35×35 $\mathbf{M}_{\mathrm{macula}}(7, 4, 3)$ of rank 20 and 28×56 $\mathbf{M}_{\mathrm{macula}}(8, 6, 5)$ of rank 21, though we do not know the general rank formula.

6.3 Hadamard Matrix

We propose an instantiation of XOR-GTM, which we call *Hadamard MAC*.

Definition 6. *Let s be a positive integer, and let $m = 2^s - 1$. Let $\mathbf{M}_{\mathrm{hadamard}}(x)$ be the Sylvester-type Hadamard matrix of order x. Let Hadamard MAC be an instance of XOR-GTM with $\mathbf{H}^R$ set to the $m \times m$ matrix obtained by replacing the all -1 entries of $\mathbf{M}_{\mathrm{hadamard}}(2^s)$ with 0, and removing the (all-one) first row and column.*

Note that $\mathbf{H}^R$ is equivalent to a sub-code of the punctured Reed-Muller code.

Example 3. Let $s = 3$. Then $\mathbf{M}_{\mathrm{hadamard}}(8)$ and the $\mathbf{H}^R$ are described as

$$\mathbf{M}_{\mathrm{Hadamard}}(8) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}, \quad \mathbf{H}^R = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Lemma 1. *The rank of* $\mathbf{H}^R$ *is at most* $s + 1$.

Proof. Let $\mathbf{C}$ be a matrix obtained by a bit-complement of $\mathbf{H}^R$, i.e., $\mathbf{C}_{i,j} = \mathbf{H}^R_{i,j} \oplus 1$ for all i, j. We observe that $\mathbf{C}$ is a codebook of the simplex code excluding the all-zero codeword, which has a basis matrix of s rows [39]. Hence the rank of $\mathbf{C}$ is s. Let $(\mathbf{H}^R)'$ be $\mathbf{H}^R$ with an all-one row added. Row operations over $(\mathbf{H}^R)'$ can yield $\mathbf{C}$ with an all-one row added. Hence the rank of $(\mathbf{H}^R)'$ is at most the rank of $\mathbf{C}$ plus one. □

Theorem 4. $\mathbf{H}^R$ *is* 2*-disjunct for any* $s \geq 1$.

Proof. Since $\mathbf{H}^R$ is a bit-complement of $\mathbf{C}$, what we need to prove are

A) For any i-th column and any other k-th column of $\mathbf{C}$, there exists a row index h such that $(\mathbf{C}_{h,i}, \mathbf{C}_{h,k}) = (1, 0)$, or equivalently, no column of $\mathbf{H}^R$ is contained by another column, i.e., $\mathbf{H}^R_{*,i} \not\subseteq \mathbf{H}^R_{*,j}$ for any $i \neq j$.
B) For any distinct i-th and j-th columns and any other k-th column of $\mathbf{C}$, there exists a row index h such that $(\mathbf{C}_{h,i}, \mathbf{C}_{h,j}, \mathbf{C}_{h,k}) = (1, 1, 0)$.

As $\mathbf{C}$ is a codebook of simplex code excluding the all-zero codeword, $\mathbf{C}$ is symmetric. Moreover,

$$\mathbf{C}_i \oplus \mathbf{C}_j = \bigoplus_{i(h) \neq j(h) : h \in [\![s]\!]} \mathbf{C}_h = \mathbf{C}_{i \oplus j}, \tag{14}$$

holds for any distinct i and j, where $i(h)$ and $j(h)$ denote the h-th bit of i and j in the standard binary encoding. In addition, the simplex code has a constant weight of 2^{s-1} [39], and since it is a linear code, a sum of distinct pair of rows (or columns) also has a constant weight of 2^{s-1}. Therefore, combined with (14) we have

$$\mathrm{Hw}(\mathbf{C}_i) = 2^{s-1} \text{ for any } 1 \leq i \leq 2^s - 1, \tag{15}$$

$$\mathrm{Hw}(\mathbf{C}_i \oplus \mathbf{C}_j) = \mathrm{Hw}(\mathbf{C}_{i \oplus j}) = 2^{s-1} \text{ for any } 1 \leq i < j \leq 2^s - 1. \tag{16}$$

For any $\{j_1, j_2, \ldots, j_h\} \subseteq [\![m]\!]$ for $h \in [\![m]\!]$, and $b = (b_1, \ldots, b_h) \in \{0, 1\}^h$, let

$$\mathcal{S}_{j_1, j_2, \ldots, j_h}(b) = \{i \in [\![m]\!] : (\mathbf{C}_{i,j_1}, \mathbf{C}_{i,j_2}, \ldots, \mathbf{C}_{i,j_h}) = b\},$$

which denotes the set of row indexes that yield b as the row vector bound to j_1 to j_h-th columns. From (14) and (16) we observe

$$\begin{aligned} |\mathcal{S}_i(1)| &= 2^{s-1}, \\ |\mathcal{S}_{i,j}(1,1)| = |\mathcal{S}_{i,j}(0,1)| &= |\mathcal{S}_{i,j}(1,0)| = 2^{s-2} \end{aligned} \tag{17}$$

for any distinct $i, j \in [\![m]\!]$.

Now we are ready to prove the theorem. First, A) holds from (15), since this implies any column of $\mathbf{H}^R$ has a constant weight of $m - 2^{s-1}$, hence no column of $\mathbf{H}^R$ can be contained by another column. To prove B), we fix a pair of distinct column indexes, $i, j \in [\![m]\!]$. Then we need to show that

$$|\mathcal{S}_{i,j,k}(1,1,0)| \neq 0 \tag{18}$$

Table 1: The number of tags for DirectGTM using the known smallest 2-disjunct matrices [8], and that of Hadamard MAC. The latter sets parameter $s = \lceil \log m \rceil$.

m	100	400	900	1600	3600	14400
DirectGTM	21	31	38	44	51	63
Hadamard MAC	8	10	11	12	13	15

holds for any $k \notin \{i, j\}$. When $k = i \oplus j$, (18) holds since $|\mathcal{S}_{i,j,k}(1,1,0)| = |\mathcal{S}_{i,j}(1,1)| = 2^{s-2}$ holds from (14) and (17). When $k \neq i \oplus j$, let $\delta = i \oplus j$ and $k' = \delta \oplus k$. Since $\mathbf{C}_k \oplus \mathbf{C}_{k'} = \mathbf{C}_\delta$, we observe

$$\begin{aligned}
\mathcal{S}_{i,j}(1,1) &= \mathcal{S}_{i,j,\delta}(1,1,0) = \mathcal{S}_{i,j,\delta,k,k'}(1,1,0,0,0) \cup \mathcal{S}_{i,j,\delta,k,k'}(1,1,0,1,1) \\
\mathcal{S}_{i,j}(0,1) &= \mathcal{S}_{i,j,\delta}(0,1,1) = \mathcal{S}_{i,j,\delta,k,k'}(0,1,1,0,1) \cup \mathcal{S}_{i,j,\delta,k,k'}(0,1,1,1,0) \\
\mathcal{S}_{i,j}(1,0) &= \mathcal{S}_{i,j,\delta}(1,0,1) = \mathcal{S}_{i,j,\delta,k,k'}(1,0,1,0,1) \cup \mathcal{S}_{i,j,\delta,k,k'}(1,0,1,1,0) \\
\mathcal{S}_{i,j}(0,0) &= \mathcal{S}_{i,j,\delta}(0,0,0) = \mathcal{S}_{i,j,\delta,k,k'}(0,0,0,0,0) \cup \mathcal{S}_{i,j,\delta,k,k'}(0,0,0,1,1)
\end{aligned} \tag{19}$$

and the eight $\mathcal{S}_{i,j,\delta,k,k'}(b)$s appear in the above partition the whole row index set (i.e. all are distinct and the union is $[\![m]\!]$). Let

$$\begin{aligned}
x_1 &= |\mathcal{S}_{i,j,\delta,k,k'}(1,1,0,0,0)|, & x_2 &= |\mathcal{S}_{i,j,\delta,k,k'}(1,1,0,1,1)|, \\
x_3 &= |\mathcal{S}_{i,j,\delta,k,k'}(0,1,1,0,1)|, & x_4 &= |\mathcal{S}_{i,j,\delta,k,k'}(0,1,1,1,0)|, \\
x_5 &= |\mathcal{S}_{i,j,\delta,k,k'}(1,0,1,0,1)|, & x_6 &= |\mathcal{S}_{i,j,\delta,k,k'}(1,0,1,1,0)|, \\
x_7 &= |\mathcal{S}_{i,j,\delta,k,k'}(0,0,0,0,0)|, & x_8 &= |\mathcal{S}_{i,j,\delta,k,k'}(0,0,0,1,1)|.
\end{aligned}$$

Then we have

$$x_2 + x_8 = 2^{s-2} \text{ from } |\mathcal{S}_{k,k'}(1,1)| = 2^{s-2}, \tag{20}$$

$$x_4 + x_6 = 2^{s-2} \text{ from } |\mathcal{S}_{k,k'}(1,0)| = 2^{s-2},$$

$$x_2 + x_6 = 2^{s-2} \text{ from } |\mathcal{S}_{i,k'}(1,1)| = 2^{s-2}. \tag{21}$$

Suppose $x_1 = 0$. Then $x_2 = 2^{s-2}$ from (19)(17), implying $x_6 = x_8 = 0$ from (20)(21), thus $x_4 = 2^{s-2}$, $x_3 = 0$, $x_5 = 2^{s-2}$, and $x_7 = 2^{s-1} - 3 \cdot 2^{s-2}$ are derived from (17). This implies that $\mathbf{C}_k = \mathbf{C}_j$, hence a contradiction. Therefore, we have $x_1 \neq 0$ and (18) holds true. Thus, B) is proved. □

Comparison with DirectGTM. Erdös et al. [25] (also see [21, Theorem 7.5.9]) showed a shaper bound of $t_{\min}(2, m)$ than (1) and (2):

$$\log_{1.25} m < t_{\min}(2, m) < \log_{1.134} m. \tag{22}$$

Thus, $t_{\min}(2, m)$ is roughly in the range of $3.10 \log m$ to $5.51 \log m$. Hadamard MAC needs $\log(m+1) + 1$ tags. Hence the number of tags is fewer than that of DirectGTM with the smallest 2-disjunct matrix by a factor around 3.1 to 5.5. Since [25] did not show a concrete construction, we show a comparison of Hadamard MAC and DirectGTM using the known smallest 2-disjunct matrices [8] [13] in Table 1. The gain is in a similar range as the above estimation, 3 to 4.

6.4 Matrices from Projective Plane

The example of Section 6.3 is scalable in terms of m, however, still d is fixed to 2. In the following, we show two matrix classes that are scalable both for m and d. They are based on finite geometry. See e.g. [32] for the technical terms that will appear in the following descriptions.

Let s be a positive integer. Let $\mathbf{P}^{(s)}$ be a square matrix of $m = 2^{2s} + 2^s + 1$ rows, and is defined as a point-line incidence matrix for the two-dimensional finite projective plane over GF(2^s). To be more concrete,

each row (column) of $\mathbf{P}^{(s)}$ represents $2^{2s}+2^s+1$ points (lines) over the projective plane. All points and lines on the projective plane are indexed, and $\mathbf{P}^{(s)}_{i,j}$ is 1 if i-th point is on the j-th line, and 0 otherwise. It is known that the GF(2) rank of $\mathbf{P}^{(s)}$ is $t=3^s+1$ [49]. Its disjunctness is proved as follows.

Proposition 2. *$\mathbf{P}^{(s)}$ is 2^s-disjunct.*

Proof. $\mathbf{P}^{(s)}$ has the following properties [32]:

A) Each column (resp. row) has uniform weight 2^s+1;
B) Any two columns (resp. rows) have exactly one 1-entry in common.

We observe that by B), any two columns of $\mathbf{P}^{(s)}$ have exactly one intersection. It follows that any column has at most 2^s intersections with the union of any other 2^s columns, and thus it cannot be contained in that union since any column has weight 2^s+1 by A). This proves the proposition. □

An example of $\mathbf{P}^{(s)}$ is shown in Example 4. We call this instantiation XOR-GTM-PPI for the use of Projective-Plane-Incidence matrix. It uses t independent rows of $\mathbf{P}^{(s)}$ as $\mathbf{H}$ and defines R accordingly so that $\mathbf{H}^R=\mathbf{P}^{(s)}$. Therefore, XOR-GTM-PPI can detect $d=2^s=O(\sqrt{m})$ corruptions using $t=3^s+1=O(\sqrt{m})$ tags. This implies a significant improvement over DirectGTM, since t grows as $t=d^{\log 3}+1\approx d^{1.58}$ and $t_{\min}(d,m)=O(d^2\log m)$. Table 2 shows the profiles of the disjunct matrices obtained by the projective plane.

Example 4. $\mathbf{P}^{(1)}$ is a 7×7 matrix of rank 4 and disjunctness 2 which is described as follows (note: it depends on the polynomial defining the field).

$$\mathbf{P}^{(1)}=\begin{pmatrix}0&1&1&0&1&0&0\\0&0&1&1&0&1&0\\0&0&0&1&1&0&1\\1&0&0&0&1&1&0\\0&1&0&0&0&1&1\\1&0&1&0&0&0&1\\1&1&0&1&0&0&0\end{pmatrix}. \tag{23}$$

The matrix is a circulant with the first row (0110100). The first 4 rows in (23) are linearly independent, and they span the row space of $\mathbf{P}^{(1)}$.

6.5 Matrices from Affine plane

The projective plane-based matrix is quite efficient and scalable. However, since it is determined by a single parameter s, the space of possible (m,d,t) is sparse and thus may not fit in the real-world use cases.

In a different context, Kamiya [33] proposed a family of matrices derived from the finite geometry. He studied LDPC code (a class of linear error-correcting codes), however, we can use his proposal to realize a family of disjunct matrices with a large parameter flexibility. As well as the case of projective plane, let s be the parameter and let $\mathbf{A}^{(s)}$ be the point-line incidence matrix of affine plane. Since the affine plane has 2^{2s} points and $2^{2s}+2^s$ lines, $\mathbf{A}^{(s)}$ is of size $2^{2s}\times(2^{2s}+2^s)$. Each row and column has weight 2^s+1 and 2^s, respectively. We can show it is (2^s-1)-disjunct and has rank 3^s as well. Note that $\mathbf{A}^{(s)}$ itself has a similar profile as $\mathbf{P}^{(s)}$. Kamiya's matrix is a certain sub-matrix of $\mathbf{A}^{(s)}$ described as follows. Let $\mathcal{L}=\{L_1,L_2,\ldots,L_\ell\}$ be a set of ℓ lines passing through the origin in the affine plane over GF(2^s). The proposed matrix is denoted by $\mathbf{A}^{(s)}_\ell$, which is the incidence matrix of points contained in the union of the lines in $\mathcal{L}$ and lines having an intersection with a line in $\mathcal{L}$ at a non-origin point. The matrix $\mathbf{A}^{(s)}_\ell$ is then a sub-matrix of $\mathbf{A}^{(s)}$ of size $(\ell(2^s-1)+1)\times(2^{2s}-1+\ell)$, where $3\le\ell\le 2^s+1$. It is easy to see that each column of $\mathbf{A}^{(s)}_\ell$ has weight at least $\ell-1$ and any two columns have at most one 1-entry in common. Thus we have the following.

Proposition 3. *$\mathbf{A}^{(s)}_\ell$ is $(\ell-2)$-disjunct.*

Table 2: Disjunct matrices from projective plane. The matrices are square. The last column shows $t_{\min}(d, m)$, the best known lower bound for the number of rows of d-disjunct matrix with m columns, i.e. (3) for $d \leq 5$ and (2) for $d > 5$.

s	# rows	Rank	Disjunctness	Bound
1	7	4	2	7
2	21	10	4	21
3	73	28	8	56
4	273	82	16	222
5	1057	244	32	886
6	4161	730	64	3541
7	16513	2188	128	14162
8	65793	6562	256	56647
9	262657	19684	512	226586
10	1049601	59050	1024	906344
11	4196353	177148	2048	3625376
12	16781313	531442	4096	14501501
13	67117057	1594324	8192	58006002

Table 3: Disjunct matrix from affine plane. The last column shows $t_{\min}(d, m)$ in the same manner as Table 2.

s	# rows	# columns	Rank	Disjunctness	Bound
1	4	6	3	1	4
2	16	20	9	3	16
3	64	72	27	7	43
4	256	272	81	15	195
5	1024	1056	243	31	831
6	4096	4160	729	63	3431
7	16384	16512	2187	127	13942
8	65536	65792	6561	255	56205
9	262144	262656	19683	511	225702
10	1048576	1049600	59049	1023	904575
11	4194304	4196352	177147	2047	3621836
12	16777216	16781312	531441	4095	14494421
13	67108864	67117056	1594323	8191	57991841

Furthermore, it can be shown [33] that the GF(2)-rank of $\mathbf{A}_\ell^{(s)}$ is given as

$$\sum_{i=0}^{\lfloor \log \ell \rfloor} \binom{s}{i} 2^i + \sum_{i=\lfloor \log \ell \rfloor + 1}^{s} \binom{s}{i} \ell = 3^s - \sum_{i=\lfloor \log \ell \rfloor + 1}^{s} \binom{s}{i} (2^i - \ell).$$

See Table. 4 for some examples of $\mathbf{A}_\ell^{(s)}$s. As can be seen, this construction enables many parameter values not covered by the projective plane.

On rank optimality. The rank formulas of incident matrices built on finite geometry have been intensively studied for long years, in particular regarding the well-known Hamada's formula [29]. However, given m and d, whether these projective-plane/affine matrices achieve the minimality of the rank t is not known in general. To the best of our knowledge, these matrices achieve the lowest rank for given m and d.

Table 4: Rank of disjunct matrices from Kamiya's construction. Disjunctness is $\ell - 2$.

s	4	5	6
#rows	$15\ell + 1$	$31\ell + 1$	$63\ell + 1$
#columns	$\ell + 255$	$\ell + 1023$	$\ell + 4095$
Range of ℓ	Rank		
$3 \le \ell \le 4$	$11\ell + 9$	$26\ell + 11$	$57\ell + 13$
$4 \le \ell \le 8$	$5\ell + 33$	$16\ell + 51$	$42\ell + 73$
$8 \le \ell \le 16$	$\ell + 65$	$6\ell + 131$	$22\ell + 233$
$16 \le \ell \le 32$	81 ($\ell \le 17$)	$\ell + 211$	$7\ell + 473$
$32 \le \ell \le 64$	—	243 ($\ell \le 33$)	$\ell + 665$
$64 \le \ell \le 128$	—	—	729 ($\ell \le 65$)

7 Practical Aspects of XOR-GTM

7.1 Utilization of Matrix Structure

A straightforward implementation of XOR-GTM needs to store the whole test matrix. Since all of the matrices presented in Section 6 are highly structured, we can utilize these structures to significantly reduce the memory consumption of XOR-GTM. In particular, $\mathbf{H}^R$ of XOR-GTM-PPI is a circulant matrix thus we can greatly reduce the memory for implementation.

For example, $\mathbf{H}^R$ of XOR-GTM-PPI is an $m \times m$ circulant matrix and has constant row weight and column weight of $w = 2^s + 1$, and $\mathbf{H}$ consists of certain $t = 3^s + 1$ rows of $\mathbf{H}^R$. This allows a memory-efficient implementation for tag generation and verification and detection. Figure 3 shows the pseudocode of XOR-GTM-PPI of parameter s. In Figure 3, we record the column-index information that determines $\mathbf{H}$ from $\mathbf{H}^R$ as sequence $(a_1, \ldots, a_m)$, and encode $\mathbf{H}^R_{*,1}$ as $(b_1, \ldots, b_w)$ with $w = 2^s$. Then, the tags are computed with $O(w)$ memory. Assuming each PRF and TPRP computation needs a constant time, the time complexity is $O(m)$ which is inevitable as a MAC. For corruption detection, we additionally encode $\mathbf{H}^R_1$ as $(c_1, \ldots, c_w)$ for simplifying the eviction step, $\mathcal{P} \leftarrow \mathcal{P} \setminus \mathbf{H}^R_i$. Then the detection can be implemented by maintaining an m-bit sequence. This still needs $O(m)$ memory, however the actual constant can be much smaller than the entire data items. For example, when XOR-GTM-PPI with $s = 15$ is applied to the sectors of 4.4 TB HDD (See Section 7.2), the tags need 230 MB memory, and the eviction step needs 135 MB memory. We expect the memory efficiency of the detection step to be greatly improved by a dedicated decoder.

7.2 Comparison

We compare the commutation cost of XOR-GTM-PPI and DirectGTM. Figure 4 shows the ratio t/m for XOR-GTM-PPI and DirectGTM, where the latter is assumed to use a d-disjunct matrix achieving the lower bound of (2). Note that t/m represents the relative communication ratio from the trivial scheme using m tags, whose ratio is 1 (lower is better). Note that the plots of DirectGTM may be unachievable. We also show the conjectured lower bound (3) which was constantly 1 in the figure. The ratio of XOR-GTM-PPI quickly approaches to zero. For example, for $s = 1$ it is about 0.57 and for $s = 10$ it is about 0.056. In contrast, the communication ratio of DirectGTM is 1 up to $d = 5$ (as (3) holds for $d \le 5$) and is more than 0.8 even if s is large.

Numerical examples for storage integrity applications. Suppose we apply XOR-GTM-PPI to detect corruptions in the storages, such as HDDs or USB memories. Here, a data item represents the contents of a sector which is 4,096 bytes. When $s = 15$, the size of storage (HDD) amounts to about 4.4 TB , and XOR-GTM-PPI needs about 230 MB for storing the tags and is capable of up to 135 MB corruptions. The trivial scheme, which computes a tag for each sector, and DirectGTM using a disjunct matrix achieving (2) need about 17.2 GB and 14.8 GB for the tags respectively. In terms of the amounts of tags, the improvement factor from the trivial scheme is 74.82 for XOR-GTM-PPI, while only 1.15 for the DirectGTM. Table 5 shows more examples.

Algorithm XOR-GTM$[F_K, G_{K'}]$.tag(M):

1. $S \leftarrow$ XOR-GTM$[F_K]$.hash(M)
2. **for** $i = 1$ **to** t **do**
3. $\quad T[i] \leftarrow G^i_{K'}(S[i])$
4. $T \leftarrow (T[1], \ldots, T[t])$
5. **return** T

Algorithm XOR-GTM$[F_K, G_{K'}]$.hash(M):
// $\mathbf{H}_i = \mathbf{H}^R_i$ for all $i \in [\![t]\!]$
// $b_i \in (\!|m|\!)$, $i \in [\![w]\!]$: $\mathbf{H}^R_{*,1} = \{b_i + 1 : i \in [\![w]\!]\}$
// $w = 2^s + 1$

1. **for** $i = 1$ **to** t **do**
2. $\quad S[i] \leftarrow 0^n$
3. **for** $j = 1$ **to** m **do**
4. $\quad Z \leftarrow F^j_K(M[j])$
5. $\quad \mathcal{I} \leftarrow \{((b_k + (j-1)) \bmod m) + 1 : k \in [\![w]\!]\}$
6. $\quad$ **for all** $i \in \mathcal{I}$ **do** $S[i] \leftarrow S[i] \oplus Z$
7. $S \leftarrow S([1], \ldots, S[t])$
8. **return** S

Algorithm XOR-GTM$[F_K, G_{K'}]$.detect(M', T'):
//$c_i \in (\!|m|\!)$, $i \in [\![w]\!]$: $\mathbf{H}_1 = \{c_i + 1 : i \in [\![w]\!]\}$

1. $\mathcal{P} \leftarrow [\![m]\!]$
2. **for** $i = 1$ **to** t **do**
3. $\quad S' \leftarrow G^{i,-1}_{K'}(T'[i])$
4. $\widehat{S} \leftarrow$ XOR-GTM$[F_K]$.hash(M')
5. **for** $i = 1$ **to** v **do**
6. $\quad \widehat{S}^R[i] \leftarrow \bigoplus_{j \in R_i} \widehat{S}[j]$
7. $\quad (S')^R[i] \leftarrow \bigoplus_{j \in R_i} S'[j]$
8. **for** $i = 1$ **to** v **do**
9. $\quad$ **if** $\widehat{S}^R[i] = (S')^R[i]$
10. $\quad\quad \mathcal{S} \leftarrow \{((c_j + i - 1) \bmod m) + 1 : j \in [\![w]\!]\}$
11. $\quad\quad \mathcal{P} \leftarrow \mathcal{P} \setminus \mathcal{S}$
12. **return** $\mathcal{P}$

Fig. 3: XOR-GTM-PPI: XOR-GTM using projective-plane incidence matrix for $\mathbf{H}^R$.

Experimental Implementation. In a similar manner to Min15, we implemented XOR-GTM-PPI for tagging, verification and detection on a conventional server, using PMAC-AES for F and XEX-AES for G [46], for $s \in [\![5]\!]$. We have utilized the fact that the matrix $\mathbf{H}^R$ is circulant (see Example 4) for reducing memory. Since all the procedures are essentially $O(m\lceil x/128 \rceil)$ AES computations when $|M[i]| = x$ and simple linear operations, the computation cost is expected to be close to that of single PMAC-AES on entire M. We observed this when x is about 1 Mbyte. See Figure 6 for our preliminary result. Improving performance and extension to larger s are future work.

8 Experimental Implementation

An experimental implementation result for XOR-GTM-PPI is shown in Table 6. We use a server running Ubuntu 16.04 on Intel Xeon E5-2699 (Broadwell) v4 at 2.2 GHz. We use a variant of PMAC for PRF F and XEX [46] for TBC G, using AES-128 with AESNI instructions (single-core implementation), written in C with `gcc` 5.4.0. In our environment, PMAC runs at 5.2 cycles per byte for long inputs[3]. Table 6 shows the speed (in Cycles/Byte) for tag generation and a combined operation of verification and detection, with each message item size from 1KB to 1MB, for parameter $s \in [\![5]\!]$. For corruption detection, we insert random errors to the message including the number of corruptions (i.e. sample $v \xleftarrow{\$} [\![2^s]\!]$ and randomly choose v message items and randomly insert a bit error for them). As expected, all corrupted items were always correctly found. As Table 6 shows, the speed quickly approaches to the speed of F itself as message size gets larger, which verifies our claims. Optimized implementation for a larger s is a future plan.

9 Conclusions

We have described a new approach to corruption detectable MAC. As well as previous work, our XOR-GTM is based on the theory of combinatorial group testing (CGT). However, our scheme significantly reduces the

[3] The performance of single PMAC is worse than the typical implementation fully utilizing AES-NI (which is about 0.7 cycles per byte), which may be caused by some implementation details that we have not identified this moment.

Table 5: Numerical examples for XOR-GTM-PPI. The last column (improvement factor) shows the inverse ratio of Tag size to that of Trivial MAC.

Target: 4.4 TB HDD	Total tag size	Corrupted data	Imp. Factor
Trivial scheme	17.18 GB	No limit	1
DirectGTM	14.85 GB	135 MB	1.15
XOR-GTM-PPI ($s = 15$)	229.58 MB	135 MB	74.82
Target: 1.1 TB HDD	Total tag size	Corrupted data	Imp. Factor
Trivial scheme	4.29 GB	No limit	1
DirectGTM	3.71 GB	68 MB	1.15
XOR-GTM-PPI ($s = 14$)	76.52 MB	68 MB	56.06
Target: 4.3 GB Memory	Total tag size	Corrupted data	Imp. Factor
Trivial scheme	16.79 MB	No limit	1
DirectGTM	14.50 MB	5 MB	1.15
XOR-GTM-PPI ($s = 10$)	0.94 MB	5 MB	17.86

Table 6: Preliminary implementation results of XOR-GTM-PPI in cycles per input byte. Verification (verf) includes corruption detection. Environment: Ubuntu 16.04 on Intel Xeon E5-2699 (Broadwell) v4 at 2.2 GHz. Code written in C with gcc 5.4.0, using AESNI. Single PMAC runs at 5.2 cycles per byte for long inputs.

Size of each	$s = 1$		$s = 2$		$s = 3$		$s = 4$		$s = 5$	
message item	tag	verf	tag	verf	tag	verf	tag	verf	tag	verf
1 KB	14.6	20.8	16.6	20.7	14.8	22.5	20.67	23.5	15.4	15.5
2 KB	14.5	18.2	14.5	18.2	10.8	17.6	15.0	15.1	16.8	16.9
4 KB	13.5	16.9	10.1	16.9	12.9	14.0	6.3	10.5	12.6	12.7
1 MB	5.2	8.5	5.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2

number of MAC tags from the previous CGT-based schemes whose test matrices are (a variant of) d-disjunct. Besides, the computational cost is quite small, roughly the same as taking a single MAC for the whole data items.

The key idea is to utilize the linearity of the message hash values in Min15 [40] with a low-rank test matrix whose row span is d-disjunct. We also show several examples of matrices suitable for our scheme, where the matrices are near-square but the rank is small. All examples attain smaller communication cost than any of existing proposals of the same detection capability, no matter how the test matrix is. In particular, the effectiveness of matrices based on the finite geometry is remarkable: taking a storage integrity checking system for example, we show that we can reduce the storage for tags by a factor of 17 to 74 from the trivial scheme. In a similar manner to [40], we prove the security of XOR-GTM under the appropriate security notions we defined, and derived concrete security bounds based on the pseudorandomness of the components.

The idea presented here can be extended to various related problems. For example, as described at Section 5.4, the decoder unforgeability can be extended to allow false positive tests (by only corrupting a tag string while the corresponding message subsequence is intact), if the (extended) test matrix has a certain robustness. It will be interesting to consider a formal model capturing false positive tests and build an efficient and secure scheme for it. Another direction is to consider corruption-detectable cryptographic hash function or digital signatures as analogues of corruption-detectable MAC.

Finally, our idea is also directly applicable to aggregate MACs. Hirose and Shikata [30] proposed to take linear combinations of tags following d-disjunct matrix. By using a low-rank matrix whose row span is d-disjunct, we can reduce the number of tags. A formal analysis of security and concrete proposals will also be an interesting topic.

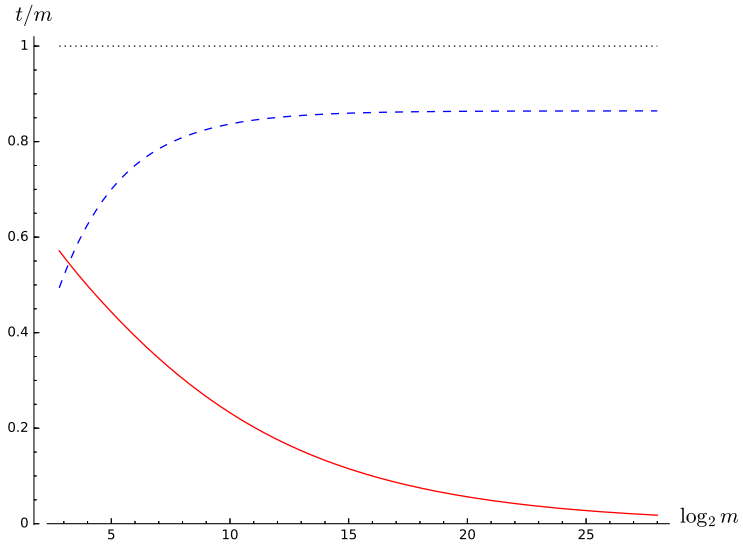

Fig. 4: Comparison of communication ratios. The red solid line : XOR-GTM-PPI. The blue dashed line : the lower bound of DirectGTM from (2). The black dotted line : the conjectured lower bound of DirectGTM from (3) (true for $d \leq 5$).

Acknowledgements

The authors would like to thank Hiroyasu Kubo, Nao Shibata, and Maki Shigeri for implementation. The authors also would like to thank anonymous reviewers of ESORICS 2019 and Eurocrypt 2018 for insightful comments.

References

1. Atallah, M.J., Frikken, K.B., Blanton, M., Cho, Y.: Private combinatorial group testing. In: AsiaCCS. pp. 312–320. ACM (2008)
2. Barg, A., Mazumdar, A.: Group Testing Schemes From Codes and Designs. IEEE Trans. Information Theory **63**(11), 7131–7141 (2017)
3. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: FOCS. pp. 394–403. IEEE Computer Society (1997)
4. Bellare, M., Goldreich, O., Goldwasser, S.: Incremental cryptography and application to virus protection. In: STOC. pp. 45–56. ACM (1995)
5. Bellare, M., Goldreich, O., Mityagin, A.: The Power of Verification Queries in Message Authentication and Authenticated Encryption. Cryptology ePrint Archive, Report 2004/309 (2004), `http://eprint.iacr.org/`
6. Bellare, M., Guérin, R., Rogaway, P.: XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions. In: CRYPTO. Lecture Notes in Computer Science, vol. 963, pp. 15–28. Springer (1995)
7. Bellare, M., Kilian, J., Rogaway, P.: The Security of the Cipher Block Chaining Message Authentication Code. J. Comput. Syst. Sci. **61**(3), 362–399 (2000)
8. van den Berg, E., Candès, E.J., Chinn, G., Levin, C.S., Olcott, P.D., Sing-Long, C.: A single-photon sampling architecture for solid-state imaging. CoRR **abs/1209.2262** (2012)
9. Black, J., Rogaway, P.: CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions. In: Bellare, M. (ed.) CRYPTO 2000. Lecture Notes in Computer Science, vol. 1880, pp. 197–215. Springer (2000)
10. Black, J., Rogaway, P.: A Block-Cipher Mode of Operation for Parallelizable Message Authentication. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. Lecture Notes in Computer Science, vol. 2332, pp. 384–397. Springer (2002)

11. Boneh, D., Crescenzo, G.D., Ostrovsky, R., Persiano, G.: Public Key Encryption with Keyword Search. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 3027, pp. 506–522. Springer (2004)
12. Bonis, A.D., Crescenzo, G.D.: Combinatorial Group Testing for Corruption Localizing Hashing. In: COCOON. Lecture Notes in Computer Science, vol. 6842, pp. 579–591. Springer (2011)
13. Brouwer, A.E.: Bounds for Constant Wight Codes, `http://www.win.tue.nl/~aeb/codes/Andw.html`
14. Cheraghchi, M.: Noise-resilient group testing: Limitations and constructions. Discrete Applied Mathematics **161**(1-2), 81–95 (2013)
15. Chor, B., Fiat, A., Naor, M.: Tracing Traitors. In: CRYPTO. Lecture Notes in Computer Science, vol. 839, pp. 257–270. Springer (1994)
16. Crescenzo, G.D., Arce, G.R.: Data Forensics Constructions from Cryptographic Hashing and Coding. In: IWDW. Lecture Notes in Computer Science, vol. 7128, pp. 494–509. Springer (2011)
17. Crescenzo, G.D., Ge, R., Arce, G.R.: Design and analysis of DBMAC, an error localizing message authentication code. In: GLOBECOM. pp. 2224–2228. IEEE (2004)
18. Crescenzo, G.D., Jiang, S., Safavi-Naini, R.: Corruption-Localizing Hashing. In: ESORICS. Lecture Notes in Computer Science, vol. 5789, pp. 489–504. Springer (2009)
19. Crescenzo, G.D., Vakil, F.: Cryptographic hashing for virus localization. In: WORM. pp. 41–48. ACM Press (2006)
20. Dorfman, R.: The Detection of Defective Members of Large Populations. The Annals of Mathematical Statistics **14**(4), 436–440 (1943)
21. Du, D., Hwang, F.: Combinatorial Group Testing and Its Applications. Applied Mathematics, World Scientific (2000)
22. Dýachkov, A.G., Rykov, V.V.: A Survey of Superimposed Code Theory. Problems of Control and Information Theory **12**(4), 229–242 (1983)
23. Emad, A., Milenkovic, O.: Poisson Group Testing: A Probabilistic Model for Boolean Compressed Sensing. IEEE Trans. Signal Processing **63**(16), 4396–4410 (2015)
24. Eppstein, D., Goodrich, M.T., Hirschberg, D.S.: Improved Combinatorial Group Testing Algorithms for Real-World Problem Sizes. SIAM J. Comput. **36**(5), 1360–1375 (2007)
25. Erdös, P., Frankl, P., Füredi, Z.: Families of Finite Sets in Which No Set Is Covered by the Union of Two Others. J. Comb. Theory, Ser. A **33**(2), 158–166 (1982)
26. Erdös, P., Frankl, P., Füredi, Z.: Families of finite sets in which no set is covered by the union of r others. Israel Journal of Mathematics **51**(1), 79–89 (1985)
27. Fang, J., L., J.Z., Yiu, S., Hui, L.C.: Hard Disk Integrity Check by Hashing with Combinatorial Group Testing. CSA 2009 pp. 1–6 (2009)
28. Goodrich, M.T., Atallah, M.J., Tamassia, R.: Indexing Information for Data Forensics. In: ACNS. Lecture Notes in Computer Science, vol. 3531, pp. 206–221 (2005)
29. Hamada, N.: On the p-Rank of the Incidence Matrix of a Balanced or Partially Balanced Incomplete Block Design and its Applications to Error Correcting Codes. Hiroshima Math Journal **3**, 153–226 (1973)
30. Hirose, S., Shikata, J.: Non-adaptive group-testing aggregate MAC scheme. In: ISPEC. Lecture Notes in Computer Science, vol. 11125, pp. 357–372. Springer (2018)
31. Inan, H.A., Kairouz, P., Özgür, A.: Sparse group testing codes for low-energy massive random access. In: Allerton. pp. 658–665. IEEE (2017)
32. Jr., E.F.A., Key, J.D.: Designs and Their Codes, Cambridge Tracts in Mathematics. Cambridge, U.K., vol. 103. Cambridge University Press (1992)
33. Kamiya, N.: High-Rate Quasi-Cyclic Low-Density Parity-Check Codes Derived From Finite Affine Planes. IEEE Trans. Information Theory **53**(4), 1444–1459 (2007)
34. Katz, J., Lindell, A.Y.: Aggregate Message Authentication Codes. In: CT-RSA. Lecture Notes in Computer Science, vol. 4964, pp. 155–169. Springer (2008)
35. Kautz, W.H., Singleton, R.C.: Nonrandom binary superimposed codes. IEEE Trans. Information Theory **10**(4), 363–377 (1964)
36. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. In: Yung, M. (ed.) CRYPTO 2002. Lecture Notes in Computer Science, vol. 2442, pp. 31–46. Springer (2002)
37. Macula, A.J.: A simple construction of d-disjunct matrices with certain constant weights. Discrete Mathematics **162**(1-3), 311–312 (1996)
38. Macula, A.J., Popyack, L.J.: A group testing method for finding patterns in data. Discrete Applied Mathematics **144**(1-2), 149–157 (2004)
39. MacWilliams, F., Sloane, N.: The Theory of Error-Correcting Codes. North-holland Publishing Company, 2nd edn. (1978)

40. Minematsu, K.: Efficient Message Authentication Codes with Combinatorial Group Testing. In: ESORICS (1). Lecture Notes in Computer Science, vol. 9326, pp. 185–202. Springer (2015)
41. Ngo, H.Q., Du, D.Z.: A Survey on Combinatorial Group Testing Algorithms with Applications to DNA Library Screening. DIMACS Series in Discrete Mathematics and Theoretical Computer Science (2000)
42. Ngo, H.Q., Porat, E., Rudra, A.: Efficiently Decodable Error-Correcting List Disjunct Matrices and Applications - (Extended Abstract). In: ICALP (1). Lecture Notes in Computer Science, vol. 6755, pp. 557–568. Springer (2011)
43. Oprea, A., Reiter, M.K.: Space-Efficient Block Storage Integrity. In: NDSS. The Internet Society (2005)
44. Oprea, A., Reiter, M.K.: Integrity Checking in Cryptographic File Systems with Constant Trusted Storage. In: USENIX Security Symposium. USENIX Association (2007)
45. Porat, E., Rothschild, A.: Explicit Nonadaptive Combinatorial Group Testing Schemes. IEEE Trans. Information Theory **57**(12), 7982–7989 (2011)
46. Rogaway, P.: Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 3329, pp. 16–31. Springer (2004)
47. Rudra, A.: Cse 709: Compressed sensing and group testing, part i (fall 2011 seminar)
48. Shangguan, C., Ge, G.: New Bounds on the Number of Tests for Disjunct Matrices. IEEE Trans. Information Theory **62**(12), 7518–7521 (2016)
49. Smith, K.J.C.: Majority Decodable Codes Derived from Finite Geometries. Institute of Statistics mimeo series **561** (1967)
50. Thierry-Mieg, N.: A New Pooling Strategy for High-Throughput Screening: the Shifted Transversal Design. BMC Bioinformatics **7**(28) (2006)
51. Ubaru, S., Mazumdar, A.: Multilabel Classification with Group Testing and Codes. In: ICML. Proceedings of Machine Learning Research, vol. 70, pp. 3492–3501. PMLR (2017)
52. Zaverucha, G.M., Stinson, D.R.: Group Testing and Batch Verification. In: ICITS. Lecture Notes in Computer Science, vol. 5973, pp. 140–157. Springer (2009)