On Algebraic Manipulation Detection Codes from Linear Codes and their Application to Storage Systems

Harshan, Jagadeesh; Oggier, Frederique

2015

Harshan, J., & Oggier, F. (2015). On Algebraic Manipulation Detection codes from linear codes and their application to storage systems. Information Theory Workshop, 64-68.

<https://hdl.handle.net/10356/82649>

<https://doi.org/10.1109/ITWF.2015.7360735>

© 2015 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at: [<http://dx.doi.org/10.1109/ITWF.2015.7360735>].

Downloaded on 26 Apr 2023 21:09:32 SGT

On Algebraic Manipulation Detection Codes from Linear Codes and their Application to Storage Systems

J. Harshan and Frédérique Oggier
Division of Mathematical Sciences
Nanyang Technological University, Singapore
Email: {jharshan, frederique}@ntu.edu.sg

Abstract—Algebraic Manipulation Detection (AMD) codes form a cryptographic primitive designed to detect data corruption of the form of an additive operation in an Abelian group. In this paper, we discuss the applicability of AMD codes to protect erasure-code based storage systems from a Byzantine adversary injecting fake data in the distributed storage system. We study a special class of AMD codes which relies on classical linear codes for its construction. We explore the design aspects of such AMD codes namely, (i) understanding its design criteria, (ii) studying the fundamental limits of such codes, to facilitate data integrity, and (iii) present some examples.

I. INTRODUCTION

Consider a distributed storage system, where data is stored redundantly across different nodes of the network for fault tolerance in case of failure of some nodes. It is by now well accepted by both research communities and industry players that erasure codes can be used to provide this redundancy, at a reasonable cost in terms of storage overhead. A plethora of research articles have focused on designing erasure codes for storage systems, focusing on the so-called repairability property, that of being able to reconstruct partial amount of the stored data without going through a whole decoding process (see e.g. [1] for a survey).

Once the reliability aspect of codes for distributed storage systems has been better understood, it is natural to question the security of such systems, under different threat models, starting from a passive adversary (or wiretapper) which is interested in reading the stored data, to an active (malicious) one which is keen on manipulating the existing data. In this paper, we will not be concerned with confidentiality, instead, we focus on a Byzantine adversary, which is trying to actively harm a storage system by inserting fake data. This scenario has already attracted some amount of attention, and the known results can be broadly categorized as follows: (1) capacity-like results, (2) schemes using integrity checks, (3) use of error correction codes instead of erasure codes.

Since the repair process in a distributed storage system can be seen from a network coding point of view, the repair bandwidth (or amount of bandwidth needed to perform a repair) has been studied using a classical max-flow min-cut approach. In that light, capacity-like results are characterizing the amount of storage and bandwidth needed to safely repair failed nodes in the presence of an active adversary (see e.g. [2]

for the case of a single failure repair, or [3] for simultaneous multiple failures). The repair of multiple failures can be done collaboratively, but when some of the nodes do not behave honestly, [3] showed that collaboration in the presence of corrupted nodes is actually worse than no collaboration at all.

A natural way to detect corrupted data is to use integrity tags, which was for example proposed in [3], or studied in [4] together with the joint design of storage codes. In both cases, the idea is to store some data, and attach to it its hash. It is known to be hard to forge data with a convincing hash, thus allowing to detect corrupted data. Finally, one could consider the code used for reliability to be an error-correcting code instead of an erasure code, which would allow the code to self-correct corrupted data (e.g. [5]). The drawbacks of this approach is the increase cost in storage, and the fact that only a small number of errors can be fixed. It is also possible to combine error-correction with a cryptographic primitive such as message authentication code as in [6].

In this paper, we address the suitability of a cryptographic primitive called *Algebraic Manipulation Detection (AMD) code* to protect storage systems from Byzantine adversaries. AMD codes [7] were primarily proposed as a primitive to be used in secret sharing schemes or fuzzy extractors. Fundamentally, these codes are designed such that algebraic manipulations on stored data (where corruption takes the form of an addition operation in an abelian group) can be detected with a given probability.

The contributions and the organization of this paper are summarized below. In Section II, we introduce AMD codes, and discuss why corruption in the form of addition in an abelian group suits well the distributed storage setting. We focus on a special type of AMD codes, constructed from linear codes. Our choice of such codes is attributed to the fact that this type of codes comprises a tag whose size is small relative to the size of the stored data, and also the scheme is *information theoretical secure*, in that breaking the scheme does not rely on the computational power of the adversary. Although AMD codes from linear codes have been proposed in [8] together with one such code, their constructions in general have not been studied yet. In Section III, we thus start their study, and propose several bounds and examples. In Section IV, we present an example of AMD code with non-trivial

error probability. Finally, we briefly discuss some aspects of the integration of AMD codes into a distributed storage system in Section V.

II. SYSTEM AND THREAT MODELS

Consider an erasure code based distributed storage system (which contains a replication based storage system as a particular case). Let $\mathbb{F}_q$ denote the finite with q elements, q a prime power. A data object to be stored, which can be represented as a vector with coefficients over $\mathbb{F}_{q^l}$, is first encoded into a codeword $\mathbf{x}$, and the coefficients x_i of this codeword will be distributed over a set of nodes. Depending on the scheme adopted, one may store one x_i per node, and since $x_i \in \mathbb{F}_{q^l}$, it can be seen as an l -dimensional vector over $\mathbb{F}_q$ by fixing an $\mathbb{F}_q$ -basis of $\mathbb{F}_{q^l}$. Alternately, one may store a subset of x_i per node, in which case the node is storing several l -dimensional vectors over $\mathbb{F}_q$. Irrespectively of the chosen strategy, the encoded pieces corresponding to the original data object will be stored over a set of nodes, and each node is assumed to then store a vector $\mathbf{v}$ of length m with coefficients in $\mathbb{F}_q$, that is $v_i \in \mathbb{F}_q$, $i = 1, \dots, l$. The length m will be equal to l if only one coefficient is stored, or to multiple of l otherwise.

We are interested in securing storage systems in the presence of a Byzantine adversary, capable of directly altering the stored content, or corrupting the communication links.

The adversary knows that every storage node will contain an l -dimensional vector $\mathbf{v}$ which contains information about a given data object, he also knows which security mechanism, if any, may protect this data. In the context of storage systems, he can also try to take advantage of the repair process during which nodes will acquire data to replace that of failed nodes, to inject corrupted data.

It is expected that the adversary will choose to inject a vector of length l and coefficients in $\mathbb{F}_q$, either on a link or in a node, to increase the success of corruption. For a data node, he will inject a vector $\mathbf{w}$ to replace the legitimate content $\mathbf{v}$. If he injects $\mathbf{w}$ on a link, a node participating in a repair process will receive $\mathbf{w}$ (instead of a legitimate vector $\mathbf{v}'$), which he may keep in a buffer for further computation, or immediately use to compute a linear combination of its own data and $\mathbf{w}$.

We thus without loss of generality write that the corruption at the node will be of the form

$$\mathbf{w} = \mathbf{v} + \Delta(\mathbf{v})$$

where $\mathbf{v}$ is a legitimate vector, and $\Delta(\mathbf{v})$ captures how far the corrupted vector $\mathbf{w}$ is from $\mathbf{v}$. This captures the above attacks on both the communication links and the node content.

The question of security is now that of finding a way to attach some tag to $\mathbf{v}$, to make it unlikely for the adversary to forge a convincing tag for $\mathbf{w}$. For this purpose, we will study the use of algebraic manipulation detection (AMD) codes.

Definition 1: [7], [8] An (M_1, M_2) -algebraic manipulation detection (AMD) code is a probabilistic encoding function $E : \mathcal{S} \rightarrow \mathcal{G}$ from a set $\mathcal{S}$, with $|\mathcal{S}| = M_1$ into a finite abelian group $\mathcal{G}$, with $|\mathcal{G}| = M_2$, together with a decoding function

$D : \mathcal{G} \rightarrow \mathcal{S} \cup \{\perp\}$ such that $D(g) = s$ if $g = E(s)$ for some $s \in \mathcal{S}$ and $\perp$ otherwise.

Definition 2: [7], [8] An (M_1, M_2) -AMD code is strongly ϵ -secure if for any $s \in \mathcal{S}$, and for δ sampled from $\mathcal{G}$ according to some distribution independent of $E(s)$ when given s , the probability that $D(E(s) + \delta) = s' \neq s$ is at most ϵ .

Since we want a storage node to store both the data and the tag that will ensure its integrity, we are interested in a systematic AMD code.

Definition 3: [7], [8] An (M_1, M_2) -AMD code is systematic if the set $\mathcal{S}$ is actually a group, and the encoding is of the form

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

and $\mathcal{G}_1, \mathcal{G}_2$ are groups (so that $\mathcal{G} = \mathcal{S} \times \mathcal{G}_1 \times \mathcal{G}_2$ is just a group obtained as direct product of the groups $\mathcal{S}, \mathcal{G}_1, \mathcal{G}_2$).

Let us translate Definition 3 in the context of distributed storage systems. The set $\mathcal{S}$ for us is given by $\mathbb{F}_q^m$ (where $M_1 = q^m$), which clearly has an abelian group structure since it is a vector space, and we will have to encode the data object $\mathbf{v}$ into $E(\mathbf{v}) = (\mathbf{v}, i, f(i, \mathbf{v})) \in \mathbb{F}_q^m \times \mathcal{G}_1 \times \mathcal{G}_2$, where $\mathcal{G}_1, \mathcal{G}_2$ and f are to be determined. With this, we get a (M_1, M_2) -AMD code, where $M_1 = q^m$ and $M_2 = q^m |\mathcal{G}_1| |\mathcal{G}_2|$.

For this scheme, a Byzantine adversary has the ability of replacing $E(\mathbf{v})$ by $E(\mathbf{v}) + \delta \in \mathcal{G}$, with $\delta = (\Delta(\mathbf{v}), g_1, g_2) \in \mathcal{G}$, so that

$$\begin{aligned} E(\mathbf{v}) + \delta &= (\mathbf{v}, i, f(i, \mathbf{v})) + (\Delta(\mathbf{v}), g_1, g_2) \\ &= (\mathbf{v} + \Delta(\mathbf{v}), i + g_1, f(i, \mathbf{v}) + g_2). \end{aligned}$$

This manipulation corresponds to our scenario of replacing $\mathbf{v}$ by $\mathbf{w} = \mathbf{v} + \Delta(\mathbf{v})$, however in this case, it is also allowing the adversary to corrupt the tag the way he wishes. The decoding function captures the security feature, in that, if we decode an encoding of $\mathbf{v} \in \mathbb{F}_q^m$, we should get $\mathbf{v}$ back. However, when trying to decode an element which is not a valid encoding, the decoder outputs an error ($\perp$), revealing that the data is corrupted. The adversary will manage a successful forgery if

$$E(\mathbf{v}) + \delta = E(\mathbf{v}')$$

and thus we want the probability that he manages this to be small. Formally, when δ is sampled to some distribution independent of $E(\mathbf{v})$, the probability that $E(\mathbf{v}) + \delta = E(\mathbf{v}')$ must be bounded by ϵ , or alternatively written the probability that

$$D(E(\mathbf{v}) + \delta) = D(E(\mathbf{v}')) = \mathbf{v}'$$

for $\mathbf{v}' \neq \mathbf{v}$ should be bounded by ϵ .

In the next section, we discuss a special class of AMD codes from linear codes. In particular, we will describe how linear codes can play the role of the function $f(i, x)$ in Definition 3.

III. CONSTRUCTIONS FROM LINEAR CODES

Let $\mathbf{G}$ be the generator matrix of a linear (N, m) code over $\mathbb{F}_q$. The following systematic AMD code was proposed [8]:

$$E : \mathbb{F}_q^m \rightarrow \mathbb{F}_q^m \times \mathbb{Z}_N \times \mathbb{F}_q, \quad \mathbf{v} \mapsto (\mathbf{v}, i, [\mathbf{v}\mathbf{G}]_i)$$

where $\mathbb{Z}_N$ is the set of integers modulo N , and $[\mathbf{vG}]_i$ means the i th coordinate of the codeword $\mathbf{vG}$. By convention, the first entry is indexed by $i = 0$. In words, this AMD code is a random entry from a codeword in the linear code $\mathcal{C}$ which is defined by the generator matrix $\mathbf{G}$. This is thus an $(q^m, q^{m+1}|\mathbb{Z}_N|)$ -AMD code.

The distance of relevance for $\mathcal{C}$ is not the usual Hamming distance, but instead the maximum of the number of times an element of $\mathbb{F}_q$ occurs as an entry of a codeword $\mathbf{c} \in \mathcal{C}$, that is

$$\mu(\mathbf{c}) := \max_{a \in \mathbb{F}_q} |i, c_i = a|,$$

where c_i denotes the i th component of $\mathbf{c}$. This induces a notion of similarity between two codewords $\mathbf{c} \neq \mathbf{c}'$, in that two codewords are "more similar" with respect to this measure when

$$\max_{a \in \mathbb{F}_q} |i, (\mathbf{c} - \mathbf{c}')_i = a|$$

increases. Note that this is not a mathematical distance, since it cannot be 0. Over a code $\mathcal{C}$, we thus get

$$\mu(\mathcal{C}) = \max_{\mathbf{c} \neq \mathbf{c}' \in \mathcal{C}} \max_{a \in \mathbb{F}_q} |i, (\mathbf{c} - \mathbf{c}')_i = a|.$$

Note that as usual, whenever $\mathcal{C}$ is linear code, $\mathbf{c} - \mathbf{c}' = \mathbf{c}'' \in \mathcal{C}$, thus instead of looking at any pair, it is enough to look at each non-zero codeword.

Let $\text{rot}_t(\mathbf{c})$ be a cyclic shift of the coefficients of a codeword $\mathbf{c}$ by t positions (to the right). Further, let $\text{cl}(\mathcal{C}) := \cup_{t \in \mathbb{Z}_N} \text{rot}_t(\mathcal{C})$. The main result known so far regarding the security of an AMD code created using a linear code is as follows.

Theorem 1: [8, Theorem 8.2.1] The above (q^m, Nq^{m+1}) -AMD code E is ϵ -secure for

$$\epsilon = \frac{\mu(\text{cl}(\mathcal{C}))}{N}, \quad \mu(\mathcal{C}) = \max_{\mathbf{c} \neq \mathbf{c}' \in \mathcal{C}} \max_{a \in \mathbb{F}_q} |i, (\mathbf{c} - \mathbf{c}')_i = a|$$

if for all $t \in \mathbb{Z}_N$, $t \neq 0$, $\text{rot}_t(\mathcal{C}) \cap \mathcal{C} = \mathbf{0}$, and $\epsilon = 1$ otherwise.

In order to construct AMD codes based on the criterion in Theorem 1, we first need to better understand $\mu(\mathcal{C})$, $\text{cl}(\mathcal{C})$ and $\mu(\text{cl}(\mathcal{C}))$. They are studied in the following subsections.

A. About $\mu(\mathcal{C})$

We start with some obvious observations about $\mu(\mathcal{C})$.

Lemma 1: If $\mathcal{C}$ is a linear (N, m) code, then

$$\mu(\mathcal{C}) \geq m.$$

Proof: Suppose that the code is systematic, then the first block of a generator matrix $\mathbf{G}$ contains the identity matrix, and when computing $\mathbf{uG}$, for $\mathbf{u} = (1, 1, \dots, 1) \in \mathbb{F}_q^m$, we get a codeword for which 1 is repeated m times. Now if the code is not systematic, it still has dimension m , thus we can find m columns of $\mathbf{G}$ which are linearly independent. Let us label them $i_1, \dots, i_m$. When $\mathbf{u}$ runs through all possible m -uples, so will the entries $i_1, \dots, i_m$ of $\mathbf{uG}$, and therefore the vector $(1, 1, \dots, 1) \in \mathbb{F}_q^m$ will appear in a codeword, thus proving that $\mu(\mathcal{C}) \geq m$. ■

The best that we can hope for is thus $\mu(\mathcal{C}) = m$ for $\mathcal{C}$ an m -dimensional linear code.

Note that $\mu(\mathcal{C})$ is a lower bound for $\mu(\text{cl}(\mathcal{C}))$, since in order to compute $\mu(\text{cl}(\mathcal{C}))$, one has to compute $\mathbf{c} - \mathbf{c}'$ for $\mathbf{c} \in \mathcal{C}$ and $\mathbf{c}'$ in some $\text{rot}_t(\mathcal{C})$. Thus when picking $\mathbf{c}' = \mathbf{0}$, we get $\mu(\mathcal{C})$, and as a result

$$\mu(\mathcal{C}) \leq \mu(\text{cl}(\mathcal{C})).$$

This in turns gives a lower bound on the security parameter ϵ :

$$\frac{\mu(\mathcal{C})}{N} \leq \epsilon = \frac{\mu(\text{cl}(\mathcal{C}))}{N}.$$

Given an m -dimensional code $\mathcal{C}$, we noted above that the best case scenario is to have $\mu(\mathcal{C}) = m$. In that case, in order to understand better the behaviour of μ , it could be interesting to figure out how large N is allowed to grow. Of course, the more N grows, the larger the number of shifted codes $\text{rot}_t(\mathcal{C})$ as well, there may thus be a later trade-off in how large N should be, but for now, let us see over some small alphabet sizes, how large can N grow, while keeping $\mu(\mathcal{C}) = m$. A trivial upper bound on N is

$$N \leq m|\mathbb{F}_q|,$$

since any longer codeword will be forced to have at least one element of $\mathbb{F}_q$ repeated more than m times.

This upper bound seems far from tight in general.

Example 1: Over $\mathbb{F}_2$, consider a simple parity check code. Then the codewords are

$$(0, 0, 0), (1, 0, 1), (0, 1, 1), (1, 1, 0).$$

One could obtain some "degenerate" version of this code by adding a zero to every coordinate, and reach the bound $m|\mathbb{F}_2| = 4$.

Example 2: A numerical search through systematic linear codes of dimension $m = 2$ over $\mathbb{F}_4$ yielded a code length of 6, with for example the following generator matrix:

$$\begin{bmatrix} 0 & 1 & w & 0 & 1 & w \\ 1 & 0 & w & w & 1 & 0 \end{bmatrix}$$

for w a primitive element. Here $6 < m|\mathbb{F}_4| = 8$.

B. About $\text{rot}_t(\mathcal{C}) \cap \mathcal{C}$

We next would like to have the property that $\text{rot}_t(\mathcal{C}) \cap \mathcal{C} = \mathbf{0}$ for all $t \in \mathbb{Z}_N$, $t \neq 0$. We show how to translate this property on the generator matrix of a linear code.

Let us write a codeword $(c_1, \dots, c_N)$ of $\mathcal{C}$ as $\mathbf{uG}$ for $\mathbf{G}$ an $m \times N$ generator matrix of $\mathcal{C}$. Let $\text{rot}_t(\mathcal{C})$ be the code obtained by shifting the coordinates of the codewords of $\mathcal{C}$ by t positions. That means, if $\mathbf{G}_t$ denotes the generator matrix of $\text{rot}_t(\mathcal{C})$, then $(c_1, \dots, c_N)$ belongs to both $\mathcal{C}$ and $\text{rot}_t(\mathcal{C})$ if and only if $\mathbf{uG} = \mathbf{vG}_t$ for some $\mathbf{v}$. Further, it is straightforward to show that

$$\mathbf{uG} = \mathbf{vG}_t \iff \mathbf{uG} - \mathbf{vG}_t = \mathbf{0} \iff (\mathbf{u}, -\mathbf{v}) \begin{bmatrix} \mathbf{G} \\ \mathbf{G}_t \end{bmatrix} = \mathbf{0}.$$

Since $\begin{bmatrix} \mathbf{G} \\ \mathbf{G}_t \end{bmatrix}$ is a $2m \times N$ matrix, having $\mathbf{uG} = \mathbf{vG}_t$ means we are looking for a non-zero vector $(\mathbf{u}, -\mathbf{v})$ in its kernel.

Therefore, to make sure its kernel is trivial, the rank of $\begin{bmatrix} \mathbf{G} \\ \mathbf{G}_t \end{bmatrix}$ has to be $2m$. In other words, to have the property $\text{rot}_t(\mathcal{C}) \cap \mathcal{C} = \mathbf{0}$ for all $t \in \mathbb{Z}_N, t \neq 0$, we have to make sure that $\begin{bmatrix} \mathbf{G} \\ \mathbf{G}_t \end{bmatrix}$ has row rank $2m$ for every $t \in \mathbb{Z}_N, t \neq 0$.

Lemma 2: For $\text{rot}_t(\mathcal{C}) \cap \mathcal{C}$ to be trivial for all $t \in \mathbb{Z}_N, t \neq 0$, it is necessary that the dimension m of $\mathcal{C}$ is less or equal to $N/2$.

Proof: From the above observation, if $m > N/2$, then the matrix

$$\begin{bmatrix} \mathbf{G} \\ \mathbf{G}_t \end{bmatrix}$$

has for rank at most N , and by the rank nullity theorem, its kernel is necessarily non-trivial. Thus $\mathcal{C}$ intersects with $\text{rot}_t(\mathcal{C})$ for some t . ■

Suppose then that $\mathbf{G}$ is the generator matrix of a linear code of dimension m less than half its length N and let $\mathbf{G}_t$ be the generator matrix corresponding to $\text{rot}_t(\mathcal{C})$. If

$$\begin{bmatrix} \mathbf{G} \\ \mathbf{G}_t \end{bmatrix}$$

has rank $2m$ for every $t \bmod N$, then $\text{rot}_t(\mathcal{C}) \cap \mathcal{C}$ is trivial.

Example 3: Here is an example of linear code of dimension 2 and length 4 defined over $\mathbb{F}_8 = \{0, 1, w, w^2, w^3 = w + 1, w^4 = w^2 + w, w^5 = w^2 + w + 1, w^6 = w^2 + 1\}$. Take the generator matrix

$$\mathbf{G} = \begin{bmatrix} w & 0 & 1 & 0 \\ 0 & w^2 & 0 & 1 \end{bmatrix}.$$

Then

$$\begin{bmatrix} \mathbf{G} \\ \mathbf{G}_1 \end{bmatrix} = \begin{bmatrix} w & 0 & 1 & 0 \\ 0 & w^2 & 0 & 1 \\ 0 & w & 0 & 1 \\ 1 & 0 & w^2 & 0 \end{bmatrix},$$

for the first rotation (shift),

$$\begin{bmatrix} \mathbf{G} \\ \mathbf{G}_2 \end{bmatrix} = \begin{bmatrix} w & 0 & 1 & 0 \\ 0 & w^2 & 0 & 1 \\ 1 & 0 & w & 0 \\ 0 & 1 & 0 & w^2 \end{bmatrix},$$

for the second shift, and

$$\begin{bmatrix} \mathbf{G} \\ \mathbf{G}_3 \end{bmatrix} = \begin{bmatrix} w & 0 & 1 & 0 \\ 0 & w^2 & 0 & 1 \\ 0 & 1 & 0 & w \\ w^2 & 0 & 1 & 0 \end{bmatrix},$$

for the last shift. All the obtained matrices are of rank 4.

C. About $\mu(\text{cl}(\mathcal{C}))$

Given an m -dimensional linear code $\mathcal{C}$, we showed above that

$$m \leq \mu(\mathcal{C}) \leq \mu(\text{cl}(\mathcal{C})).$$

Recall that

$$\text{cl}(\mathcal{C}) = \cup_{t \in \mathbb{Z}_N} \text{rot}_t(\mathcal{C}) = \cup_{t \in \mathbb{Z}_N, t \neq 0} \text{rot}_t(\mathcal{C}) \cup \mathcal{C}.$$

One could optimistically hope to reach $m = \mu(\text{cl}(\mathcal{C}))$. We will argue next that this cannot be.

Lemma 3: For an m -dimensional linear code $\mathcal{C}$, for which $\text{rot}_t(\mathcal{C}) \cap \mathcal{C} = \mathbf{0}$, for all $t \neq 0$, we have

$$\mu(\text{cl}(\mathcal{C})) \geq 2m.$$

Proof: To compute $\mu(\text{cl}(\mathcal{C}))$, we consider $c - c'$ for $c \neq c' \in \text{cl}(\mathcal{C})$. If $c, c' \in \mathcal{C}$, we get $\mu(\mathcal{C})$. Now if $c \in \mathcal{C}$ and $c' \in \text{rot}_t(\mathcal{C}), t \neq 0$, we may write

$$\begin{aligned} c - c' &= \mathbf{u}\mathbf{G} - \mathbf{u}'\mathbf{G}_t \\ &= (\mathbf{u}, -\mathbf{u}') \begin{bmatrix} \mathbf{G} \\ \mathbf{G}_t \end{bmatrix}. \end{aligned}$$

Now under the constraint that $c' \in \text{rot}_t(\mathcal{C}), t \neq 0$, as shown in the proof of Lemma 2, it must be that the matrix formed by $\mathbf{G}$ and $\mathbf{G}_t$ must be full rank, and in fact can be seen as the generator matrix of a new linear code, this time of dimension $2m$. Note that there is no constraint relating $\text{rot}_t(\mathcal{C})$ and $\text{rot}_{t'}(\mathcal{C})$ for $t \neq t', t, t' \neq 0$. Thus the matrix similarly obtained from $\mathbf{G}_t$ and $\mathbf{G}_{t'}$ could have a priori an arbitrary rank between m and $2m$. ■

IV. EXAMPLE AMD CODE FOR $m > 1$

We first recall the AMD code proposed in [8]. It is an AMD code obtained using a 1-dimensional linear code over $\mathbb{F}_q$. The corresponding AMD code is given by

$$E : \mathbb{F}_q \rightarrow \mathbb{F}_q \times \mathbb{Z}_q \times \mathbb{F}_q, v \mapsto (v, i, [v\mathbf{G}]_i),$$

where $\mathbf{G} \in \mathbb{F}_q^{1 \times N}$ and $N = q$ such that the elements of $\mathbf{G}$ are distinct. It has been shown in [8, Theorem 8.3.1] that this (q, q^3) -AMD code is ϵ -secure with $\epsilon = \frac{1}{q}$. Apart from the above trivial example with $m = 1$, a general question on the possibility of constructing AMD codes with larger m has also been asked in [8].

Towards answering that question, we present an AMD code for $m > 1$ that was obtained through numerical search via computer simulations. The above bounds and examples suggest that getting AMD code for $m > 1$ and small ϵ is not that easy. The objective of this section is to showcase the existence of an AMD code for $m = 2$ such that $\epsilon < 0.5$. The proposed code is over the finite field $\mathbb{F}_{17} = \{0, 1, \dots, 16\}$ where the arithmetic operations are over integers modulo 17. Its length is $N = 16$.

For the finite field $\mathbb{F}_{17}$, the messages are of the form $\mathbf{v} \in \mathcal{S} = \mathbb{F}_{17}^2$. Further, the generator matrix $\mathbf{G} \in \mathbb{F}_{17}^{2 \times 16}$ for the AMD code is chosen as in (1) (shown at the top of this page). The corresponding $(17^2, 17^3 \times 16)$ AMD code is given by

$$E : \mathbb{F}_{17}^2 \rightarrow \mathbb{F}_{17}^2 \times \mathbb{Z}_{16} \times \mathbb{F}_{17}, v \mapsto (\mathbf{v}, i, [\mathbf{v}\mathbf{G}]_i).$$

This code satisfies the following properties:

- The matrix

$$\begin{bmatrix} \mathbf{G} \\ \mathbf{G}_t \end{bmatrix} \in \mathbb{F}_{17}^{4 \times 16},$$

has rank 4 for any $t \in \{1, 2, \dots, 15\}$, where $\mathbf{G}_t$ represents a matrix obtained by rotating the rows of $\mathbf{G}$ by t components.

$$\mathbf{G} = \begin{bmatrix} 16 & 8 & 2 & 15 & 4 & 1 & 4 & 11 & 4 & 10 & 12 & 6 & 11 & 15 & 13 & 1 \\ 10 & 4 & 5 & 1 & 8 & 15 & 2 & 8 & 7 & 1 & 6 & 1 & 6 & 4 & 15 & 6 \end{bmatrix} \quad (1)$$

Fig. 1. The matrix $\mathbf{G} \in \mathbb{F}_{17}^{2 \times 16}$ in (1) generates the AMD code in Section IV.

- The parameter $\mu(\text{cl}(\mathcal{C})) = 7$, where $\mathcal{C} = \{\mathbf{v}\mathbf{G} \mid \mathbf{v} \in \mathcal{S}\}$.

From Theorem 1, the above properties can be used to show that probability of deception a Byzantine adversary is upper bounded by $\epsilon = 0.43$ (which is less than the trivial bound 0.5). Note that since q is not a power of 2, it would take some work to use this AMD code in practice. This leaves open the question of finding good AMD codes for arbitrary m, N , and preferably q a power of 2.

V. REMARKS ON SYSTEM INTEGRATION

In this section, we provide some remarks about possible integration of AMD codes into distributed storage systems. First, if we look at available cryptographic techniques in the literature, there are usually two ways to perform integrity checks, falling into the usual dichotomy between computational security (e.g. signatures, hash functions) and information theoretical security (e.g. authentication codes). The underlying trade-off between the two is to obtain security levels which does not rely on computational assumption. In our setting, one can use error correcting codes as authentication codes for information theoretical security. However, to attain a desired security level, heavy price has to be paid in terms of size of the tag, something not desirable for storage systems, where the storage cost is a very important parameter. In comparison with error correcting codes, an advantage of AMD codes based on linear codes is that they promise information theoretical security without too heavy a cost in terms of tag.

Suppose one would like to integrate AMD codes into their storage systems. Then, such codes are better suited to cases where the adversary is typically outside the distributed storage system, and is trying to harm it by injecting fake data. The case of nodes inside the network behaving in a rogue manner may need some adjustments, since they then have access to their own tags. To handle this scenario, a central entity (or master node) can be in charge of not only assigning the stored data, but also its corresponding tag. From this point of view, it may make sense to actually keep the index $i \in \mathbb{Z}_N$ at the master node, rather than at the node itself. This will result in reduced storage space at the nodes, in addition, can also offload the burden of integrity check since not every system assumes that storage nodes are equipped with computational ability. From this perspective, it makes sense that the central entity, which is orchestrating among other things the repair processes, will make the integrity check every now and then.

VI. FUTURE WORK

We proposed algebraic manipulation detection (AMD) codes to help secure erasure-code based distributed storage systems, thanks to their ability to detect additive alterations

in abelian groups. We looked at one family of AMD codes relying on linear codes, which has the advantage to provide information theoretical security at a low price in terms of tag length. This gave raise to interesting coding theory questions on their own, in terms of characterizing linear codes that provide good AMD codes. From system integration perspective, the AMD code for $m = 1$ [8] seems a good fit for practical distributed storage systems, as it provides smaller length tags relative to the message length. Some interesting directions for future work involves

- possible refinement to the bounds on probability of deception given in Theorem 1. It is intuitive that there is a higher possibility of obtaining small $\mu(\text{cl}(\mathcal{C}))$ by fixing N and increasing q , and hence, we believe that the current upper bound ϵ in Theorem 1 can be improved by incorporating the field size q in it,
- a deeper study of linear codes to be used as underlying structure for AMD codes, and finally
- the search for good AMD codes for arbitrary m, N , and preferably q a power of 2.

VII. ACKNOWLEDGMENTS

This work is supported by the MoE Tier-2 grant eCode: erasure codes for data center environments.

REFERENCES

- [1] F. Oggier, A. Datta, "Coding Techniques for Repairability in Networked Distributed Storage Systems," *Foundations and Trends in Communication and Information Theory*, 2013.
- [2] S. Pawar, S. El Rouayheb and K. Ramchandran, "Securing Dynamic Distributed Storage Systems against Eavesdropping and Adversarial Attacks," *IEEE Trans. on Information Theory*, Vol. 57, No. 9, September 2011.
- [3] F. Oggier, A. Datta, "Byzantine fault tolerance of regenerating codes", *Peer-to-Peer Computing (P2P)*, 2011.
- [4] Y. S. Han, R. Zheng, W. H. Mow, "Exact Regenerating Codes for Byzantine Fault Tolerance in Distributed Storage", *INFOCOM 2012*.
- [5] K. V. Rashmi, N. B. Shah, K. Ramchandran, P. Vijay Kumar, "Regenerating Codes for Errors and Erasures in Distributed Storage", *IEEE Int. Symposium on Information Theory (ISIT)*, 2012.
- [6] H. C. H. Chen, P. P. C. Lee, "Enabling Data Integrity Protection in Regenerating-Coding-Based Cloud Storage: Theory and Implementation", *IEEE Trans. on Parallel and Distributed Systems*, Vol. 25, No. 2, June 2013.
- [7] R. Cramer, Y. Dodis, S. Fehr, C. Padro, D. Wichs, "Detection of Algebraic Manipulation with Applications to Robust Secret Sharing and Fuzzy Extractors", *EUROCRYPT 2008*.
- [8] E. Jongsma, "Algebraic Manipulation Detection Codes", Bachelor Thesis, 2008.