IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 17, 2020, accepted November 15, 2020, date of publication December 1, 2020,
date of current version December 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3041638

Aggregate Message Authentication Code Capable of Non-Adaptive Group-Testing

SHOICHI HIROSE[1] AND JUNJI SHIKATA[2,3], (Member, IEEE)
[1]Faculty of Engineering, University of Fukui, Fukui 910-8507, Japan
[2]Graduate School of Environment and Information Sciences, Yokohama National University, Yokohama 240-8501, Japan
[3]Institute of Advanced Sciences, Yokohama National University, Yokohama 240-8501, Japan

Corresponding author: Shoichi Hirose (hrs_shch@u-fukui.ac.jp)

This work was supported by the Research and Development for Expansion of Radio Wave Resources funded by the Ministry of Internal Affairs and Communications, Japan.

ABSTRACT We introduce group-testing aggregate message authentication code (GTA MAC) and provide its formal study. We first specify its syntax and security requirements. Then, we present a scheme of generic construction which applies non-adaptive group-testing to aggregate MAC. We also confirm the security of the generic construction based on that of underlying aggregate MAC and a useful property of matrices representing non-adaptive group-testing. In addition, we instantiate the generic construction using the aggregate MAC scheme proposed by Katz and Lindell or a scheme using a cryptographic hash function for aggregating tags. Finally, we present some implementation results to show the effectiveness of our proposed GTA MAC.

INDEX TERMS Group testing, message authentication, provable security.

I. INTRODUCTION

A. BACKGROUND

The number of IoT (Internet of Things) devices is increasing, and there will be an enormous number of devices connected to networks including 5G in the near future. Even in such a situation, it is required to realize efficient communications or data transmissions in an authenticated manner. Therefore, it is important to study lightweight and secure authentication systems that can be deployed in such a situation.

Message authentication code (MAC) is one of the most fundamental cryptographic primitives, and it can be used as a lightweight cryptographic primitive for message authentication. For authenticated communication using MAC, a tag is attached to each message to detect tampering of the message. The tag is computed with a cryptographic symmetric-key primitive called a MAC function such as HMAC [1], [2] and CMAC [3], [4]. However, one-to-one authenticated communication using MAC requires an enormous number of tags that is proportional to that of communicating IoT devices in the network.

Aggregate MAC is a cryptographic primitive which can compress tags on multiple messages into a short aggregate tag [5]. It is possible to verify the validity of the multiple messages only with the shorter tag. One may think of use of aggregate MAC to reduce the total amount of tag-size compared to one-to-one authenticated communication using MAC. However, in general, it is impossible to identify invalid messages once the multiple messages are judged invalid with respect to the aggregate tag.

The purpose of the paper is to study aggregate MAC that has the following functionality: multiple tags generated by MAC can be compressed into aggregate tags whose total size is smaller than that of the tags and invalid messages are correctly identified from the aggregate tags. To realize aggregate MAC with such functionality, it is expected that techniques of group testing [6] can be utilized. Group testing is applied to a set of items each of which are either negative (valid) or positive (invalid). It assumes that a test can be run on multiple items and that its result is negative if all the items are negative and positive otherwise. If the number of positive items is not so large, by selecting items for each test properly, one can identify positive items more efficiently than by simply testing one by one. The group testing is called adaptive if one can choose items for a new test after seeing the result of the previous test and is called non-adaptive otherwise.

B. CONTRIBUTION

We initiate formal study of group-testing aggregate MAC (GTA MAC).

The associate editor coordinating the review of this manuscript and approving it for publication was Giacomo Verticale.

First, we give formal descriptions of its syntax and security requirements. The security properties we require of GTA MAC are unforgeability and identifiability. Unforgeability is an extended notion of that of standard MAC: A given message should be judged invalid unless a legitimate user produces its tag. Identifiability is a characteristic notion for GTA MAC: (In)valid pairs of a message and a tag should be exactly idetified by group-testing.

Then, we present a scheme of generic construction for GTA MAC: It simply combines aggregate MAC specified by Katz and Lindell [5] with non-adaptive group-testing. We show that the generic construction enables us to reduce the security of GTA MAC to that of aggregate MAC and a well-known property of matrices representing group-testing.

We also discuss instantiations of the generic construction using the aggregate MAC scheme by Katz and Lindell and a scheme using a cryptographic hash function for aggregating tags.

Finally, we present some implementation results on generation of group-testing matrices and performance of the proposed GTA MAC.

This paper is the full version of our conference paper [7]. In the formalization of this paper, the GTA MAC enables the users to use multiple group-testing matrices, while it enables them to use only a single group-testing matrix in [7]. For aggregate MAC, soundness is introduced as a security requirement in this paper, and accordingly, proofs of theorems are revised. Minor errors in some of the proofs are also fixed. In addition, some implementation results are added.

C. RELATED WORK

Katz and Lindell [5] initiated formal study of aggregate MAC. They formalized its syntax and security and proposed a scheme with a proof of its security on the assumption that the underlying MAC function is unforgeable. Their scheme simply uses a MAC function for generating tags and aggregates them by bitwise XOR.

Eikemeier *et al.* [8] proposed sequential aggregate MAC and formalized its security requirement. They also presented a scheme using a pseudorandom function and a pseudorandom permutation with a proof of its security. A different type of sequential aggregate MAC scheme was proposed by Sato, Hirose and Shikata [9], [10]. Their scheme aggregates tags without using secret keys of the users.

Ma and Tsudik [11] proposed forward-secure sequential aggregate MAC, which was followed by Ma and Tsudik [12] and by Hirose and Kuwakado [13]. Forward-secure sequential aggregate MAC may be useful for secure logging.

Group testing is also applied to MAC by Goodrich *et al.* [14], Minematsu [15], and Minematsu and Kamiya [16]. Their schemes are different from ours in that aggregating tags from multiple users is out of their scope. The scheme proposed by Minematsu [15] is based on PMAC [17], [18] and makes it possible to reduce the amount of computation to compute multiple tags for group testing. The scheme proposed by Minematsu and Kamiya [16] makes it possible to reduce the total number of tags for group testing.

Sato and Shikata [19] proposed an aggregate MAC scheme with adaptive group-testing functionality. Thus, their scheme is interactive.

After our proposal [7], Ogawa *et al.* [20] proposed a GTA MAC scheme in the same setting as ours. Their scheme reduces the total number of tags substantially but identifies invalid messages probabilistically.

D. ORGANIZATION

We introduce MAC functions and cryptographic hash functions together with a few notations in Section II. We describe aggregate MAC in Section III. We give a formal description of its syntax and security and present the scheme by Katz and Lindell and a scheme which aggregates tags using a cryptographic hash function. In Section IV, we first describe non-adaptive group testing and then formalize the syntax and security of GTA MAC. We present a simple generic construction of GTA MAC and discuss its security in Section V. In Section VI, we present two instantiations of the generic construction using aggregate MAC schemes described in Section III. We show some implementation results in Section VII. We give a concluding remark in Section VIII.

II. PRELIMINARIES

Let $s \twoheadleftarrow \mathcal{S}$ represent selection of an element s uniformly at random from a set $\mathcal{S}$. Concatenation of sequences x and y is denoted by $x\|y$.

A. MAC FUNCTION

A MAC function is a cryptographic symmetric-key primitive used for generating a message authentication code called a tag for a given message. It is a keyed function $f : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$, where $\mathcal{K}$ is a set of keys, $\mathcal{M}$ is a set of messages and $\mathcal{T}$ is a set of tags. $f(K, \cdot)$ is often denoted by $f_K(\cdot)$.

A MAC function is required to satisfy unforgeability. Let $\mathfrak{G}^{\mathrm{mac}}_{f,\mathbf{A}}$ be a game played by an adversary $\mathbf{A}$ against f concerning unforgeability. In this game, $\mathbf{A}$ is given a pair of oracles f_K and V_K, where $K \twoheadleftarrow \mathcal{K}$ and is able to make queries adaptively to both of them. For a query $M \in \mathcal{M}$, f_K returns $f_K(M)$. For a query $(M, T) \in \mathcal{M} \times \mathcal{T}$, V_K returns $\top$ if $f_K(M) = T$ and $\perp$ otherwise. $\mathbf{A}$ is not allowed to ask (M, T) to V_K once it asks M to f_K. $\mathfrak{G}^{\mathrm{mac}}_{f,\mathbf{A}}$ outputs 1 if $\mathbf{A}$ succeeds in getting $\top$ from V_K and 0 otherwise. The advantage of $\mathbf{A}$ is defined as

$$\mathrm{Adv}^{\mathrm{mac}}_{f}(\mathbf{A}) \triangleq \Pr\left[\mathfrak{G}^{\mathrm{mac}}_{f,\mathbf{A}} = 1\right]. \tag{1}$$

f is informally said to be a secure MAC function or satisfy unforgeability if $\mathrm{Adv}^{\mathrm{mac}}_{f}(\mathbf{A})$ is negligibly small for any efficient $\mathbf{A}$.

B. CRYPTOGRAPHIC HASH FUNCTION

A cryptographic hash function takes as input a sequence with substantially arbitrary length and returns a sequence with fixed length. It is often simply called a hash function. It is used for almost all cryptographic schemes and is required various kinds of properties on security. Among them, a random oracle and collision resistance are relevant.

Let $H : \{0,1\}^* \to \{0,1\}^\tau$ be a cryptographic hash function. It is called a random oracle if it returns a sequence chosen uniformly at random from $\{0,1\}^\tau$ for a new input. A random oracle returns the same sequence for the same input sequence. It is of course an ideal assumption that H is a random oracle and it is called the random oracle model [21].

For collision resistance, let $\mathbf{A}$ be an adversary against H. The advantage of $\mathbf{A}$ against H is defined as

$$\mathrm{Adv}_H^{\mathrm{col}}(\mathbf{A}) \triangleq \Pr[(M, M') \leftarrow \mathbf{A}(H) : M \neq M' \wedge H(M) = H(M')]. \qquad (2)$$

H is informally said to satisfy collision resistance if $\mathrm{Adv}_H^{\mathrm{col}}(\mathbf{A})$ is negligibly small for any efficient $\mathbf{A}$.

Actually, the definition of collision resistance is not precise theoretically, since H should be a function chosen at random from a sufficiently large set of hash functions.

III. AGGREGATE MAC

A. SYNTAX

An aggregate MAC scheme is defined to be a tuple of algorithms $\mathsf{AM} \triangleq (\mathsf{KG}, \mathsf{Tag}, \mathsf{Agg}, \mathsf{Ver})$ associated with a set $\mathcal{I}$ of IDs, a set $\mathcal{K}$ of keys, a set $\mathcal{M}$ of messages, a set $\mathcal{T}$ of tags and a set $\mathcal{T}_{\mathrm{A}}$ of aggregate tags.

- The key generation algorithm KG is a randomized algorithm. It takes $id \in \mathcal{I}$ as input and returns $k_{id} \in \mathcal{K}$ for id. Namely, $(id, k_{id}) \leftarrow \mathsf{KG}(id)$.
- The tagging algorithm Tag is a deterministic algorithm. It takes $k \in \mathcal{K}$ and $m \in \mathcal{M}$ as input and returns $t \in \mathcal{T}$. Namely, $t \leftarrow \mathsf{Tag}(k, m)$.
- The aggregate algorithm Agg is a deterministic algorithm. It takes distinct $(id_1, m_1, t_1), \ldots, (id_n, m_n, t_n) \in \mathcal{I} \times \mathcal{M} \times \mathcal{T}$ as input, where $n \geq 1$ is a variable, and returns an aggregate tag $T \in \mathcal{T}_{\mathrm{A}}$. Namely, $T \leftarrow \mathsf{Agg}((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$.
- The verification algorithm Ver is a deterministic algorithm. It takes distinct $(id_1, m_1), \ldots, (id_n, m_n) \in \mathcal{I} \times \mathcal{M}$, $T \in \mathcal{T}_{\mathrm{A}}$, and $(id_1, k_1), \ldots, (id_n, k_n) \in \mathcal{I} \times \mathcal{K}$ as input and returns a decision $d \in \{\top, \bot\}$. Namely, $d \leftarrow \mathsf{Ver}(((id_1, k_1), \ldots, (id_n, k_n)), ((id_1, m_1), \ldots, (id_n, m_n)), T)$, where $n \geq 1$. The pair $((id_1, m_1), \ldots, (id_n, m_n))$ and T are judged valid with respect to $((id_1, k_1), \ldots, (id_n, k_n))$ if $d = \top$. They are judged invalid otherwise.

AM is required to satisfy correctness: For $(id_1, k_1), \ldots, (id_n, k_n)$ and $(id_1, m_1), \ldots, (id_n, m_n)$, if $t_j \leftarrow \mathsf{Tag}(k_j, m_j)$ for $1 \leq j \leq n$ and $T \leftarrow \mathsf{Agg}((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$, then $\mathsf{Ver}(((id_1, k_1), \ldots, (id_n, k_n)), ((id_1, m_1), \ldots, (id_n, m_n)), T) = \top$.

Remark 1: The definition of the aggregate algorithm is different from the definition by Katz and Lindell [5]. They defined it in a recursive manner, which allows gradual

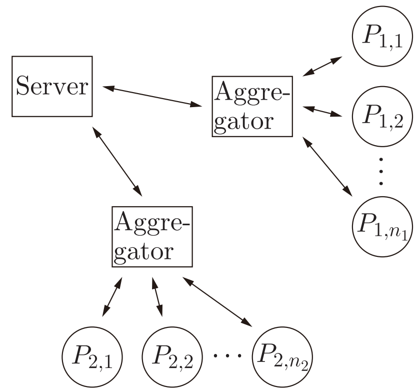

FIGURE 1. A targeted system configuration.

aggregation of tags during communication. Our definition assumes, for example as shown in Fig. 1, a setting where a server communicates with IoT devices and/or sensors in an authenticated way with MAC via aggregators such as edge devices. Each aggregator collects data, aggregates tags, and sends them to the server.

B. SECURITY REQUIREMENT

An aggregate MAC scheme is required to satisfy unforgeability and soundness. Actually, soundness is not formalized in [5], and it is introduced for the application of aggregate MAC to the group-testing aggregate MAC.

1) UNFORGEABILITY

We introduce a game $\mathfrak{G}^{\mathrm{uf}}_{\mathsf{AM},\mathbf{A}}$ played by an adversary $\mathbf{A}$ against $\mathsf{AM} \triangleq (\mathsf{KG}, \mathsf{Tag}, \mathsf{Agg}, \mathsf{Ver})$ concerning unforgeability. In this game, $\mathbf{A}$ is allowed to make multiple queries adaptively to the tagging oracle $\mathcal{TG}$, the key-disclosure oracle $\mathcal{KD}$ and the verification oracle $\mathcal{VR}$:

- For a query $(id, m) \in \mathcal{I} \times \mathcal{M}$, $\mathcal{TG}$ returns $t \leftarrow \mathsf{Tag}(k, m)$, where $k \in \mathcal{K}$ is the key for id.
- For a query $id \in \mathcal{I}$, $\mathcal{KD}$ returns the key $k \in \mathcal{K}$ for id.
- For a query $(((id_1, m_1), \ldots, (id_n, m_n)), T)$, $\mathcal{VR}$ returns $d \leftarrow \mathsf{Ver}(((id_1, k_1), \ldots, (id_n, k_n)), ((id_1, m_1), \ldots, (id_n, m_n)), T)$.

For a query $(((id_1, m_1), \ldots, (id_n, m_n)), T)$ to $\mathcal{VR}$, if $\mathbf{A}$ already asks (id_j, m_j) to $\mathcal{TG}$ or id_j to $\mathcal{KD}$, then we call it a stale pair. If it is not stale, then we call it a fresh pair. $\mathbf{A}$ is not allowed to ask $\mathcal{VR}$ a query only with stale pairs.

$\mathfrak{G}^{\mathrm{uf}}_{\mathsf{AM},\mathbf{A}}$ outputs 1 if $\mathbf{A}$ succeeds in getting $\top$ from $\mathcal{VR}$ and 0 otherwise. The advantage of $\mathbf{A}$ against AM concerning unforgeability is defined as

$$\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{AM}}(\mathbf{A}) \triangleq \Pr[\mathfrak{G}^{\mathrm{uf}}_{\mathsf{AM},\mathbf{A}} = 1]. \qquad (3)$$

AM is informally said to satisfy unforgeability if $\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{AM}}(\mathbf{A})$ is negligibly small for any efficient $\mathbf{A}$.

2) SOUNDNESS

We introduce a game $\mathfrak{G}^{\mathrm{snd}}_{\mathsf{AM},\mathbf{A}}$ played by an adversary $\mathbf{A}$ against AM concerning soundness. In this game, $\mathbf{A}$ is given access to the tagging oracle $\mathcal{TG}$ and the key-disclosure oracle $\mathcal{KD}$. $\mathbf{A}$ is also given access to the aggregate-then-verify oracle $\mathcal{AVR}$ described below. $\mathbf{A}$ is allowed to make multiple queries adaptively to each of them.

$\mathcal{AVR}$ accepts $((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$ as a query, and computes

1) For $1 \leq i \leq n$, $d_i \leftarrow \top$ if $t_i = \mathsf{Tag}(k_i, m_i)$ and $d_i \leftarrow \perp$ otherwise regarding (id_i, k_i),
2) $T \leftarrow \mathsf{Agg}((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$, and
3) $D \leftarrow \mathsf{Ver}(((id_1, k_1), \ldots, (id_n, k_n)), ((id_1, m_1), \ldots, (id_n, m_n)), T)$.

$\mathfrak{G}^{\mathrm{snd}}_{\mathsf{AM},\mathbf{A}}$ outputs 1 if $\mathbf{A}$ succeeds in making a query to $\mathcal{AVR}$ such that $D = \top$ and there exists some i such that $d_i = \perp$. Otherwise, $\mathfrak{G}^{\mathrm{snd}}_{\mathsf{AM},\mathbf{A}}$ outputs 0. The advantage of $\mathbf{A}$ against AM concerning soundness is defined as

$$\mathrm{Adv}^{\mathrm{snd}}_{\mathsf{AM}}(\mathbf{A}) \triangleq \Pr[\mathfrak{G}^{\mathrm{snd}}_{\mathsf{AM},\mathbf{A}} = 1]. \tag{4}$$

AM is informally said to satisfy soundness if $\mathrm{Adv}^{\mathrm{snd}}_{\mathsf{AM}}(\mathbf{A})$ is negligibly small for any efficient $\mathbf{A}$.

C. KATZ-LINDELL AGGREGATE MAC SCHEME

We describe an aggregate MAC scheme proposed by Katz and Lindell [5]. We call it AM_{X}.

1) SCHEME

AM_{X} uses a MAC function $F : \mathcal{K}\times\mathcal{M} \to \{0, 1\}^\tau$ for tagging.

- For given $id \in \mathcal{I}$, the key generation algorithm returns (id, k), where $k \twoheadleftarrow \mathcal{K}$.
- For given $k \in \mathcal{K}$ and $m \in \mathcal{M}$, the tagging algorithm returns $t \leftarrow F_k(m)$.
- For given $(id_1, m_1, t_1), \ldots, (id_n, m_n, t_n)$, the aggregate algorithm returns $T \leftarrow t_1 \oplus t_2 \oplus \cdots \oplus t_n$. Notice that $T \leftarrow t_1$ if $n = 1$.
- For given $(id_1, k_1), \ldots, (id_n, k_n)$ and $((id_1, m_1), \ldots, (id_n, m_n), T)$, the verification algorithm returns $\top$ if $T = F_{k_1}(m_1) \oplus \cdots \oplus F_{k_n}(m_n)$ and $\perp$ otherwise.

2) UNFORGEABILITY

It is shown in [5] that AM_{X} satisfies unforgeability for any efficient adversary making a single query to its verification oracle. AM_{X} satisfies unforgeability even for any efficient adversary making multiple queries to its verification oracle:

Proposition 1: Let $\mathbf{A}$ be any adversary against AM_{X} with ℓ users. Suppose that $\mathbf{A}$ makes q_{t} queries to its tagging oracle and q_{v} queries to its verification oracle. Suppose that each verification query involves at most p pairs of an ID and a message. Then, there exists some adversary $\mathbf{B}$ against F such that

$$\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{AM}_{\mathrm{X}}}(\mathbf{A}) \leq \ell q_{\mathrm{v}} \cdot \mathrm{Adv}^{\mathrm{mac}}_{F}(\mathbf{B}). \tag{5}$$

$\mathbf{B}$ makes at most $q_{\mathrm{t}} + p$ queries to its tagging oracle and at most a single query to its verification oracle. The run time of $\mathbf{B}$ is at most about $\mathrm{T}_{\mathbf{A}} + \mathrm{T}_F(q_{\mathrm{t}} + p)$, where $\mathrm{T}_{\mathbf{A}}$ is the run time of $\mathbf{A}$ and T_F is time required to compute F.

Proof: In $\mathfrak{G}^{\mathrm{mac}}_{F,\mathbf{B}}$, $\mathbf{B}$ is given F_{k^*} and V_{k^*}, where $k^* \twoheadleftarrow \mathcal{K}$. $\mathbf{B}$ simulates $\mathfrak{G}^{\mathrm{uf}}_{\mathsf{AM}_{\mathrm{X}},\mathbf{A}}$.

First, $\mathbf{B}$ selects a user id^* uniformly at random among ℓ users. $\mathbf{B}$ assigns secret keys in $\mathcal{K}$ for all the users other than id^*. $\mathbf{B}$ also selects a positive integer $a^* \leq q_{\mathrm{v}}$ uniformly at random.

For a tagging query (id, m) by $\mathbf{A}$, if $id = id^*$, then $\mathbf{B}$ asks it to F_{k^*} and returns $F_{k^*}(m)$ to $\mathbf{A}$. Otherwise, $\mathbf{B}$ returns the tag computed by using the corresponding secret key.

For a key-disclosure query id by $\mathbf{A}$, if $id \neq id^*$, then $\mathbf{B}$ returns the corresponding secret key. Otherwise, $\mathbf{B}$ aborts.

For the a-th verification query by $\mathbf{A}$ such that $a < a^*$, $\mathbf{B}$ simply returns $\perp$. For the a^*-th verification query $(((id_1, m_1), \ldots, (id_n, m_n)), T)$, if $id_j \neq id^*$ for every $j \in \{1, 2, \ldots, n\}$, then $\mathbf{B}$ verifies it and returns the decision. Otherwise, there exists some j such that $id_j = id^*$. If, for every (id_j, m_j) such that $id_j = id^*$, $\mathbf{A}$ already asks it to its tagging oracle and know the corresponding tag, then $\mathbf{B}$ also verifies the verification query and returns the decision. Otherwise, let j^* be an integer such that $id_{j^*} = id^*$ and $\mathbf{A}$ does not yet ask (id_{j^*}, m_{j^*}) to its tagging oracle. $\mathbf{B}$ gets a tag t_j of (id_j, m_j) for every $j \in \{1, 2, \ldots, n\} \setminus \{j^*\}$. Then, $\mathbf{B}$ asks $(m_{j^*}, T \oplus \bigoplus_{j\neq j^*} t_j)$ to its verification oracle, returns the answer to $\mathbf{A}$ and terminates.

Suppose that $\mathfrak{G}^{\mathrm{uf}}_{\mathsf{AM}_{\mathrm{X}},\mathbf{A}}$ outputs 1. Let Win be the event that the a^*-th verification query Q is the first successful query made by $\mathbf{A}$ and Q involves (id^*, m^*) such that $\mathbf{A}$ asks neither (id^*, m^*) to its tagging oracle nor id^* to its key-disclosure oracle before asking Q. Then,

$$\begin{aligned}
\Pr[\mathfrak{G}^{\mathrm{mac}}_{F,\mathbf{B}} = 1] &= \Pr[(\mathfrak{G}^{\mathrm{uf}}_{\mathsf{AM}_{\mathrm{X}},\mathbf{A}} = 1) \wedge \mathsf{Win}] && (6)\\
&= \Pr[\mathsf{Win} \mid \mathfrak{G}^{\mathrm{uf}}_{\mathsf{AM}_{\mathrm{X}},\mathbf{A}} = 1] \Pr[\mathfrak{G}^{\mathrm{uf}}_{\mathsf{AM}_{\mathrm{X}},\mathbf{A}} = 1] && (7)\\
&\geq (\ell q_{\mathrm{v}})^{-1} \cdot \Pr[\mathfrak{G}^{\mathrm{uf}}_{\mathsf{AM}_{\mathrm{X}},\mathbf{A}} = 1]. && (8)
\end{aligned}$$

Thus, $\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{AM}_{\mathrm{X}}}(\mathbf{A}) \leq \ell q_{\mathrm{v}} \cdot \mathrm{Adv}^{\mathrm{mac}}_{F}(\mathbf{B})$. □

3) SOUNDNESS

AM_{X} does not satisfy soundness. For example, let (id_1, m_1, t_1) and (id_2, m_2, t_2) be tuples such that $t_1 = F_{k_1}(m_1)$ and $t_2 = F_{k_2}(m_2)$. Let $\tilde{t}_1 \triangleq t_1 \oplus c$ and $\tilde{t}_2 \triangleq t_2 \oplus c$, where c is non-zero. Then, the aggregate tag is $\tilde{t}_1 \oplus \tilde{t}_2 = t_1 \oplus t_2$. Thus, with respect to (id_1, k_1) and (id_2, k_2), $((id_1, m_1), (id_2, m_2))$, $\tilde{t}_1 \oplus \tilde{t}_2)$ is judged valid though $(id_1, m_1, \tilde{t}_1)$ and $(id_2, m_2, \tilde{t}_2)$ are invalid.

D. AGGREGATE MAC SCHEME USING HASHING

We describe an aggregate MAC scheme using hashing for aggregate. We call it AM_{H}.

1) SCHEME

AM_{H} aggregates tags with a cryptographic hash function $H : \{0, 1\}^* \to \{0, 1\}^\tau$. Let $F : \mathcal{K} \times \mathcal{M} \to \{0, 1\}^\tau$ be a MAC function. The key generation and tagging algorithms of AM_{H} are identical to those of AM_{X}.

- For given $(id_1, m_1, t_1), \ldots, (id_n, m_n, t_n)$, the aggregate algorithm returns $T \leftarrow H(t_1 \| t_2 \| \cdots \| t_n)$. To make each aggregate tag unique, it is assumed that $(id_1, m_1, t_1), \ldots, (id_n, m_n, t_n)$ is ordered in a lexicographic order. Notice that $T \leftarrow H(t_1)$ if $n = 1$ just for simplifying the notation. It does not cause any problem if we specify $T \leftarrow t_1$ for $n = 1$ and $T \leftarrow H(t_1 \| t_2 \| \cdots \| t_n)$ for $n \geq 2$.
- For given $(id_1, k_1), \ldots, (id_n, k_n)$ and $((id_1, m_1), \ldots, (id_n, m_n), T)$, the verification algorithm returns $\top$ if $T = H(F_{k_1}(m_1) \| F_{k_2}(m_2) \| \cdots \| F_{k_n}(m_n))$ and $\perp$ otherwise.

2) UNFORGEABILITY

$\mathsf{AM_H}$ satisfies unforgeability if F is a secure MAC function and H is a random oracle:

Theorem 1: Let $\mathbf{A}$ be any adversary against $\mathsf{AM_H}$ with ℓ users. For $\mathbf{A}$, let q_h be the number of its queries to the random oracle H, q_t be the number of the queries to its tagging oracle, and q_v be the number of the queries to its verification oracle. Suppose that each verification query involves at most p pairs of an ID and a message. Then, there exists some adversary $\mathbf{B}$ against F such that

$$\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{AM_H}}(\mathbf{A}) \leq \ell q_v \cdot \mathrm{Adv}^{\mathrm{mac}}_F(\mathbf{B}) + q_v/2^\tau. \tag{9}$$

$\mathbf{B}$ makes at most $q_h + q_v$ queries to H, at most $q_t + p$ queries to its tagging oracle and at most a single query to its verification oracle. The run time of $\mathbf{B}$ is at most about $\mathrm{T}_{\mathbf{A}} + \mathrm{T}_F(q_t + p)$, where $\mathrm{T}_{\mathbf{A}}$ is the run time of $\mathbf{A}$ and T_F is time required to compute F.

Proof: Let $\mathsf{F0}$ be the event that there exists at least one successful forgery $((id_1, m_1), \ldots, (id_n, m_n), T)$ by $\mathbf{A}$ such that $\mathbf{A}$ makes a query $(t_1, \ldots, t_n)$ to H satisfying $H(t_1 \| \cdots \| t_n) = T$, where t_i is the valid tag for (id_i, m_i) for $1 \leq i \leq n$. Then,

$$\begin{aligned} \mathrm{Adv}^{\mathrm{uf}}_{\mathsf{AM_H}}(\mathbf{A}) &= \Pr[\mathfrak{G}^{\mathrm{uf}}_{\mathsf{AM_H},\mathbf{A}} = 1] && (10)\\ &= \Pr[\mathsf{F0}] + \Pr[(\mathfrak{G}^{\mathrm{uf}}_{\mathsf{AM_H},\mathbf{A}} = 1) \wedge \overline{\mathsf{F0}}] && (11)\\ &\leq \Pr[\mathsf{F0}] + q_v/2^\tau \ . && (12) \end{aligned}$$

For $\Pr[\mathsf{F0}]$, the proof is similar to that of Proposition 1. There exists an adversary $\mathbf{B}$ against F such that $\Pr[\mathsf{F0}] \leq \ell q_v \cdot \mathrm{Adv}^{\mathrm{mac}}_F(\mathbf{B})$. □

3) SOUNDNESS

$\mathsf{AM_H}$ satisfies soundness if H is collision-resistant:

Theorem 2: Let $\mathbf{A}$ be any adversary against $\mathsf{AM_H}$ concerning soundness. Suppose that $\mathbf{A}$ makes q_t queries to its tagging oracle and q_a queries to its aggregate-then-verification oracle. Suppose that the total number of the tuples of an ID, a message and a tag in the queries to the aggregate-then-verification oracle is at most n_a. Then, there exists some adversary $\mathbf{B}$ against H concerning collision resistance such that

$$\mathrm{Adv}^{\mathrm{snd}}_{\mathsf{AM_H}}(\mathbf{A}) \leq \mathrm{Adv}^{\mathrm{col}}_H(\mathbf{B}). \tag{13}$$

The run time of $\mathbf{B}$ is at most about $\mathrm{T}_{\mathbf{A}} + \mathrm{T}_F(q_t + n_a) + \mathrm{T}_H q_a$, where $\mathrm{T}_{\mathbf{A}}$ is the run time of $\mathbf{A}$, and T_F and T_H are amounts of time required to compute F and H, respectively.

Proof: $\mathbf{B}$ runs $\mathfrak{G}^{\mathrm{snd}}_{\mathsf{AM_H},\mathbf{A}}$. $\mathbf{B}$ generates the keys of all users. $\mathbf{B}$ simulates the tagging oracle, the key-disclosure oracle and the aggregate-then-verification oracle for $\mathbf{A}$. Suppose that $\mathfrak{G}^{\mathrm{snd}}_{\mathsf{AM_H},\mathbf{A}}$ outputs 1. Then, $\mathbf{A}$ makes a query $((id_1, m_1, t'_1), \ldots, (id_n, m_n, t'_n))$ to its aggregate-then-verification oracle such that the decision for the query is $\top$ and there exists some j such that t'_j is not a valid tag for (id_j, m_j). For $1 \leq i \leq n$, let t_i be the valid tag for (id_i, m_i). Then, since F is deterministic, $(t'_1, \ldots, t'_n) \neq (t_1, \ldots, t_n)$ and $H(t'_1 \| \cdots \| t'_n) = H(t_1 \| \cdots \| t_n)$. Thus, if $\mathfrak{G}^{\mathrm{snd}}_{\mathsf{AM_H},\mathbf{A}}$ outputs 1, then $\mathbf{B}$ finds a collision for H. □

E. UNFORGEABILITY VERSUS SOUNDNESS

The security analyses of $\mathsf{AM_X}$ and $\mathsf{AM_H}$ make it clear that unforgeability and soundness are separated.

Soundness is not implied by unforgeability since $\mathsf{AM_X}$ satisfies unforgeability but does not satisfy soundness.

Unforgeability is not implied by soundness, either. Theorem 2 shows that $\mathsf{AM_H}$ satisfies soundness only if the underlying hash function satisfies collision resistance, and it is the property of the aggregate algorithm. Suppose that $\mathsf{AM_H}$ uses a collision-resistant hash function and the following forgeable MAC function: $t \leftarrow F_k(m)$ and t is the most significant τ bits of m. Then, $\mathsf{AM_H}$ does not satisfy unforgeability but still satisfies soundness.

IV. GROUP-TESTING AGGREGATE MAC

A. NON-ADAPTIVE GROUP TESTING

A non-adaptive group-testing is applied to a set of items. It assumes that each item is either positive or negative. It also assumes that multiple items can be examined by a single test and that its result is negative if and only if all the items in the test are negative. One may be able to identify positive items with a smaller number of tests by non-adaptive group-testing than by simply examining one by one.

A non-adaptive group-testing for n items with u tests can be represented by a $u \times n$ $\{0, 1\}$-matrix: The (i, j) element of the matrix equals 1 if and only if the i-th test is applied to the j-th item. A matrix representing a non-adaptive group-testing is called a group-testing matrix. In a group-testing, each item should be tested and each test should be run on at least one item. Thus, without loss of generality, we can assume that a goup-testing matrix should have at least one element equal to 1 in every row and every column.

A useful property of a group-testing matrix is known:

*Definition 1 (**d**-disjunct):* A $\{0, 1\}$-matrix $\boldsymbol{G}$ is called d-disjunct if, for any $(d + 1)$ columns $\boldsymbol{g}^{\mathrm{c}}_{j_1}, \boldsymbol{g}^{\mathrm{c}}_{j_2}, \ldots, \boldsymbol{g}^{\mathrm{c}}_{j_{d+1}}$ of $\boldsymbol{G}$, $\bigvee_{l=1}^{d} \boldsymbol{g}^{\mathrm{c}}_{j_l} \neq \bigvee_{l=1}^{d+1} \boldsymbol{g}^{\mathrm{c}}_{j_l}$, where $\vee$ is the component-wise disjunction.

Notice that, if $\boldsymbol{G}$ is d-disjunct, then it is d'-disjunct for any d' such that $d' \leq d$.

If there are at most d positive items among n items, then, using a d-disjunct group-testing matrix $\boldsymbol{G} = (g_{i,j})$, the following procedure identifies all the positive items:

Let $j \in \{1, 2, \ldots, n\}$ represent the j-th item. For $\boldsymbol{G}$, let $\mathcal{S}_i \triangleq \{j \mid 1 \le j \le n,\ g_{i,j} = 1\}$.

1) $\mathcal{J} \leftarrow \{1, 2, \ldots, n\}$.
2) For $1 \le i \le u$, if the result of the i-th test is negative, then $\mathcal{J} \leftarrow \mathcal{J} \setminus \mathcal{S}_i$.
3) Output $\mathcal{J}$.

The output $\mathcal{J}$ is exactly the set of all positive items.

B. SYNTAX

A group-testing aggregate MAC (GTA MAC) scheme is defined to be a tuple of algorithms $\mathsf{GTAM} \triangleq (\mathsf{KG}, \mathsf{Tag}, \mathsf{GTA}, \mathsf{GTV})$ associated with a set $\mathcal{G}$ of group-testing matrices, a set $\mathcal{I}$ of IDs, a set $\mathcal{K}$ of keys, a set $\mathcal{M}$ of messages, a set $\mathcal{T}$ of tags and a set $\mathcal{T}_{\mathrm{A}}$ of aggregate tags. The sizes (both the numbers of rows and the numbers of columns) of the matrices in $\mathcal{G}$ may be different from each other in general.

- The key generation algorithm KG is a randomized algorithm. It takes $id \in \mathcal{I}$ as input and returns $k_{id} \in \mathcal{K}$ for id. Namely, $(id, k_{id}) \leftarrow \mathsf{KG}(id)$.
- The tagging algorithm Tag is a deterministic algorithm. It takes $k \in \mathcal{K}$ and $m \in \mathcal{M}$ as input and returns $t \in \mathcal{T}$. Namely, $t \leftarrow \mathsf{Tag}(k, m)$.
- The group-testing aggregate algorithm GTA is a deterministic algorithm. It takes a group-testing matrix $\boldsymbol{G} \in \mathcal{G}$ and $((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n)) \in (\mathcal{I} \times \mathcal{M} \times \mathcal{T})^n$ as input and returns aggregate tags $(T_1, \ldots, T_u) \in \mathcal{T}_{\mathrm{A}}^u$, where $\boldsymbol{G}$ is assumed to be a $u \times n$ matrix. Namely, $(T_1, \ldots, T_u) \leftarrow \mathsf{GTA}(\boldsymbol{G}; (id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$. It is required that $(id_1, m_1, t_1), \ldots, (id_n, m_n, t_n)$ are distinct.
- The group-testing verification algorithm GTV is a deterministic algorithm. It takes $\boldsymbol{G} \in \mathcal{G}$, $((id_1, m_1), \ldots, (id_n, m_n)) \in (\mathcal{I} \times \mathcal{M})^n$, $(T_1, \ldots, T_u) \in \mathcal{T}_{\mathrm{A}}^u$, and $(id_1, k_1), \ldots, (id_n, k_n) \in \mathcal{I} \times \mathcal{K}$ as input and returns a set of (id_j, m_j)'s, where $\boldsymbol{G}$ is assumed to be a $u \times n$ matrix. Namely, $\mathcal{J} \leftarrow \mathsf{GTV}(((id_1, k_1), \ldots, (id_n, k_n)), (\boldsymbol{G}; (id_1, m_1), \ldots, (id_n, m_n), (T_1, \ldots, T_u)))$, and $\mathcal{J}$ is a set of (id_j, m_j)'s judged invalid. It is required that $(id_1, m_1), \ldots, (id_n, m_n)$ are distinct.

GTAM is required to satisfy correctness: For a $u \times n$ group-testing matrix $\boldsymbol{G}$, $((id_1, k_1), \ldots, (id_n, k_n))$, $((id_1, m_1), \ldots, (id_n, m_n))$ and $(T_1, \ldots, T_u)$, if $t_j \leftarrow \mathsf{Tag}(k_j, m_j)$ for $1 \le j \le n$, and $(T_1, \ldots, T_u) \leftarrow \mathsf{GTA}(\boldsymbol{G}; (id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$, then $\mathsf{GTV}(((id_1, k_1), \ldots, (id_n, k_n)), (\boldsymbol{G}; (id_1, m_1), \ldots, (id_n, m_n), (T_1, \ldots, T_u))) = \emptyset$.

Remark 2: With the use of a $u \times n$ matrix for the group-testing aggregate algorithm GTA, the total tag size is reduced by a factor of u/n if a tag t_j and an aggregate tag T_i have an equal length. It is shown that the minimum number of rows for a d-disjunct matrix with n columns is $O(d^2 \log n)$ [6].

C. SECURITY REQUIREMENT

For a GTA MAC scheme GTAM, we formalize two security requirements, unforgeability and identifiability.

1) UNFORGEABILITY

We introduce a game $\mathfrak{G}^{\mathrm{uf}}_{\mathsf{GTAM},\mathbf{A}}$ played by an adversary $\mathbf{A}$ against GTAM concerning unforgeability. In this game, $\mathbf{A}$ is able to make queries adaptively to the tagging oracle $\mathcal{TG}$, the key disclosure oracle $\mathcal{KD}$ and the group-testing verification oracle $\mathcal{GTV}$:

- For a query $(id, m) \in \mathcal{I} \times \mathcal{M}$, $\mathcal{TG}$ returns $t \leftarrow \mathsf{Tag}(k, m)$, where $k \in \mathcal{K}$ is the key for id.
- For a query $id \in \mathcal{I}$, $\mathcal{KD}$ returns $k \in \mathcal{K}$ for id.
- For a query $(\boldsymbol{G}; (id_1, m_1), \ldots, (id_n, m_n), (T_1, \ldots, T_u))$, $\mathcal{GTV}$ returns $\mathcal{J} \leftarrow \mathsf{GTV}(((id_1, k_1), \ldots, (id_n, k_n)), (\boldsymbol{G}; (id_1, m_1), \ldots, (id_n, m_n), (T_1, \ldots, T_u)))$.

For a query $(\boldsymbol{G}; (id_1, m_1), \ldots, (id_n, m_n), (T_1, \ldots, T_u))$ to $\mathcal{GTV}$, if $\mathbf{A}$ already asks (id_j, m_j) to $\mathcal{TG}$ or id_j to $\mathcal{KD}$, then we call it a stale pair. If it is not stale, then we call it a fresh pair. $\mathbf{A}$ is not allowed to ask $\mathcal{GTV}$ a query only with stale pairs.

$\mathfrak{G}^{\mathrm{uf}}_{\mathsf{GTAM},\mathbf{A}}$ outputs 1 if $\mathbf{A}$ succeeds in asking a query $Q = (\boldsymbol{G}; (id_1, m_1), \ldots, (id_n, m_n), (T_1, \ldots, T_u))$ to $\mathcal{GTV}$ such that $\mathcal{F}(Q) \setminus \mathcal{J} \ne \emptyset$, where $\mathcal{J} \leftarrow \mathsf{GTV}(((id_1, k_1), \ldots, (id_n, k_n)), Q)$ and $\mathcal{F}(Q)$ is the set of fresh pairs of Q. $\mathfrak{G}^{\mathrm{uf}}_{\mathsf{GTAM},\mathbf{A}}$ outputs 0 otherwise.

The advantage of $\mathbf{A}$ against GTAM concerning unforgeability is defined as

$$\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{GTAM}}(\mathbf{A}) \triangleq \Pr[\mathfrak{G}^{\mathrm{uf}}_{\mathsf{GTAM},\mathbf{A}} = 1]. \tag{14}$$

GTAM is informally said to satisfy unforgeability if $\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{GTAM}}(\mathbf{A})$ is negligibly small for any efficient $\mathbf{A}$.

2) IDENTIFIABILITY

We introduce completeness and (weak) soundness for identifiability. Completeness captures the notion that any valid tuple (id, m, t) should be judged valid. (Weak) soundness captures the notion that any invalid tuple should be judged invalid.

We introduce games $\mathfrak{G}^{\text{id-c}}_{\mathsf{GTAM},\mathbf{A}}$, $\mathfrak{G}^{\text{id-s}}_{\mathsf{GTAM},\mathbf{A}}$ and $\mathfrak{G}^{\text{id-ws}}_{\mathsf{GTAM},\mathbf{A}}$ played by an adversary $\mathbf{A}$ against GTAM concerning completeness, soundness and weak soundness, respectively. In all of the games, $\mathbf{A}$ is allowed to make multiple queries to $\mathcal{TG}$ and $\mathcal{KD}$ described in the definition of $\mathfrak{G}^{\mathrm{uf}}_{\mathsf{GTAM},\mathbf{A}}$. $\mathbf{A}$ is also allowed to make queries adaptively to a group-testing oracle $\mathcal{GT}_x$ in $\mathfrak{G}^{\text{id-}x}_{\mathsf{GTAM},\mathbf{A}}$ for $x \in \{\mathrm{c}, \mathrm{s}, \mathrm{ws}\}$. For a query $Q = (\boldsymbol{G}; (id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$, $\mathcal{GT}_x$ computes

1) For $1 \le j \le n$, $d_j \leftarrow \top$ if $t_j = \mathsf{Tag}(k_j, m_j)$ and $d_j \leftarrow \bot$ otherwise regarding (id_j, k_j),
2) $(T_1, \ldots, T_u) \leftarrow \mathsf{GTA}(Q)$ and
3) $\mathcal{J} \leftarrow \mathsf{GTV}(((id_1, k_1), \ldots, (id_n, k_n)), (\boldsymbol{G}; (id_1, m_1), \ldots, (id_n, m_n), (T_1, \ldots, T_u)))$,

where $\boldsymbol{G}$ is assumed to be a $u \times n$ matrix. $\mathcal{GT}_{\mathrm{c}}$ returns 1 if

$$\mathcal{J} \cap \{(id_j, m_j) \mid d_j = \top\} \ne \emptyset \tag{15}$$

and 0 otherwise. $\mathcal{GT}_{\mathrm{s}}$ returns 1 if

$$\{(id_j, m_j) \mid d_j = \perp\} \setminus \mathcal{J} \neq \emptyset \tag{16}$$

and 0 otherwise. $\mathcal{GT}_{\mathrm{ws}}$ returns 1 if the set on the left side of (16) contains one or more fresh pairs. Otherwise, it returns 0.

$\mathfrak{G}^{\text{id-}x}_{\mathsf{GTAM},\mathbf{A}}$ outputs 1 if $\mathcal{GT}_x$ returns 1 for some query and 0 otherwise. The advantage of $\mathbf{A}$ against GTAM concerning identifiability is defined as

$$\mathrm{Adv}^{\text{id-}x}_{\mathsf{GTAM}}(\mathbf{A}) \triangleq \Pr[\mathfrak{G}^{\text{id-}x}_{\mathsf{GTAM},\mathbf{A}} = 1] \tag{17}$$

for $x \in \{\mathrm{c}, \mathrm{s}, \mathrm{ws}\}$.

3) IMPLICATION AND SEPARATION

It is clear that weak soundness is implied by soundness. Weak soundness is also implied by unforgeability:

Proposition 2: Let $\mathbf{A}$ be any adversary against GTAM concerning weak soundness. For $\mathbf{A}$, let q_{t}, q_{k} and q_{g} be the numbers of the queries to its tagging oracle, key-disclosure oracle and group-testing oracle, respectively. Then, there exists some adversary $\mathbf{B}$ against GTAM concerning unforgeability such that

$$\mathrm{Adv}^{\text{id-ws}}_{\mathsf{GTAM}}(\mathbf{A}) \leq \mathrm{Adv}^{\mathrm{uf}}_{\mathsf{GTAM}}(\mathbf{B}). \tag{18}$$

The numbers of queries made by $\mathbf{B}$ to its tagging, key-disclosure and group-testing verification oracles are at most q_{t}, q_{k} and q_{g}, respectively. The run time of $\mathbf{B}$ is at most about the total of the run time of $\mathbf{A}$ and $q_{\mathrm{g}}\mathrm{T}_{\mathsf{GTA}}$, where $\mathrm{T}_{\mathsf{GTA}}$ is time required to run GTA.

Proof: $\mathbf{B}$ simulates $\mathfrak{G}^{\text{id-ws}}_{\mathsf{GTAM},\mathbf{A}}$. In $\mathfrak{G}^{\mathrm{uf}}_{\mathsf{GTAM},\mathbf{B}}$, simulations of the tagging and key disclosure oracles for $\mathbf{A}$ are trivial. For a query $Q \triangleq (\boldsymbol{G}; (id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$ made by $\mathbf{A}$ to its group-testing oracle, $\mathbf{B}$ computes $(T_1, \ldots, T_u) \leftarrow \mathsf{GTA}(Q)$, gets $\mathcal{J} \leftarrow \mathsf{GTV}(\boldsymbol{G}; ((id_1, m_1), \ldots, (id_n, m_n)), (T_1, \ldots, T_u))$ from its group-testing verification oracle. If there exists some fresh $(id_j, m_j) \notin \mathcal{J}$, then $\mathbf{B}$ terminates. Otherwise, $\mathbf{B}$ returns 0 to $\mathbf{A}$ and continues the simulation. If $\mathfrak{G}^{\text{id-ws}}_{\mathsf{GTAM},\mathbf{A}} = 1$, then $\mathfrak{G}^{\mathrm{uf}}_{\mathsf{GTAM},\mathbf{B}} = 1$. □

Similar to the case of aggregate MAC, unforgeability and soundness are separated: Soundness is not implied by unforgeability and unforgeability is not implied by soundness.

V. GENERIC CONSTRUCTION OF GTA MAC SCHEME

Let $\boldsymbol{v} = (v_1, v_2, \ldots, v_n) \in \{0,1\}^n$ and $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathcal{X}^n$ for some set $\mathcal{X}$. Let $\boldsymbol{v} \boxdot \boldsymbol{x} = (x_{j_1}, x_{j_2}, \ldots, x_{j_w})$, where $1 \leq j_1 < j_2 < \cdots < j_w \leq n$, and $v_j = 1$ if and only if $j \in \{j_1, j_2, \ldots, j_w\}$.

For a group-testing matrix $\boldsymbol{G} \in \mathcal{G}$, let $\boldsymbol{G} = (g_{i,j})$ and $\boldsymbol{g}_i$ be the i-th row of $\boldsymbol{G}$.

A. GENERIC CONSTRUCTION

We can construct a GTA MAC scheme $\mathsf{GTAM}_{\mathrm{g}} \triangleq (\mathsf{KG}_{\mathrm{g}}, \mathsf{Tag}_{\mathrm{g}}, \mathsf{GTA}_{\mathrm{g}}, \mathsf{GTV}_{\mathrm{g}})$ associated with a set $\mathcal{G}_{\mathrm{g}}$ of group-testing matrices using an aggregate MAC scheme $\mathsf{AM} \triangleq (\mathsf{KG}, \mathsf{Tag}, \mathsf{Agg}, \mathsf{Ver})$ in the following way: Let $\boldsymbol{G} \in \mathcal{G}_{\mathrm{g}}$ be a $u \times n$ matrix.

- KG_{g} and $\mathsf{Tag}_{\mathrm{g}}$ are identical to KG and Tag, respectively.
- For input $\boldsymbol{G}$ and $(id_1, m_1, t_1), \ldots, (id_n, m_n, t_n)$, $\mathsf{GTA}_{\mathrm{g}}$ computes $T_i \leftarrow \mathsf{Agg}(\boldsymbol{g}_i \boxdot ((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n)))$ for $1 \leq i \leq u$ and outputs $(T_1, \ldots, T_u)$.
- For input $(\boldsymbol{G}; (id_1, m_1), \ldots, (id_n, m_n), (T_1, \ldots, T_u))$ and $(id_1, k_1), \ldots, (id_n, k_n)$, $\mathsf{GTV}_{\mathrm{g}}$ computes
  1) $\mathcal{J} \leftarrow \{(id_1, m_1), \ldots, (id_n, m_n)\}$,
  2) For $1 \leq i \leq u$, if $\mathsf{Ver}(\boldsymbol{g}_i \boxdot ((id_1, k_1), \ldots, (id_n, k_n)), \boldsymbol{g}_i \boxdot ((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n)), T_i) = \top$, then $\mathcal{J} \leftarrow \mathcal{J} \setminus \{(id_j, m_j) \mid 1 \leq j \leq n \wedge g_{i,j} = 1\}$

  and outputs $\mathcal{J}$.

B. UNFORGEABILITY

Unforgeability of $\mathsf{GTAM}_{\mathrm{g}}$ is implied by that of underlying AM:

Theorem 3: For any adversary $\mathbf{A}$ against $\mathsf{GTAM}_{\mathrm{g}}$ with ℓ users, there exists some adversary $\mathbf{B}$ against AM with ℓ users such that

$$\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{GTAM}_{\mathrm{g}}}(\mathbf{A}) \leq \mathrm{Adv}^{\mathrm{uf}}_{\mathsf{AM}}(\mathbf{B}). \tag{19}$$

For $\mathbf{A}$, let q_{t} be the number of the queries to its tagging oracle, q_{k} be the number of the queries to its key-disclosure oracle, and u_{v} and n_{v} be the total number of the tests and the total numebr of the ID-message pairs, respectively, in the queries to its group-testing verification oracle. Then, the numbers of queries made by $\mathbf{B}$ to its tagging, key-disclosure and verification oracles are at most $q_{\mathrm{t}} + n_{\mathrm{v}}$, q_{k} and u_{v}, respectively. The run time of $\mathbf{B}$ is at most about the total of the run time of $\mathbf{A}$ and $n_{\mathrm{v}}\mathrm{T}_{\mathsf{Tag}} + u_{\mathrm{v}}\mathrm{T}_{\mathsf{Agg}}$, where $\mathrm{T}_{\mathsf{Tag}}$ and $\mathrm{T}_{\mathsf{Agg}}$ are amounts of time required to run Tag and Agg, respectively.

Proof: $\mathbf{B}$ simulates $\mathfrak{G}^{\mathrm{uf}}_{\mathsf{GTAM}_{\mathrm{g}},\mathbf{A}}$. In $\mathfrak{G}^{\mathrm{uf}}_{\mathsf{AM},\mathbf{B}}$, simulations of the tagging, key disclosure and group-testing verification oracles for $\mathbf{A}$ are trivial. Notice that $\mathbf{B}$ is not allowed to make verification queries only with stale pairs. If $\mathfrak{G}^{\mathrm{uf}}_{\mathsf{GTAM}_{\mathrm{g}},\mathbf{A}}$ outputs 1, then $\mathbf{A}$ asks a group-testing verification query with a fresh pair judged valid. Thus, $\mathfrak{G}^{\mathrm{uf}}_{\mathsf{AM},\mathbf{B}}$ also outputs 1. □

C. IDENTIFIABILITY

We call an adversary $\mathbf{A}$ against $\mathsf{GTAM}_{\mathrm{g}}$ concerning identifiability d-dishonest if $\mathbf{A}$ only asks group-testing queries $(\boldsymbol{G}; (id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$ such that $|\{(id_j, m_j) \mid t_j \neq \mathsf{Tag}_{\mathrm{g}}(k_j, m_j) \text{ for } (id_j, k_j)\}| \leq d$.

1) COMPLETENESS

Theorem 4: If $\mathsf{GTAM}_{\mathrm{g}}$ is associated with a set of d-disjunct group-testing matrices, then, for any d-dishonest adversary $\mathbf{A}$,

$$\mathrm{Adv}^{\text{id-c}}_{\mathsf{GTAM}_{\mathrm{g}}}(\mathbf{A}) = 0. \tag{20}$$

Proof: Let $(\boldsymbol{G}; (id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$ be a group-testing query made by $\mathbf{A}$. If $\boldsymbol{G}$ is d-disjunct and $\mathbf{A}$ is d-dishonest, then for any valid pair (id_j, m_j), that is,

$t_j = \mathsf{Tag}_{\mathrm{g}}(k_j, m_j)$ for (id_j, k_j), there exists some test in $\boldsymbol{G}$ including (id_j, m_j) and no invalid pairs. $\square$

2) SOUNDNESS

$\mathsf{GTAM}_{\mathrm{g}}$ inherits soundness from AM:

Theorem 5: Let $\mathbf{A}$ be any adversary against $\mathsf{GTAM}_{\mathrm{g}}$. For $\mathbf{A}$, let q_{t} and q_{k} be the numbers of queries to its tagging and key-disclosure oracles, respectively, and u_{v} be the total number of tests in the queries to its group-testing oracle. Then, there exists some adversary $\mathbf{B}$ against AM such that

$$\mathrm{Adv}^{\text{id-s}}_{\mathsf{GTAM}_{\mathrm{g}}}(\mathbf{A}) \leq \mathrm{Adv}^{\text{snd}}_{\mathsf{AM}}(\mathbf{B}). \tag{21}$$

The numbers of queries made by $\mathbf{B}$ to its tagging, key-disclosure and aggregate-then-verify oracles are at most q_{t}, q_{k} and u_{v}, respectively. The run time of $\mathbf{B}$ is at most about that of $\mathbf{A}$.

Proof: $\mathbf{B}$ simulates $\mathfrak{G}^{\text{id-s}}_{\mathsf{GTAM}_{\mathrm{g}},\mathbf{A}}$. In $\mathfrak{G}^{\text{snd}}_{\mathsf{AM},\mathbf{B}}$, simulations of the tagging and key-disclosure oracles are trivial. For a group-testing query $(\boldsymbol{G}; (id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$ made by $\mathbf{A}$, $\mathbf{B}$ makes a query $\boldsymbol{g}_i \boxdot ((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$ to its aggregate-then-verify oracle for each $\boldsymbol{g}_i$. If $\mathfrak{G}^{\text{id-s}}_{\mathsf{GTAM}_{\mathrm{g}},\mathbf{A}}$ outputs 1, then $\mathfrak{G}^{\text{snd}}_{\mathsf{AM},\mathbf{B}}$ also outputs 1. $\square$

VI. INSTANTIATIONS OF GENERIC CONSTRUCTION

We can instantiate $\mathsf{GTAM}_{\mathrm{g}}$ with AM_{X} or AM_{H} as an underlying aggregate MAC scheme.

In the description below, $F : \mathcal{K} \times \mathcal{M} \rightarrow \{0,1\}^{\tau}$ is a MAC function, and $\boldsymbol{G} = (g_{i,j})$ is a $u \times n$ group-testing matrix.

A. GTA MAC SCHEME BASED ON KATZ-LINDELL AGGREGATE MAC

1) SCHEME

The GTA MAC scheme $\mathsf{GTAM}_{\mathrm{X}}$ using AM_{X} is specified below:

- For given $id \in \mathcal{I}$, the key generation algorithm returns (id, k), where $k \twoheadleftarrow \mathcal{K}$.
- For given $k \in \mathcal{K}$ and $m \in \mathcal{M}$, the tagging algorithm returns $t \leftarrow F_k(m)$.
- For given $(\boldsymbol{G}; (id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$, the group-testing aggregate algorithm computes $T_i \leftarrow \bigoplus_{g_{i,j}=1} t_j$ for $1 \leq i \leq u$, and returns $(T_1, \ldots, T_u)$.
- For given $(\boldsymbol{G}; ((id_1, m_1), \ldots, (id_n, m_n)), (T_1, \ldots, T_u))$ and $(id_1, k_1), \ldots, (id_n, k_n)$, the group-testing verification algorithm executes
  1) $\mathcal{J} \leftarrow \{(id_1, m_1), \ldots, (id_n, m_n)\}$.
  2) For $1 \leq i \leq u$, if $T_i = \bigoplus_{g_{i,j}=1} F_{k_j}(m_j)$, then $\mathcal{J} \leftarrow \mathcal{J} \setminus \{(id_j, m_j) \mid 1 \leq j \leq n \wedge g_{i,j} = 1\}$.

  and returns $\mathcal{J}$.

2) UNFORGEABILITY

Unforgeability of $\mathsf{GTAM}_{\mathrm{X}}$ is implied by unforgeability of the underlying MAC function F:

Corollary 1: Let $\mathbf{A}$ be any adversary against $\mathsf{GTAM}_{\mathrm{X}}$ with ℓ users. For $\mathbf{A}$, let q_{t} be the number of the queries to its tagging oracle, q_{k} be the number of the queries to its key-disclosure oracle, and u_{v} and n_{v} be the total number of the tests and the total numebr of the ID-message pairs, respectively, in the queries to its group-testing verification oracle. Then, there exists some adversary $\mathbf{B}$ against F such that

$$\mathrm{Adv}^{\text{uf}}_{\mathsf{GTAM}_{\mathrm{X}}}(\mathbf{A}) \leq \ell u_{\mathrm{v}} \cdot \mathrm{Adv}^{\text{mac}}_{F}(\mathbf{B}). \tag{22}$$

The numbers of queries made by $\mathbf{B}$ to its tagging and verification oracles are at most $q_{\mathrm{t}} + n_{\mathrm{v}}$ and 1, respectively. The run time of $\mathbf{B}$ is at most about the total of the run time of $\mathbf{A}$ and $\mathrm{T}_F(q_{\mathrm{t}} + n_{\mathrm{v}})$, where T_F is time required to compute F.

Corollary 1 follows from Proposition 1 and Theorem 3.

3) IDENTIFIABILITY

a: COMPLETENESS

From Theorem 4, if $\mathsf{GTAM}_{\mathrm{X}}$ is associated with a set of d-disjunct group-testing matrices, then it satisfies completeness against any d-dishonest adversary.

b: SOUNDNESS

Weak soundness of $\mathsf{GTAM}_{\mathrm{X}}$ is confirmed by Corollary 1.

It is easy to see that $\mathsf{GTAM}_{\mathrm{X}}$ does not satisfy soundness. However, if it is associated with a set of d-disjunct group-testing matrices and an adversary is d-dishonest, then one can easily verify whether the result of a group-testing is correct or not.

Suppose that $\boldsymbol{G}$ is d-disjunct and that $\mathbf{A}$ is d-dishonest. Let $(\boldsymbol{G}; (id_1, m_1, \tilde{t}_1), \ldots, (id_n, m_n, \tilde{t}_n))$ be a query made by $\mathbf{A}$ to $\mathcal{GT}_{\mathrm{s}}$ and $\mathcal{J}$ be the set of pairs of an ID and a message judged invalid by group-testing verification. Then, let $\mathcal{P}$ be the set of the tests in $\boldsymbol{G}$ which result in positive. Let $\mathcal{Q}$ be the set of tests in $\boldsymbol{G}$ which involve one or more pairs in $\mathcal{J}$. It is easy to see that $\mathcal{Q} \subseteq \mathcal{P}$.

Let $\mathcal{J}' \triangleq \{(id_j, m_j) \mid 1 \leq j \leq n \wedge \tilde{t}_j \neq \mathsf{Tag}_{\mathrm{g}}(k_j, m_j)$ for $(id_j, k_j)\}$. Since $|\mathcal{J}'| \leq d$ and $\boldsymbol{G}$ is d-disjunct, $\mathcal{J} \subseteq \mathcal{J}'$.

Suppose that $\mathcal{J} \subsetneq \mathcal{J}'$ and $(id_{j^*}, m_{j^*}) \in \mathcal{J}' \setminus \mathcal{J}$. Then, since $\boldsymbol{G}$ is d-disjunct, there exists some test in $\boldsymbol{G}$ such that it involves (id_{j^*}, m_{j^*}) and none of the other ID-message pairs it involves are in $\mathcal{J}'$. The result of the test is positive for AM_{X} and $\mathcal{Q} \subsetneq \mathcal{P}$. On the other hand, if $\mathcal{Q} \subsetneq \mathcal{P}$, then each of the tests in $\mathcal{P} \setminus \mathcal{Q}$ involves some pair in $\mathcal{J}' \setminus \mathcal{J}$.

It is concluded that $\mathcal{P} \subsetneq \mathcal{Q}$ if and only if $\mathcal{J} \subsetneq \mathcal{J}'$.

B. GTA MAC SCHEME USING HASHING

1) SCHEME

The GTA MAC scheme $\mathsf{GTAM}_{\mathrm{H}}$ using AM_{H} is specified below: Let $H : \{0,1\}^* \rightarrow \{0,1\}^{\tau}$ be a cryptographic hash function.

- For given $id \in \mathcal{I}$, the key generation algorithm returns (id, k), where $k \twoheadleftarrow \mathcal{K}$.
- For given $k \in \mathcal{K}$ and $m \in \mathcal{M}$, the tagging algorithm returns $t \leftarrow F_k(m)$.
- For given $(\boldsymbol{G}; (id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$, let $\langle\!\langle \boldsymbol{g}_i, (t_1, t_2, \cdots, t_n) \rangle\!\rangle = t_{j_1} \| t_{j_2} \| \cdots \| t_{j_{w_i}}$, where $1 \leq j_1 < j_2 < \cdots < j_{w_i} \leq n$, and $g_{i,j} = 1$ if and only if $j \in \{i_1, i_2, \ldots, i_{w_i}\}$. Then, the group-testing aggregate

algorithm computes $T_i \leftarrow H(\langle\langle \boldsymbol{g}_i, (t_1, t_2, \cdots, t_n)\rangle\rangle)$ for $1 \le i \le u$ and returns $(T_1, \ldots, T_u)$.

- For given $(\boldsymbol{G}; ((id_1, m_1), \ldots, (id_n, m_n)), (T_1, \ldots, T_u))$ and $(id_1, k_1), \ldots, (id_n, k_n)$, the group-testing verification algorithm executes
  1) $\mathcal{J} \leftarrow \{(id_1, m_1), \ldots, (id_n, m_n)\}$.
  2) For $1 \le i \le u$, if $T_i = H(\langle\langle \boldsymbol{g}_i, (F_{k_1}(m_1), \ldots, F_{k_n}(m_n))\rangle\rangle)$, then $\mathcal{J} \leftarrow \mathcal{J} \setminus \{(id_j, m_j) \mid 1 \le j \le n \wedge g_{i,j} = 1\}$.

  and returns $\mathcal{J}$.

2) UNFORGEABILITY

Unforgeability of GTAM_H is implied by that of F in the random oracle model:

Corollary 2: Let $\mathbf{A}$ be any adversary against GTAM_H with ℓ users. For $\mathbf{A}$, let q_h be the number of its queries to H, q_t be the number of the queries to its tagging oracle, q_k be the number of the queries to its key-disclosure oracle and u_v and n_v be the total number of the tests and the total numebr of the ID-message pairs, respectively, in the queries to its group-testing verification oracle. Then, there exists some adversary $\mathbf{B}$ against F such that

$$\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{GTAM}_\mathrm{H}}(\mathbf{A}) \le \ell u_\mathrm{v} \cdot \mathrm{Adv}^{\mathrm{mac}}_{F}(\mathbf{B}) + u_\mathrm{v}/2^{\tau}. \tag{23}$$

$\mathbf{B}$ makes at most $q_\mathrm{h}+u_\mathrm{v}$ queries to H, at most $q_\mathrm{t}+n_\mathrm{v}$ queries to its tagging oracle and at most a single query to its verification oracle. The run time of $\mathbf{B}$ is at most about the total of that of $\mathbf{A}$ and $\mathrm{T}_F(q_\mathrm{t} + n_\mathrm{v})$, where T_F is time to compute F.

Corollary 2 follows from Theorems 1 and 3.

3) IDENTIFIABILITY

a: COMPLETENESS

From Theorem 4, if GTAM_H is associated with a set of d-disjunct group-testing matrices, then it satisfies completeness against any d-dishonest adversary.

b: SOUNDNESS

From Theorems 2 and 5, soundness of GTAM_H is implied by collision resistance of H:

Corollary 3: Let $\mathbf{A}$ be any adversary against GTAM_H. For $\mathbf{A}$, let q_t be the number of the queries to its tagging oracle, q_k be the number of the queries to its key-disclosure oracle, and u_v and n_v be the total number of the tests and the total numebr of the ID-message pairs, respectively, in the queries to its group-testing oracle. Then, there exists some adversary $\mathbf{B}$ against H such that

$$\mathrm{Adv}^{\mathrm{id\text{-}s}}_{\mathsf{GTAM}_\mathrm{H}}(\mathbf{A}) \le \mathrm{Adv}^{\mathrm{col}}_{H}(\mathbf{B}). \tag{24}$$

The run time of $\mathbf{B}$ is at most about the total of that of $\mathbf{A}$ and $\mathrm{T}_F(q_\mathrm{t} + n_\mathrm{v}) + \mathrm{T}_H u_\mathrm{v}$, where T_F and T_H are amounts of time to compute F and H, respectively.

VII. IMPLEMENTATION

A. GROUP-TESTING MATRIX

We generated d-disjunct $u \times n$ group-testing matrices for a few values of n using the shifted transversal design (STD) [22].

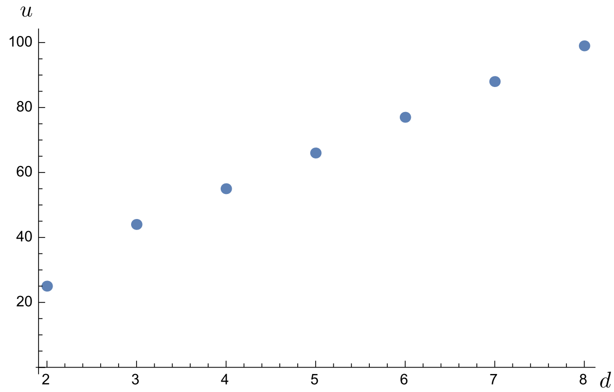

FIGURE 2. Relationship between d and u of d-disjunct $u \times 100$ group-testing matrices generated by STD [22], where $d < \sqrt{u}$.

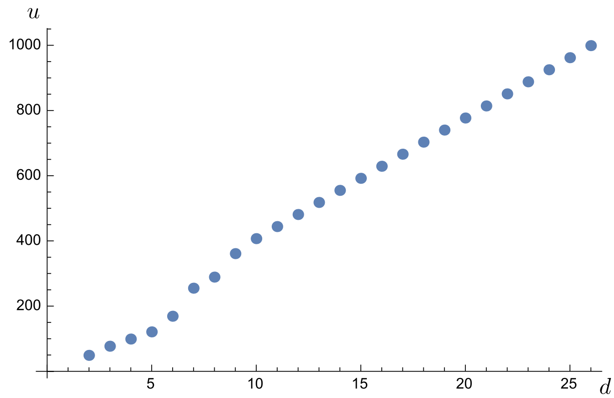

FIGURE 3. Relationship between d and u of d-disjunct $u \times 1000$ group-testing matrices generated by STD [22], where $d < \sqrt{u}$.

We generated a matrix for every d resulting in a matrix satisfying $u < n$.

The relationships between d and u such that $d < \sqrt{u}$ are shown in Figures 2, 3 and 4 for $n = 100$, $n = 1000$ and $n = 10000$, respectively. For example, the rate u/n is smaller than 0.7 if $d \le 5$ for $n = 100$, $d \le 17$ for $n = 1000$ and $d \le 68$ for $n = 10000$. We can see that our GTA MAC is effective in those cases though it still remains open how to design optimal d-disjunct matrices.

B. GTA MAC

We implemented the GTA MAC scheme GTAM_X in Python 3. We adopted HMAC-SHA256 as its underlying MAC function for tagging. We simply used modules `hmac` and `hashlib` to implement HMAC-SHA256.

We measured the runtime of our implementation on MacBook Pro with 2.3GHz Intel Core i5, 16GB memory and macOS Mojave 10.14.6. The results are shown in Table 1, where each entry (time in milliseconds) is the minimum among 10 measurements. "Users" indicates the number of users in group-testing, which equals the number of columns (n) of the corresponding group-testing matrix. "Tagging" indicates total time for generating tags of all users involved in the group-testing. "Verif" indicates total time for verifying tags of all users one by one. "GT Verif" indicates total time

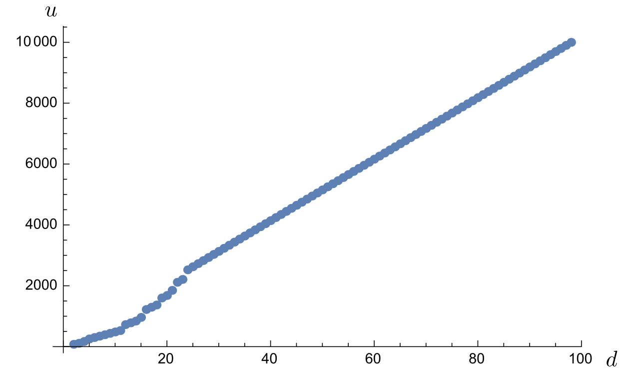

FIGURE 4. Relationship between d and u of d-disjunct $u \times 10000$ group-testing matrices generated by STD [22], where $d < \sqrt{u}$.

TABLE 1. Runtime in milliseconds.

Users	Tagging	Verif	GT Verif
100	5.04×10^{-1}	5.21×10^{-1}	8.20×10^{-1}
1000	4.83	4.97	1.41×10^{1}
10000	5.01×10^{1}	5.17×10^{1}	3.92×10^{2}

for group-testing verification. We did not measure the run time of group-testing aggregate since it is almost equal to the difference between "GT Verif" and "Verif."

The group-testing matrices used for group-testing verification are 66×100, 666×1000 and 6969×10000 d-disjunct matrices, where $d = 5, 17, 68$, respectively. Thus, for each case, the total size of aggregate tags of our GTA MAC scheme is more than 30% smaller than that of the traditional MAC scheme, which attaches a tag to each message. Nevertheless, our group-testing verification is able to identify maliscious users as long as the number of them is at most d.

On the other hand, the runtime of "GT Verif" is larger than that of "Verif" mainly due to the time for generating aggregate tags. It is proportional to the total number of 1's in the group-testing matrix. The total numbers of 1's in the 66×100, 666×1000 and 6969×10000 group-testing matrices are 600, 18000 and 690000, respectively.

VIII. CONCLUSION

We have introduced and formalized GTA MAC. We have presented simple generic construction applying non-adaptive group-testing to aggregate MAC. The generic construction reduces the security of GTA MAC to its underlying cryptographic primitives and d-disjunct property of group-testing matrices. We have discussed two kinds of instantiations of the generic construction. Finally, we have presented some implementation results to show the effectiveness of the proposed GTA MAC. Future work includes design of an algorithm to produce d-disjunct matrices allowing more efficient group-testing. It is also interesting to design an efficient algorithm to verify whether a given group-testing matrix is d-disjunct or not.

ACKNOWLEDGMENT

The authors would like to thank Dr. Kazuhiko Minematsu for providing us data of disjunct matrices generated by the STD method.

REFERENCES

[1] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 1109. Berlin, Germany: Springer, 1996, pp. 1–15.

[2] *The Keyed-Hash Message Authentication Code (HMAC)*, document FIPS PUB 198-1, 2008.

[3] T. Iwata and K. Kurosawa, "OMAC: One-key CBC MAC," in *Fast Software Encryption* (Lecture Notes in Computer Science), vol. 2887. Berlin, Germany: Springer, 2003, pp. 129–153.

[4] *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication* document NIST Special Publication 800-38B, 2005.

[5] J. Katz and A. Y. Lindell, "Aggregate message authentication codes," in *Topics in Cryptology—CT-RSA* (Lecture Notes in Computer Science), vol. 4964. Berlin, Germany: Springer, 2008, pp. 155–169.

[6] D. Z. Du and F. K. Hwang, "Combinatorial group testing and its applications," in *Series on Applied Mathematics*, vol. 12, 2nd ed. Singapore: World Scientific, 2000.

[7] S. Hirose and J. Shikata, "Non-adaptive group-testing aggregate MAC scheme," in *Information Security Practice and Experience* (Lecture Notes in Computer Science), vol. 11125, C. Su and H. Kikuchi, Eds. Cham, Switzerland: Springer, 2018, pp. 357–372.

[8] O. Eikemeier, M. Fischlin, J. F. Götzmann, A. Lehmann, D. Schröder, P. Schröder, and D. Wagner, "History-free aggregate message authentication codes," in *Security and Cryptography for Networks* (Lecture Notes in Computer Science), vol. 6280. Berlin, Germany: Springer, 2010, pp. 309–328.

[9] S. Sato, S. Hirose, and J. Shikata, "Sequential aggregate MACs with detecting functionality revisited," in *Network and System Security* (Lecture Notes in Computer Science), vol. 11928, J. K. Liu and X. Huang, Eds. Cham, Switzerland: Springer, 2019, pp. 387–407.

[10] S. Sato, S. Hirose, and J. Shikata, "Sequential aggregate MACs from any MACs: Aggregation and detecting functionality," *J. Internet Services Inf. Secur.*, vol. 9, no. 1, pp. 2–23, 2019.

[11] D. Ma and G. Tsudik, "Extended abstract: Forward-secure sequential aggregate authentication," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 86–91.

[12] D. Ma and G. Tsudik, "A new approach to secure logging," *ACM Trans. Storage*, vol. 5, no. 1, pp. 1–21, Mar. 2009.

[13] S. Hirose and H. Kuwakado, "Forward-secure sequential aggregate message authentication revisited," in *Provable Security* (Lecture Notes in Computer Science), vol. 8782, S. S. M. Chow, J. K. Liu, L. C. K. Hui, and S. Yiu, Eds. Cham, Switzerland: Springer, 2014, pp. 87–102.

[14] M. T. Goodrich, M. J. Atallah, and R. Tamassia, "Indexing information for data forensics," in *Applied Cryptography and Network Security* (Lecture Notes in Computer Science), vol. 3531. Berlin, Germany: Springer, 2005, pp. 206–221.

[15] K. Minematsu, "Efficient message authentication codes with combinatorial group testing," in *Computer Security—ESORICS* (Lecture Notes in Computer Science), vol. 9326, G. Pernul, P. Y. A. Ryan, and E.R. Weippl, Eds. Cham, Switzerland: Springer, 2015, pp. 185–202.

[16] K. Minematsu and N. Kamiya, "Symmetric-key corruption detection: When XOR-MACs meet combinatorial group testing," in *Computer Security—ESORICS* (Lecture Notes in Computer Science), vol. 11735, K. Sako, S. A. Schneider, and P. Y. A. Ryan, Eds. Cham, Switzerland: Springer, 2019, pp. 595–615.

[17] J. Black and P. Rogaway, "A block-cipher mode of operation for parallelizable message authentication," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 2332, L. R. Knudsen, Ed. Berlin, Germany: Springer, 2002, pp. 384–397.

[18] P. Rogaway, "Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC," in *Advances in Cryptology—ASIACRYPT* (Lecture Notes in Computer Science), vol. 3329, P. J. Lee, Ed. Berlin, Germany: Springer, 2004, pp. 16–31.

[19] S. Sato and J. Shikata, "Interactive aggregate message authentication scheme with detecting functionality," in *Proc. 33rd Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, 2019, pp. 1316–1328.

[20] Y. Ogawa, S. Sato, J. Shikata, and H. Imai, "Aggregate message authentication codes with detecting functionality from biorthogonal codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 868–873.

[21] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proc. 1st ACM Conf. Comput. Commun. Secur. - CCS*, 1993, pp. 62–73.

[22] N. Thierry-Mieg, "A new pooling strategy for high-throughput screening: The Shifted transversal design," *BMC Bioinf.*, vol. 7, no. 28, p. 28, 2006.

SHOICHI HIROSE received the B.E., M.E., and D.E. degrees in information science from Kyoto University, Kyoto, Japan, in 1988, 1990, and 1995, respectively.

From 1990 to 1998, he was a Research Associate with the Faculty of Engineering, Kyoto University. From 1998 to 2005, he was a Lecturer with the Graduate School of Informatics, Kyoto University. From 2005 to 2009, he was an Associate Professor with the Faculty of Engineering, University of Fukui. Since 2009, he has been a Professor with the Graduate School of Engineering, University of Fukui. His current interests include cryptography and information security.

JUNJI SHIKATA (Member, IEEE) received the B.S. and M.S. degrees in mathematics from Kyoto University, Kyoto, Japan, in 1994 and 1997, respectively, and the Ph.D. degree in mathematics from Osaka University, Osaka, Japan, in 2000.

From 2000 to 2002, he was a Postdoctoral Fellow with the Institute of Industrial Science, The University of Tokyo, Tokyo, Japan. Since 2002, he has been with the Graduate School of Environment and Information Sciences, Yokohama National University, Yokohama, Japan. From 2008 to 2009, he was a Visiting Researcher with the Department of Computer Science, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland. He is currently a Professor with Yokohama National University. His research interests include cryptology, information theory, theoretical computer science, and computational number theory.

Dr. Shikata received several awards, including the Wilkes Award 2006 from the British Computer Society, and the Young Scientists' Prize, the Commendation for Science and Technology by the Minister of Education, Culture, Sports, Science and Technology, Japan, in 2010.