Non-adaptive Group-Testing Aggregate MAC Scheme

Shoichi Hirose[1] and Junji Shikata[2]

[1] University of Fukui, Japan
[2] Yokohama National University, Japan

Abstract. This paper applies non-adaptive group testing to aggregate message authentication code (MAC) and introduces non-adaptive group-testing aggregate MAC. After formalization of its syntax and security requirements, simple and generic construction is presented, which can be applied to any aggregate MAC scheme formalized by Katz and Lindell in 2008. Then, two instantioations of the construction is presented. One is based on the aggregate MAC scheme by Katz and Lindell and uses addition for tag aggregate. The other uses cryptographic hashing for tag aggregate. Provable security of the generic construction and two instantiations are also discussed.

Keywords: Message authentication · Aggregate · Group testing · Provable security

1 Introduction

Background. A message authentication code (MAC) is a tag attached to a 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, 5] and CMAC [8, 13].

An aggregate MAC scheme allows one to aggregate multiple tags to multiple messages into a shorter tag. It is possible to verify the validity of the multiple messages only with the single tag. It is impossible in general, however, to identify invalid messages once the multiple messages are judged invalid with respect to the single tag.

It is expected that the problem above can be solved with group testing [3]. Group testing is a method to be able to verify if each sample is negative or positive with a smaller number of tests than a naive method to test each sample infividually on the assumption that the number of positive samples is at most a constant. In group testing, each test involves a subset of the given samples. The result of a test is negative if and only if all the involved samples are negative. The group testing is called adaptive if one can choose samples to be tested after one sees the result of the previous test and is called non-adaptive otherwise.

Contribution. This paper applies non-adaptive group testing to aggregate MAC and introduces non-adaptive group testing aggregate MAC (GTA MAC).

First, GTA MAC and its security requirements are formalized. The security requirements are unforgeability and identifiability. Unforgeability means that a message is judged invalid by the group testing if the tag to the message is not generated by a legitimate user. Identifiability is composed of completeness and soundness. Completeness captures the notion that a group-testing for pairs of a message and a tag should judge a pair valid if it is valid. Soundness captures the notion that a group-testing should judge a pair invalid if it is invalid. Then, simple and generic construction of a GTA MAC scheme is presented. It can be applied to any aggregate MAC scheme formalized by Katz and Lindell [9]. The generic construction produces a GTA MAC scheme satisfying unforgeability and completeness from any unforgeable aggregate MAC scheme. Finally, two instantiations are presented: One is from the Katz-Lindell aggregate MAC scheme [9] and the other is from an aggregate MAC scheme using hashing for aggregate. The former does not satisfy soundness. The latter is shown to satisfy soundness if the underlying hash function is a random oracle.

Related Work. Aggregate MAC and its security requirement were first formalized by Katz and Lindell [9]. They also proposed an aggregate MAC scheme and proved its security on the assumption that the underlying MAC function is unforgeable. Their scheme aggregates tags by their addition. The formalization of GTA MAC in the paper is based on that of aggregate MAC by Katz and Lindell.

Sequential aggregate MAC and its security requirement were formalized by Eikemeier et al. [4]. They also presented a provably secure sequential aggregate MAC scheme. Forward-secure sequential aggregate MAC was introduced by Ma and Tsudik [10]. It was also discussed by Ma and Tsudik [11] and Hirose and Kuwakado [7]. A typical application of the forward-secure sequential aggregate MAC is secure audit log.

The group testing was already applied to MAC schemes by Goodrich et al. [6] and Minematsu [12]. The major difference between their approach and ours is that their schemes do not, precisely speaking, aggregate tags for messages. Their schemes compute a tag of a subset of messages for each test in group testing. Minematsu [12] proposed a scheme based on PMAC [2, 14] aiming at reduction of amount of computation required to compute tags for group testing.

Organization. This paper is organized as follows. Section 2 gives notations and introduces MAC functions and non-adaptive group testing. Section 3 introduces the syntax and security requirement of aggregate MAC. It also describes the aggregate MAC scheme proposed by Katz and Lindell. Section 4 formalizes the syntax and security requirements of GTA MAC. Section 5 gives a method for generic construction of GTA MAC schemes. It also describes provable security for the generic construction. Section 6 presents a GTA MAC scheme based on the Katz-Lindell aggregate MAC scheme. Section 7 presents another GTA MAC scheme using a cryptographic hash function for aggregate. Section 8 gives a brief concluding remark.

2 Preliminaries

2.1 Notations

Selecting an element s uniformly at random from a set S is denoted by $s \twoheadleftarrow S$.

For $\{0,1\}$-sequences x and y, $x\|y$ represents their concatenation.

Let $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$ and $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$ be vectors such that $v_i \in \{0,1\}$ and $w_i \in \{0,1\}^l$ for $1 \leq i \leq n$. Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in X^n$ for some set X. $\langle \boldsymbol{v}, \boldsymbol{w} \rangle$ represents inner product of $\boldsymbol{v}$ and $\boldsymbol{w}$, that is, $\langle \boldsymbol{v}, \boldsymbol{w} \rangle = \bigoplus_{i=1}^{n} v_i w_i$. Let $\langle\langle \boldsymbol{v}, \boldsymbol{w} \rangle\rangle = w_{i_1} \| w_{i_2} \| \cdots \| w_{i_d}$, and $\boldsymbol{v} \boxdot \boldsymbol{x} = (x_{i_1}, x_{i_2}, \ldots, x_{i_d})$, where $1 \leq i_1 < i_2 < \cdots < i_d \leq n$, and $v_i = 1$ if $i \in \{i_1, i_2, \ldots, i_d\}$ and $v_i = 0$ otherwise.

For vectors $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$ and $\boldsymbol{v}' = (v'_1, v'_2, \ldots, v'_n)$ in $\{0,1\}^n$, $\boldsymbol{v} \preceq \boldsymbol{v}'$ if $v_i \leq v'_i$ for $1 \leq i \leq n$.

2.2 MAC Functions

A MAC function is defined to be a keyed function $f : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$, where $\mathcal{K}$ is its key space, $\mathcal{M}$ is its messsage space, and $\mathcal{T}$ is its tag space. $f(K, \cdot)$ is often denoted by $f_K(\cdot)$. The security requirement for a MAC function is unforgeability. Let A be an adversary against f. A is given access to the tagging oracle f_K and the corresponding verification oracle V_K, where $K \twoheadleftarrow \mathcal{K}$. The tagging oracle f_K returns $f_K(M)$ in reply to a query $M \in \mathcal{M}$. The verification oracle V_K, in reply to a query $(M, T) \in \mathcal{M} \times \mathcal{T}$, returns $\top$ if $f_K(M) = T$ and $\bot$ otherwise. It is assumed that A does not make a query on $(M, T) \in \mathcal{M} \times \mathcal{T}$ once it gets T from f_K as a reply to its query M. Let $\mathsf{Forge}(A)$ represent an event that A succeeds in making a query to which V_K returns $\top$. The mac-advantage of A against f is defined as

$$\mathrm{Adv}_f^{\mathrm{mac}}(A) \triangleq \Pr\left[\mathsf{Forge}(A)\right] \quad .$$

2.3 Non-adaptive Group Testing

A non-adaptive group-testing algorithm with n samples and u tests can be represented by a $u \times n$ $\{0,1\}$-matrix, which is called a group-testing matrix. For $1 \leq i \leq u$ and $1 \leq j \leq n$, the i-th test involves the j-th sample if and only if the (i,j)-th element of the corresponding group-testing matrix equals 1. Each sample is either positive or negative. It is assumed that the result of a test is negative if all the samples involved in the test are negative and positive otherwise. All of the positive samples can be detected by the following simple procedure:

1. $J \leftarrow \{1, 2, \ldots, n\}$, where $j \in \{1, 2, \ldots, n\}$ represents the j-th sample.
2. For $1 \leq i \leq u$, if the result of the i-th test is negative, then $J \leftarrow J \setminus \{j_{i,1}, j_{i,2}, \ldots, j_{i,w_i}\}$, where $\{j_{i,1}, j_{i,2}, \ldots, j_{i,w_i}\}$ are all of the samples involved in the i-th test.
3. Output J.

The output J of the procedure presented above includes all the positive samples. It may also include (some of) the negative samples in general.

Definition 1 (d-disjunct). *A $\{0,1\}$-matrix $\boldsymbol{G}$ is said to be d-disjunct if, any d columns of $\boldsymbol{G}$ do not cover any other column of $\boldsymbol{G}$. Here, d columns $\boldsymbol{g}^{\mathrm{c}}_{j_1}, \boldsymbol{g}^{\mathrm{c}}_{j_2}, \ldots, \boldsymbol{g}^{\mathrm{c}}_{j_d}$ are said to cover a column $\boldsymbol{g}^{\mathrm{c}}$ if $\boldsymbol{g}^{\mathrm{c}} \preceq \boldsymbol{g}^{\mathrm{c}}_{j_1} \vee \boldsymbol{g}^{\mathrm{c}}_{j_2} \vee \cdots \vee \boldsymbol{g}^{\mathrm{c}}_{j_d}$, where $\vee$ is the component-wise disjunction.*

d-disjunct matrices are useful for group testing. If the group testing matrix is d-disjunct and at most d of n samples are positive, then the set J computed by the procedure above does not contain any negative samples.

3 Aggregate MAC

3.1 Syntax

An aggregate MAC scheme is composed of the following algorithms:

Key generation $k \leftarrow \mathsf{KG}(1^p)$.
: This algorithm takes as input a security parameter p and produces a secret key k.

Tagging $t \leftarrow \mathsf{Tag}(k_{id}, id, m)$.
: This algorithm takes as input a pair of an ID and a message (id, m) and a secret key k_{id} corresponding to id, and produces as output a tag t.

Aggregate $T \leftarrow \mathsf{Agg}((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$.
: This algorithm takes tuples of an ID, a message, and a tag (id_i, m_i, t_i)'s as input and produces an aggregate tag T as output. Notice that it is not given secret keys used by the tagging algorithm Tag.

Verification $d \leftarrow \mathsf{Ver}((k_1, \ldots, k_n), ((id_1, m_1), \ldots, (id_n, m_n)), T)$.
: This algorithm takes pairs of an ID and a message (id_i, m_i)'s and an aggregate tag T as input and checks their validity with respect to the keys corresponding to the given IDs. Here, k_i is a key corresponding to id_i for $1 \leq i \leq n$. The decision d is either $\top$ or $\bot$. If $d = \top$, the pair $((id_1, m_1), \ldots, (id_n, m_n))$ and T are judged as valid with respect to $(k_1, \ldots, k_n)$. Othewise, they are judged invalid.

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

3.2 Security Requirement

The security requirement of an aggregate MAC scheme $\mathsf{AM} \triangleq (\mathsf{KG}, \mathsf{Tag}, \mathsf{Agg}, \mathsf{Ver})$ is unforgeability. An adversary against AM is given access to the oracles listed below:

Tagging The tagging oracle $\mathcal{TG}$ receives a pair of ID and a message (id, m) as a query and returns a tag t, where $t \leftarrow \mathsf{Tag}(k_{id}, id, m)$.

Key disclosure The key-disclosure oracle $\mathcal{KD}$ receives an ID id as a query and returns the corresponding key k_{id}.

Verification The verification oracle $\mathcal{VR}$ receives $(((id_1, m_1), \ldots, (id_n, m_n)), T)$ as a query and returns d, where

$$d \leftarrow \mathsf{Ver}((k_1, \ldots, k_n), ((id_1, m_1), \ldots, (id_n, m_n)), T) \ .$$

Definition 2 (Unforgeability). *Let A be an adversary against an aggregate MAC scheme AM. A is given access to $\mathcal{TG}, \mathcal{KD}, \mathcal{VR}$, and is allowed to make multiple queries adaptively to each of them. Let $\mathsf{Forge}(A)$ be an event that A succeeds in asking $\mathcal{VR}$ a query $(((id_1, m_1), \ldots, (id_n, m_n)), T)$ satisfying the following conditions:*

- $\mathsf{Ver}((k_1, \ldots, k_n), ((id_1, m_1), \ldots, (id_n, m_n)), T) = \top$
- *Before asking $(((id_1, m_1), \ldots, (id_n, m_n)), T)$, for some $1 \leq j \leq n$, A asks neither (id_j, m_j) to $\mathcal{TG}$ nor id_j to $\mathcal{KD}$.*

Then, the advantage of A against AM with respect to unforgeability is defined as

$$\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{AM}}(A) \triangleq \Pr[\mathsf{Forge}(A)] \ .$$

An aggregate MAC scheme AM is informally said to satisfy unforgeability if $\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{AM}}(A)$ is negligibly small for any adversary A with realistic computational resources.

3.3 Katz-Lindell Aggregate MAC Scheme

An aggregate MAC scheme proposed by Katz and Lindell [9] is described in this section. Here, their scheme is called $\mathsf{KL\text{-}AM}$.

Scheme. Let $F : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ be a MAC function.

- The key generation algorithm just picks up a secret key uniformly at random from $\mathcal{K}$ for each user.
- For an input (id, m), the tagging algorithm returns $t \triangleq F(k_{id}, m)$.
- For an input $((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$, the aggregate algorithm returns $T = t_1 \oplus t_2 \oplus \cdots \oplus t_n$.
- For an input $(((id_1, m_1), \ldots, (id_n, m_n)), T)$, the verification algorithm returns d such that

$$d = \begin{cases} \top & \text{if } T = F(k_1, m_1) \oplus \cdots \oplus F(k_n, m_n), \\ \bot & \text{otherwise.} \end{cases}$$

Security. Katz and Lindell [9] showed that their aggregate MAC scheme satisfies unforgeability assuming a single query to the verification oracle. We show for later use that their scheme satisfies unforgeability assuming multiple queries to the verification oracle.

Theorem 1 (Unforgeability). *For the Katz-Lindell aggregate MAC scheme* KL-AM*, let ℓ be the number of the users. For any adversary A against* KL-AM *running in time at most s, making at most q_{t} queries to its tagging oracle, and making at most q_{v} queries to its verification oracle, there exists some adversary B against F such that*

$$\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{KL\text{-}AM}}(A) \leq \ell \cdot \mathrm{Adv}^{\mathrm{mac}}_{F}(B) \ ,$$

where B runs in time at most $s + S_F(q_{\mathrm{t}} + \ell q_{\mathrm{v}})$, making at most q_{t} queries to its tagging oracle, and making at most q_{v} queries to its verification oracle. S_F is time required to compute F.

Proof. The adversary B attacks F by making use of an adversary A against KL-AM. B has oracle access to the tagging oracle F_K and the verification oracle V_K, where $K \twoheadleftarrow \mathcal{K}$.

B first picks up a user id_r uniformly at random among ℓ users. B also selects a secret key uniformly at random from $\mathcal{K}$ for each of the other $(\ell - 1)$ users. Then, B runs A.

For a tagging query on the user id_r made by A, B transfers it to F_K and returns the reply from F_K to A. For a tagging query on a user other than id_r made by A, B computes the tag using the corresponding secret key chosen by itself and returns it to A. If A makes a key-disclosure query on a user other than id_r, then B simply returns the corresponding secret key to A. If A makes a key-disclosure query on id_r, then B aborts.

Suppose that A succeeds in forgery. Then, A makes a verification query such that the verification oracle returns $\top$ in reply to it and, for some (id', m') included in it, A asks neither (id', m') to the tagging oracle nor id' to the key-disclosure oracle prior to it. Let Hit be the event such that $id' = id_r$. The conditional probability that Hit occurs when A succeeds in forgery is at least $1/\ell$.

Suppose that A succeeds in forgery and that Hit occurs. For a verification query from A not related to id_r, B verifies it by itself and returns the result. For a verification query from A including (id_r, m_r), B computes a tag t_r for (id_r, m_r) from the query and the secret keys of the other users and asks (m_r, t_r) to its verification query. Then, B makes at most q_{v} queries to its verification oracle, which returns $\top$ for at least one of them. Thus,

$$\begin{aligned}
\Pr[\mathsf{Forge}(B)] &= \Pr[\mathsf{Forge}(A) \cap \mathsf{Hit}] \\
&= \Pr[\mathsf{Hit} \mid \mathsf{Forge}(A)] \Pr[\mathsf{Forge}(A)] \\
&\geq \frac{1}{\ell} \Pr[\mathsf{Forge}(A)]
\end{aligned}$$

and $\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{KL\text{-}AM}}(A) \leq \ell \cdot \mathrm{Adv}^{\mathrm{mac}}_{F}(B)$. □

4 Group-Testing Aggregate MAC

4.1 Syntax

A group-testing aggregate MAC (GTA MAC) scheme consists of the following algorithms:

Key generation $k \leftarrow \mathsf{KG}(1^p)$.
This algorithm takes as input a security parameter p and produces a secret key k.

Tagging $t \leftarrow \mathsf{Tag}(k_{id}, id, m)$.
This algorithm takes as input a pair of an ID and a message (id, m) and a secret key k_{id} corresponding to id, and produces as output a tag t.

Group-testing aggregate This algorithm GTA takes tuples of an ID, a message, and a tag (id_j, m_j, t_j)'s as input and produces a tuple of aggregate tags $T_1, \ldots, T_u$ as output:

$$(T_1, \ldots, T_u) \leftarrow \mathsf{GTA}((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n)) \ .$$

Notice that it is not given secret keys used by the tagging algorithm.

Group-testing verification This algorithm GTV takes pairs of an ID and a message (id_j, m_j)'s and a tuple of aggregate tags T_i's as input and tries to identify invalid pairs of an ID and a message using the corresponding keys:

$$J \leftarrow \mathsf{GTV}((k_1, \ldots, k_n), ((id_1, m_1), \ldots, (id_n, m_n)), (T_1, \ldots, T_u)) \ .$$

The output J of this algorithm is a set of $(id_{j'}, m_{j'})$'s which are judged invalid.

For $((id_1, m_1), \ldots, (id_n, m_n))$ and $(T_1, \ldots, T_u)$, if $t_j = \mathsf{Tag}(k_j, id_j, m_j)$ for $1 \leq j \leq n$ and $(T_1, \ldots, T_u) = \mathsf{GTA}((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$, then $\mathsf{GTV}((k_1, \ldots, k_n), ((id_1, m_1), \ldots, (id_n, m_n)), (T_1, \ldots, T_u)) = \emptyset$.

4.2 Security Requirement

The security requirements of a GTA MAC scheme $\mathsf{GTAM} \triangleq (\mathsf{KG}, \mathsf{Tag}, \mathsf{GTA}, \mathsf{GTV})$ are unforgeability and identifiability. An adversary against GTAM is given access to the oracles listed below:

Tagging This oracle $\mathcal{TG}$ receives a pair of ID and a message (id, m) as a query and returns a tag $t \leftarrow \mathsf{Tag}(k_{id}, id, m)$, where k_{id} is the secret key of the user id.

Key disclosure This oracle $\mathcal{KD}$ receives an ID id as a query and returns the corresponding secret key k_{id}.

Group-testing verification Given $((id_1, m_1), \ldots, (id_n, m_n)), (T_1, \ldots, T_u))$ as a query, this oracle $\mathcal{GTV}$ returns

$$J \leftarrow \mathsf{GTV}((k_1, \ldots, k_n), ((id_1, m_1), \ldots, (id_n, m_n)), (T_1, \ldots, T_u)) \ .$$

Unforgeability. Let A be an adversary against a GTA MAC scheme GTAM. A is given access to the oracles $\mathcal{TG}$, $\mathcal{KD}$, $\mathcal{GTV}$, and is allowed to make multiple queries adaptively to each of them. Let $\mathsf{GTForge}(A)$ be an event that A succeeds in asking $\mathcal{GTV}$ a query $(((id_1, m_1), \ldots, (id_n, m_n)), (T_1, \ldots, T_u))$ satisfying the following conditions: There exists some $1 \leq j \leq n$ such that

- $(id_j, m_j) \notin \mathsf{GTV}((k_1, \ldots, k_n), ((id_1, m_1), \ldots, (id_n, m_n)), (T_1, \ldots, T_u))$,
- Before asking $(((id_1, m_1), \ldots, (id_n, m_n)), (T_1, \ldots, T_u))$, A asks neither (id_j, m_j) to $\mathcal{TG}$ nor id_j to $\mathcal{KD}$.

Then, the advantage of A against GTAM with respect to unforgeability is defined as

$$\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{GTAM}}(A) \triangleq \Pr[\mathsf{GTForge}(A)] \ .$$

GTAM is informally said to satisfy unforgeability if $\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{GTAM}}(A)$ is negligibly small for any adversary A with realistic computational resources.

Identifiability. For identifiability, completeness and soundness are introduced. Let A be an adversary for identifiability. Let us consider the experiments presented in Fig. 1 and in Fig. 2 for completeness and soundness, respectively. Steps from 1 to 4 are identical in both of the experiments. The adversary A is given access to the tagging oracle $\mathcal{TG}$ and the key-disclosure oracle $\mathcal{KD}$. Then, A outputs tuples $((id_{j_1}, m_{j_1}, t_{j_1}), \ldots, (id_{j_n}, m_{j_n}, t_{j_n}))$, and the group testing is applied to them. Completeness requires that any valid tuple $(id_{j_v}, m_{j_v}, t_{j_v})$ is judged valid by the group testing. On the other hand, soundness requires that any invalid tuple is judged invalid. The advantage of A against GTAM with respect to completeness and soundness is defined as

$$\mathrm{Adv}^{\text{id-c}}_{\mathsf{GTAM}}(A) \triangleq \Pr\left[\mathtt{Exp}^{\text{id-c}}_{\mathsf{GTAM}}(A) = 1\right] \ ,$$

and

$$\mathrm{Adv}^{\text{id-s}}_{\mathsf{GTAM}}(A) \triangleq \Pr\left[\mathtt{Exp}^{\text{id-s}}_{\mathsf{GTAM}}(A) = 1\right] \ ,$$

respectively.

1: $b \leftarrow 0$
2: $((id_{j_1}, m_{j_1}, t_{j_1}), \ldots, (id_{j_n}, m_{j_n}, t_{j_n})) \leftarrow A^{\mathcal{TG}, \mathcal{KD}}$
3: $(T_1, \ldots, T_u) \leftarrow \mathsf{GTA}((id_{j_1}, m_{j_1}, t_{j_1}) \ldots, (id_{j_n}, m_{j_n}, t_{j_n}))$
4: $J \leftarrow \mathsf{GTV}((k_{j_1}, \ldots, k_{j_n}), (id_{j_1}, m_{j_1}) \ldots, (id_{j_n}, m_{j_n}), (T_1, \ldots, T_u))$
5: **if** $J \cap \{(id_{j_v}, m_{j_v}) \mid t_{j_v} = \mathsf{Tag}(k_{j_v}, id_{j_v}, m_{j_v})\} \neq \emptyset$ **then**
6: $\quad b \leftarrow 1$
7: **end if**
8: **return** b

Fig. 1: Experiment $\mathtt{Exp}^{\text{id-c}}_{\mathsf{GTAM}}(A)$

$b \leftarrow 0$
$((id_{j_1}, m_{j_1}, t_{j_1}), \ldots, (id_{j_n}, m_{j_n}, t_{j_n})) \leftarrow A^{\mathcal{TG},\mathcal{KD}}$
$(T_1, \ldots, T_u) \leftarrow \mathsf{GTA}((id_{j_1}, m_{j_1}, t_{j_1}) \ldots, (id_{j_n}, m_{j_n}, t_{j_n}))$
$J \leftarrow \mathsf{GTV}((k_{j_1}, \ldots, k_{j_n}), (id_{j_1}, m_{j_1}) \ldots, (id_{j_n}, m_{j_n}), (T_1, \ldots, T_u))$
if $\{(id_{j_v}, m_{j_v}) \mid t_{j_v} \neq \mathsf{Tag}(k_{j_v}, id_{j_v}, m_{j_v})\} \setminus J \neq \emptyset$ **then**
    $b \leftarrow 1$
end if
return b

Fig. 2: Experiment $\mathsf{Exp}^{\text{id-s}}_{\mathsf{GTAM}}(A)$

5 Generic Construction of GTA MAC Scheme

This section first presents generic construction of a GTA MAC scheme from an aggregate MAC scheme and a group-testing matrix. Then, it discusses the security of the GTA MAC scheme.

5.1 Generic Construction

Let $\mathsf{AM} = (\mathsf{KG}, \mathsf{Tag}, \mathsf{Agg}, \mathsf{Ver})$ be an aggregate MAC scheme. Let $\boldsymbol{G}$ be a $u \times n$ group-testing matrix, where $\boldsymbol{G} = (g_{i,j})$ for $1 \leq i \leq u$ and $1 \leq j \leq n$ and $\boldsymbol{g}_i = (g_{i,1}, \ldots, g_{i,n}) \in \{0,1\}^n$ is the i-th row of $\boldsymbol{G}$ for $1 \leq i \leq u$. A GTA MAC scheme $\mathsf{GTAM}_{\mathrm{g}} = (\mathsf{KG}_{\mathrm{g}}, \mathsf{Tag}_{\mathrm{g}}, \mathsf{GTA}_{\mathrm{g}}, \mathsf{GTV}_{\mathrm{g}})$ is constructed from AM and $\boldsymbol{G}$ as follows:

- $\mathsf{KG}_{\mathrm{g}} \triangleq \mathsf{KG}$.
- $\mathsf{Tag}_{\mathrm{g}} \triangleq \mathsf{Tag}$.
- $(T_1, \ldots, T_u) \leftarrow \mathsf{GTA}_{\mathrm{g}}((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$. where, for $1 \leq i \leq u$, $T_i \leftarrow \mathsf{Agg}(\boldsymbol{g}_i \boxdot ((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n)))$.
- $J \leftarrow \mathsf{GTV}_{\mathrm{g}}((k_1, \ldots, k_n), ((id_1, m_1), \ldots, (id_n, m_n)), (T_1, \ldots, T_u))$, where
  1. $J \leftarrow \{(id_1, m_1), \ldots, (id_n, m_n)\}$.
  2. For $1 \leq i \leq u$, if

$$\mathsf{Ver}(\boldsymbol{g}_i \boxdot (k_1, \ldots, k_n), \boldsymbol{g}_i \boxdot ((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n)), T_i) = \top \ ,$$

  then

$$J \leftarrow J \setminus \{(id_j, m_j) \mid 1 \leq j \leq n \wedge g_{i,j} = 1\} \ .$$

5.2 Unforgeability

The following theorem says that generic construction produces an unforgeable GTA MAC scheme from any unforgeable aggregate MAC scheme.

Theorem 2. *For the GTA MAC scheme* $\mathsf{GTAM}_{\mathrm{g}}$*, let* ℓ *be the number of the users. For any adversary* A *against* $\mathsf{GTAM}_{\mathrm{g}}$ *running in time at most* s*, making*

at most q_{t} queries to its tagging oracle, making at most q_{k} queries to its key-disclosure oracle, and making at most q_{v} queries to its group-testing verification oracle, there exists some adversary B against AM *with ℓ users such that*

$$\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{GTAM}_{\mathrm{g}}}(A) \leq \mathrm{Adv}^{\mathrm{uf}}_{\mathsf{AM}}(B) \ ,$$

where B runs in time at most s, making at most q_{t} queries to its tagging oracle, making at most q_{k} queries to its key-disclosure oracle, and making at most $u\,q_{\mathrm{v}}$ queries to its verification oracle.

Proof. The adversary B against AM tries forgery by making use of the adversary A against $\mathsf{GTAM}_{\mathrm{g}}$. B has oracle access to the tagging oracle, the key-disclosure oracle, and the verification oracle.

B simply runs A. For a tagging query made by A, B transfers it to its tagging oracle and returns the reply to A. For a key-disclosure query made by A, B also transfers it to its key-disclosure oracle and returns the reply to A. For a group-testing verification query made by A, B executes $\mathsf{GTV}_{\mathrm{g}}$ using its verification oracle u times.

Suppose that A succeeds in forgery and $(((id_1, m_1), \ldots, (id_n, m_n)), (T_1, \ldots, T_u))$ is a successful forgery. Then, there exists some $1 \leq j \leq n$ such that

- $(id_j, m_j) \notin \mathsf{GTV}_{\mathrm{g}}((k_1, \ldots, k_n), ((id_1, m_1), \ldots, (id_n, m_n)), (T_1, \ldots, T_u))$, and
- Before asking $((id_1, m_1), \ldots, (id_n, m_n)), (T_1, \ldots, T_u))$, A asks neither (id_j, m_j) to its tagging oracle nor id_j to its key-disclosure oracle.

It implies that there exists some $1 \leq i \leq u$ such that the i-th test involves (id_j, m_j) and passes the verification. Thus, the i-th test is a successful query made by B to its verification oracle. □

5.3 Identifiability

An adversary A is said to be d-dishonest if A outputs $((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$ such that $|\{(id_j, m_j) \mid t_j \neq \mathsf{Tag}_{\mathrm{g}}(k_j, id_j, m_j)\}| \leq d$ in $\mathsf{Exp}^{\text{id-c}}_{\mathsf{GTAM}_{\mathrm{g}}}$ or $\mathsf{Exp}^{\text{id-s}}_{\mathsf{GTAM}_{\mathrm{g}}}$.

Completeness. The theorem below says that the GTA MAC scheme $\mathsf{GTAM}_{\mathrm{g}}$ satisfies completeness against any d-dishonest adversary if the group-testing matrix is d-disjunct.

Theorem 3 (Completeness). *For the GTA MAC scheme $\mathsf{GTAM}_{\mathrm{g}}$, suppose that the group-testing matrix $\boldsymbol{G}$ is d-disjunct. Then, for any d-dishonest adversary A,*

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

Proof. Let A be any d-dishonest adversary. Suppose that A outputs $((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$ in $\mathsf{Exp}^{\text{id-c}}_{\mathsf{GTAM}_{\mathrm{g}}}(A)$ and let $V = \{(id_j, m_j) \mid t_j = \mathsf{Tag}_{\mathrm{g}}(k_j, id_j, m_j)\}$. Since the group-testing matrix $\boldsymbol{G}$ is d-disjuct and A is d-dishonest, for any $(id, m) \in V$, there exists some test in $\boldsymbol{G}$ involving (id, m) and no invalid pairs. Thus, $V \cap J = \emptyset$, where J is the set computed in $\mathsf{Exp}^{\text{id-c}}_{\mathsf{GTAM}_{\mathrm{g}}}(A)$. □

Soundness. The GTA MAC scheme $\mathsf{GTAM}_{\mathrm{g}}$ may not satisfy soundness. It depends on how to aggregate tags.

Let us consider the following adversary $\tilde{A}$ in $\mathsf{Exp}^{\text{id-s}}_{\mathsf{GTAM}_{\mathrm{g}}}$. $\tilde{A}$ first obtains valid (id_j, m_j, t_j) such that $t_j = \mathsf{Tag}_{\mathrm{g}}(k_j, id_j, m_j)$ using its tagging oracle for $1 \leq j \leq n$. Let $T_i = \mathsf{Agg}(\boldsymbol{g}_i \boxdot ((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n)))$ for $1 \leq i \leq u$. Suppose that $\tilde{A}$ succeeds in finding $((id_1, m_1, \tilde{t}_1), \ldots, (id_n, m_n, \tilde{t}_n)))$ such that, for some i^*, $\boldsymbol{g}_{i^*} \boxdot ((id_1, m_1, \tilde{t}_1), \ldots, (id_n, m_n, \tilde{t}_n)) \neq \boldsymbol{g}_{i^*} \boxdot ((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$ and $T_{i^*} = \mathsf{Agg}(\boldsymbol{g}_{i^*} \boxdot ((id_1, m_1, \tilde{t}_1), \ldots, (id_n, m_n, \tilde{t}_n)))$. Then, the result of the i^*-th test $\boldsymbol{g}_{i^*}$ is valid, and there exists some j^* such that $\tilde{t}_{j^*} \neq \mathsf{Tag}(k_{j^*}, id_{j^*}, m_{j^*})$ and $(id_{j^*}, m_{j^*}) \notin J$.

6 GTA MAC Scheme Based on Katz-Lindell Aggregate MAC

From the generic construction, it is straightforward to obtain a GTA MAC sheme based on the Katz-Lindell aggregate MAC scheme. Let us call it $\mathsf{GTAM}_{\mathrm{X}}$.

$\mathsf{GTAM}_{\mathrm{X}}$ is unforgeable if the underlying MAC function is unforgeable. For identifiability, $\mathsf{GTAM}_{\mathrm{X}}$ satisfies completeness, while it does not satisfy soundness.

6.1 Scheme

Let $F : \mathcal{K} \times \mathcal{M} \to \{0,1\}^\tau$ be a MAC function. The key generation and tagging algorithms of $\mathsf{GTAM}_{\mathrm{X}}$ are identical to those of the Katz-Lindell scheme. It is assumed that the group-testing aggregate algorithm of $\mathsf{GTAM}_{\mathrm{X}}$ is based on a pre-specified $u \times n$ group-testing matrix $\boldsymbol{G}$. Let $\boldsymbol{G} = (g_{i,j})$ for $1 \leq i \leq u$ and $1 \leq j \leq n$ and $\boldsymbol{g}_i = (g_{i,1}, \ldots, g_{i,n}) \in \{0,1\}^n$ be the i-th row of $\boldsymbol{G}$ for $1 \leq i \leq u$.

- The key generation algorithm just picks up a secret key uniformly at random from $\mathcal{K}$ for each user.
- For an input (id, m), the tagging algorithm returns $t \triangleq F(k_{id}, m)$.
- For an input $((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$, the group-testing aggregate algorithm returns $(T_1, \ldots, T_u)$, where $T_i \triangleq \langle \boldsymbol{g}_i, (t_1, t_2, \ldots, t_n) \rangle$ for $1 \leq i \leq u$.
- For an input $(((id_1, m_1), \ldots, (id_n, m_n)), (T_1, \ldots, T_u))$, the verification algorithm returns J computed in the following way:
  1. $J \leftarrow \{(id_1, m_1), \ldots, (id_n, m_n)\}$.
  2. For $1 \leq i \leq u$, if $T_i = \langle \boldsymbol{g}_i, (F(k_1, m_1), \ldots, F(k_n, m_n)) \rangle$, then

$$J \leftarrow J \setminus \{(id_j, m_j) \mid 1 \leq j \leq n \wedge g_{i,j} = 1\} \ .$$

6.2 Unforgeability

The following theorem says that $\mathsf{GTAM}_{\mathrm{X}}$ is unforgeable if the underlying MAC function is unforgeable. It directly follows from Theorems 1 and 2, and the proof is omitted.

Theorem 4 (Unforgeability). *For* $\mathsf{GTAM}_{\mathrm{X}}$*, let* ℓ *be the number of the users. For any adversary* A *against* $\mathsf{GTAM}_{\mathrm{X}}$ *running in time at most* s*, making at most* q_{t} *queries to its tagging oracle, making at most* q_{k} *queries to its key-disclosure oracle, and making at most* q_{v} *queries to its group-testing verification oracle, there exists some adversary* B *against* F *such that*

$$\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{GTAM}_{\mathrm{X}}}(A) \leq \ell \cdot \mathrm{Adv}^{\mathrm{mac}}_{F}(B) \ ,$$

where B *runs in time at most* $s + S_F(q_{\mathrm{t}} + unq_{\mathrm{v}})$*, making at most* q_{t} *queries to its tagging oracle, and making at most* $u\,q_{\mathrm{v}}$ *queries to its verification oracle.* S_F *is time required to compute* F*.*

6.3 Identifiability

Completeness. Theorem 3 applies to $\mathsf{GTAM}_{\mathrm{X}}$, and it satisfies completeness against any d-dishonest adversary if $\boldsymbol{G}$ is d-disjunct.

Soundness. $\mathsf{GTAM}_{\mathrm{X}}$ does not satisfy soundness. Let us consider an adversary $\tilde{A}$ behaving in $\mathsf{Exp}^{\text{id-s}}_{\mathsf{GTAM}_{\mathrm{g}}}(\tilde{A})$ in the following way. $\tilde{A}$ obtains valid $(id_1, m_1, t_1), \ldots, (id_n, m_n, t_n)$ using its tagging oracle, that is, $t_j = F(k_j, m_j)$ for $1 \leq j \leq n$. Then, $\tilde{A}$ can easily compute $(\tilde{t}_1, \ldots, \tilde{t}_n)$ such that $\langle \boldsymbol{g}_{i^*}, (\tilde{t}_1, \ldots, \tilde{t}_n) \rangle = \langle \boldsymbol{g}_{i^*}, (t_1, \ldots, t_n) \rangle$ and $\boldsymbol{g}_{i^*} \boxdot (\tilde{t}_1, \ldots, \tilde{t}_n) \neq \boldsymbol{g}_{i^*} \boxdot (t_1, \ldots, t_n)$ for some i^*. Then, there exists some j^* such that $\tilde{t}_{j^*} \neq F(k_{j^*}, m_{j^*})$ and $(id_{j^*}, m_{j^*}) \notin J$.

7 GTA MAC Scheme Using Hashing for Aggregate

7.1 Scheme

Let $F : \mathcal{K} \times \mathcal{M} \to \{0,1\}^\tau$ be a MAC function. Let $H : \{0,1\}^* \to \{0,1\}^\tau$ be a cryptographic hash function. The proposed GTA MAC scheme $\mathsf{GTAM}_{\mathrm{H}}$ uses the hash function H for aggregate. The key generation and tagging algorithms of $\mathsf{GTAM}_{\mathrm{H}}$ are identical to those of $\mathsf{GTAM}_{\mathrm{X}}$. The group-testing aggregate algorithm of $\mathsf{GTAM}_{\mathrm{H}}$ is also assumed to be based on a pre-specified $u \times n$ group-testing matrix $\boldsymbol{G} = (g_{i,j})$.

- The key generation algorithm just picks up a secret key uniformly at random from $\mathcal{K}$ for each user.
- For an input (id, m), the tagging algorithm returns $t \triangleq F(k_{id}, m)$.
- For an input $((id_1, m_1, t_1), \ldots, (id_n, m_n, t_n))$, the group-testing aggregate algorithm returns $(T_1, \ldots, T_u)$, where $T_i \triangleq H(\langle\!\langle \boldsymbol{g}_i, (t_1, t_2, \cdots, t_n) \rangle\!\rangle)$. 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.
- For an input $(((id_1, m_1), \ldots, (id_n, m_n)), (T_1, \ldots, T_u))$, the verification algorithm returns J computed in the following way:
  1. $J \leftarrow \{(id_1, m_1), \ldots, (id_n, m_n)\}$.
  2. For $1 \leq i \leq u$, if $T_i = H(\langle\!\langle \boldsymbol{g}_i, (F(k_1, m_1), \ldots, F(k_n, m_n)) \rangle\!\rangle)$, then

$$J \leftarrow J \setminus \{(id_j, m_j) \,|\, 1 \leq j \leq n \wedge g_{i,j} = 1\} \ .$$

7.2 Unforgeability

The following theorem says that $\mathsf{GTAM}_{\mathrm{H}}$ is unforgeable if the underlying MAC function F is unforgeable and the underlying hash function H is a random oracle.

Theorem 5 (Unforgeability). *For the GTA MAC scheme* $\mathsf{GTAM}_{\mathrm{H}}$*, let* ℓ *be the number of the users. For any adversary* A *against* $\mathsf{GTAM}_{\mathrm{H}}$ *running in time at most* s*, making at most* q_{h} *queries to* H*, making at most* q_{t} *queries to its tagging oracle, making at most* q_{k} *queries to its key-disclosure oracle, and making at most* q_{v} *queries to its group-testing verification oracle, there exists some adversary* B *against* F *such that*

$$\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{GTAM}_{\mathrm{H}}}(A) \leq \ell \cdot \mathrm{Adv}^{\mathrm{mac}}_{F}(B) + \frac{uq_{\mathrm{v}}}{2^{\tau}} + \frac{(q_{\mathrm{h}} + uq_{\mathrm{v}})^2}{2^{\tau+1}} \ ,$$

where B *runs in time at most* $s + S_F(q_{\mathrm{t}} + unq_{\mathrm{v}})$*, making at most* $q_{\mathrm{h}} + uq_{\mathrm{v}}$ *queries to* H*, making at most* q_{t} *queries to its tagging oracle, and making at most* q_{v} *queries to its verification oracle.* S_F *is time required to compute* F*.*

Proof. Let $\mathsf{Coll}(H)$ be the event that a collision is found for H. Then,

$$\begin{aligned}
\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{GTAM}_{\mathrm{H}}}(A) &= \Pr[\mathsf{GTForge}(A)] \\
&\leq \Pr[\mathsf{GTForge}(A) \cap \overline{\mathsf{Coll}(H)}] + \Pr[\mathsf{Coll}(H)] \\
&\leq \Pr[\mathsf{GTForge}(A) \cap \overline{\mathsf{Coll}(H)}] + (q_{\mathrm{h}} + uq_{\mathrm{v}})^2/2^{\tau+1} \ .
\end{aligned}$$

Let $\mathsf{GTF}(A) \subseteq \mathsf{GTForge}(A) \cap \overline{\mathsf{Coll}(H)}$ be the event that there exists some successful group-testing verification query without a query of correct tags to H. Then,

$$\Pr[\mathsf{GTF}(A)] \leq uq_{\mathrm{v}}/2^{\tau} \ ,$$

Similarly to the proof of Theorem 1, it can be shown that there exists some adversary B against F such that

$$\Pr\Big[\big(\mathsf{GTForge}(A) \cap \overline{\mathsf{Coll}(H)}\big) \cap \overline{\mathsf{GTF}(A)}\Big] \leq \ell \cdot \mathrm{Adv}^{\mathrm{mac}}_{F}(B) \ .$$

□

7.3 Identifiability

Completeness. Theorem 3 also applies to $\mathsf{GTAM}_{\mathrm{H}}$, and it satisfies completeness against any d-dishonest adversary if $\boldsymbol{G}$ is d-disjunct.

Soundness. The following theorem says that $\mathsf{GTAM}_{\mathrm{H}}$ satisfies soundness for any d-dishonest adversary if $\boldsymbol{G}$ is d-disjunct on the assumption that H is a random oracle.

Theorem 6 (Soundness). *For the GTA MAC scheme* $\mathsf{GTAM}_{\mathrm{H}}$*, suppose that the hash function* H *is a random oracle and that the group-testing matrix* $\boldsymbol{G}$ *is* d*-disjunct. Then, for any* d*-dishonest adversary* A *making at most* q_{h} *queries to* H*,*

$$\mathrm{Adv}^{\text{id-s}}_{\mathsf{GTAM}_{\mathrm{H}}}(A) \leq \frac{(q_{\mathrm{h}} + 2u)^2}{2^{\tau+1}} \quad .$$

Proof. For $\mathtt{Exp}^{\text{id-s}}_{\mathsf{GTAM}_{\mathrm{H}}}(A)$, let $((id_1, m_1, \tilde{t}_1), \ldots, (id_n, m_n, \tilde{t}_n))$ be the output of A and $t_j = F_{k_j}(m_j)$ for $1 \leq j \leq n$. $\mathtt{Exp}^{\text{id-s}}_{\mathsf{GTAM}_{\mathrm{H}}}(A) = 1$ only if there exists some i^* such that $\boldsymbol{g}_{i^*} \boxdot (\tilde{t}_1, \ldots, \tilde{t}_n) \neq \boldsymbol{g}_{i^*} \boxdot (t_1, \ldots, t_n)$ and $H(\langle\!\langle \boldsymbol{g}, (\tilde{t}_1, \ldots, \tilde{t}_n) \rangle\!\rangle) = H(\langle\!\langle \boldsymbol{g}, (t_1, \ldots, t_n) \rangle\!\rangle)$, which implies a collision for H. H is called $(q_{\mathrm{h}} + 2u)$ times in total. □

8 Conclusion

The paper has formalized the syntax and security requirements of GTA MAC schemes and presented their generic construction. Then, it has also presented two instantiations with distinct aggregate methods. One is based on the Katz-Lindell aggregate MAC scheme and aggregates tags with addition for group testing. The other aggregates tags with hashing. The paper has analyzed the provable security of the proposed schemes. Future work includes design of an efficient algorithm to verify whether a given group-testing matrix is d-disjunct or not.

References

1. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 1109, pp. 1–15. Springer (1996)
2. Black, J., Rogaway, P.: A block-cipher mode of operation for parallelizable message authentication. In: Knudsen, L.R. (ed.) Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2332, pp. 384–397. Springer (2002), `http://dx.doi.org/10.1007/3-540-46035-7_25`
3. Du, D.Z., Hwang, F.K.: Combinatorial Group Testing and Its Applications. Series on Applied Mathematics: Volume 12, World Scientific, 2nd edn. (2000)
4. Eikemeier, O., Fischlin, M., Götzmann, J.F., Lehmann, A., Schröder, D., Schröder, P., Wagner, D.: History-free aggregate message authentication codes. In: Garay, J.A., Prisco, R.D. (eds.) SCN. Lecture Notes in Computer Science, vol. 6280, pp. 309–328. Springer (2010)
5. FIPS PUB 198-1: The keyed-hash message authentication code (HMAC) (2008)
6. Goodrich, M.T., Atallah, M.J., Tamassia, R.: Indexing information for data forensics. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3531, pp. 206–221 (2005), `https://doi.org/10.1007/11496137_15`

7. Hirose, S., Kuwakado, H.: Forward-secure sequential aggregate message authentication revisited. In: Chow, S.S.M., Liu, J.K., Hui, L.C.K., Yiu, S. (eds.) Provable Security - 8th International Conference, ProvSec 2014, Hong Kong, China, October 9-10, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8782, pp. 87–102. Springer (2014), http://dx.doi.org/10.1007/978-3-319-12475-9_7
8. Iwata, T., Kurosawa, K.: OMAC: One-key CBC MAC. In: Johansson, T. (ed.) FSE. Lecture Notes in Computer Science, vol. 2887, pp. 129–153. Springer (2003), an updated version is "Cryptology ePrint Archive: Report 2002/180" at http://eprint.iacr.org/
9. Katz, J., Lindell, A.Y.: Aggregate message authentication codes. In: Malkin, T. (ed.) CT-RSA. Lecture Notes in Computer Science, vol. 4964, pp. 155–169. Springer (2008)
10. Ma, D., Tsudik, G.: Extended abstract: Forward-secure sequential aggregate authentication. In: IEEE Symposium on Security and Privacy. pp. 86–91. IEEE Computer Society (2007), also published as IACR Cryptology ePrint Archive: Report 2007/052 at http://eprint.iacr.org/
11. Ma, D., Tsudik, G.: A new approach to secure logging. ACM Transactions on Storage 5(1), 2:1–2:21 (2009)
12. Minematsu, K.: Efficient message authentication codes with combinatorial group testing. In: Pernul, G., Ryan, P.Y.A., Weippl, E.R. (eds.) Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9326, pp. 185–202. Springer (2015), https://doi.org/10.1007/978-3-319-24174-6_10
13. NIST Special Publication 800-38B: Recommendation for block cipher modes of operation: The CMAC mode for authentication (2005)
14. Rogaway, P.: Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In: Lee, P.J. (ed.) Advances in Cryptology - ASIACRYPT 2004, 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3329, pp. 16–31. Springer (2004), https://doi.org/10.1007/978-3-540-30539-2_2