electronics

MDPI

Article

Aggregate Entity Authentication Identifying Invalid Entities with Group Testing

Shoichi Hirose [1,2,*] and Junji Shikata [2,3]

[1] Faculty of Engineering, University of Fukui, Fukui 910-8507, Japan
[2] Japan Datacom Co., Ltd., Tokyo 107-0052, Japan; shikata-junji-rb@ynu.ac.jp
[3] Graduate School of Environment and Information Sciences, Yokohama National University, Yokohama 240-8501, Japan
* Correspondence: hrs_shch@u-fukui.ac.jp

Abstract: It is common to implement challenge-response entity authentication with a MAC function. In such an entity authentication scheme, aggregate MAC is effective when a server needs to authenticate many entities. Aggregate MAC aggregates multiple tags (responses to a challenge) generated by entities into one short aggregate tag so that the entities can be authenticated simultaneously regarding only the aggregate tag. Then, all associated entities are valid if the pair of a challenge and the aggregate tag is valid. However, a drawback of this approach is that invalid entities cannot be identified when they exist. To resolve the drawback, we propose group-testing aggregate entity authentication by incorporating group testing into entity authentication using aggregate MAC. We first formalize the security requirements and present a generic construction. Then, we reduce the security of the generic construction to that of aggregate MAC and group testing. We also enhance the generic construction to instantiate a secure scheme from a simple and practical but weaker aggregate MAC scheme. Finally, we show some results on performance evaluation.

Keywords: entity authentication; message authentication; aggregate MAC; group testing

check for updates

Citation: Hirose, S.; Shikata, J. Aggregate Entity Authentication Identifying Invalid Entities with Group Testing. *Electronics* **2023**, *12*, 2479. https://doi.org/10.3390/electronics12112479

Academic Editor: Juan-Carlos Cano

Received: 28 March 2023
Revised: 16 May 2023
Accepted: 29 May 2023
Published: 31 May 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

1.1. Background

A MAC function is one of the most basic symmetric-key primitives for cryptography. Its typical application is challenge-response entity authentication, which assumes that a server and an entity share a secret key. In this scheme, the server first sends a challenge to the entity. Next, the entity computes a tag for the challenge using the MAC function with the shared secret key and returns it to the server. Finally, the server computes the tag in the same way and verifies the received tag.

Entity authentication is often crucial in identifying invalid entities to secure network applications and services. Additionally, a server may need to authenticate many devices simultaneously in an IoT network. In scenarios where an edge device plays the role of an aggregator, as shown in Figure 1, aggregate MAC [1] is suitable for efficient communication between the server and the aggregator for entity authentication. Aggregate MAC allows users to aggregate multiple tags into a tag so that the aggregate tag is as short as each of the multiple tags. In the situation shown in Figure 1, if the aggregator aggregates tags from devices and sends the aggregate tag to the server, then the server can authenticate the devices based only on the aggregate tag. If the aggregate tag is valid, then the server knows that all devices are valid. On the other hand, if the aggregate tag is invalid, then the server only knows that one or more invalid devices are included, which cannot be identified. The problem is if the server can identify invalid devices without knowing individual tags. As far as we know, it has not been addressed for entity authentication.

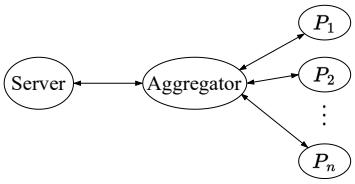

Figure 1. Targeted system configuration.

1.2. Our Contribution

We observe that group testing [2] can be employed to solve the above problem. For group testing, each item is assumed to be positive or negative. Multiple items are assumed to be able to be inspected by a test whose result is positive only if one or more positive items are included. All positive items can be identified with fewer tests than individual tests if the number of positive items is relatively small [3].

We introduce and explore group-testing aggregate authentication. It is a protocol participated in by multiple entities, an aggregator, and a server. Each entity has its own secret key shared with the server. The aggregator broadcasts a challenge from the server to the entities and collects their responses. The server then identifies the invalid entities by verifying the responses with the help of the aggregator.

We first formalizesd the scheme and its security requirements. The security requirements are impersonation resistance, completeness, and soundness. Impersonation resistance represents the notion that adversaries cannot impersonate an entity without knowing its secret key. Completeness requires that a valid response must not be judged invalid. Soundness requires that an invalid response must not be judged valid.

Furthermore, we present a generic construction combining a group-testing scheme and an aggregate MAC scheme. For each test in group testing, it aggregates the tags of entities examined by the test and verifies the aggregate tag. The aggregate tag is valid (negative) if all the involved tags are valid. Thus, invalid entities can be identified with fewer tests than by examining them individually. We also show that the generic construction satisfies impersonation resistance if the underlying aggregate MAC scheme is unforgeable, completeness if the underlying group-testing scheme satisfies completeness, and soundness if the underlying aggregate MAC scheme satisfies soundness. Furthermore, considering that the simple and practical Katz-Lindell aggregate MAC scheme [1] does not satisfy soundness, we enhance the generic construction to instantiate group-testing aggregate entity authentication satisfying soundness by using aggregate MAC not satisfying soundness.

Finally, we evaluate the performance of the proposed construction instantiated with SHA-256 [4] and HMAC [5] by software implementation.

1.3. Related Work

Katz and Lindell [1] introduced and investigated aggregate MAC. They presented a provably secure scheme for generating an aggregate tag by XOR of the associated tags. Eikemeier et al. [6] formalized sequential aggregate MAC and presented provably secure schemes. Sato et al. [7] proposed a sequential aggregate MAC scheme for aggregating tags without using the secret keys of associated users. Ishii and Tada [8] presented an aggregate MAC scheme that aggregates tags following the structure represented by a series-parallel graph.

Goodrich et al. [9] applied group testing to MAC schemes for identifying tampered data items. Along this line of research, Minematsu [10] proposed a computationally efficient scheme of *group testing MAC* based on PMAC [11]. Minematsu and Kamiya [12] proposed a method for reducing the number of tags.

Hirose and Shikata [13,14] applied group testing to aggregate MAC for identifying invalid messages from multiple senders. They used non-adaptive group testing for a generic construction. Sato and Shikata [15] presented a generic construction using adaptive group

testing. Anada and Kamibayashi [16] followed the discussion by Sato and Shikata [17] and discussed the quantum security of aggregate MAC combined with non-adaptive group-testing. Ogawa et al. [18] presented a scheme reducing the number of aggregate tags based on biorthogonal codes.

1.4. Organization

Section 2 defines notations and cryptographic primitives and describes group testing. Section 3 provides the syntax and security requirements of aggregate MAC and its concrete schemes. Section 4 formalizes group-testing aggregate entity authentication and presents its generic construction, combining group-testing and aggregate MAC. Section 5 discusses the security of the generic construction and presents its enhancement. Section 6 shows some results of the performance evaluation by software implementation. Section 7 gives a brief concluding remark.

This article is an extended and improved version of our conference paper [19]. We refine the formalization of security requirements, which are described in Section 4, based on the idea by Bellare and Rogaway [20]. Accordingly, we revise the theorems and proofs, which are given in Section 5. We also add the results on performance evaluation.

2. Preliminaries

2.1. Notation

$\{0,1\}^l$ is regarded as the set of all binary sequences of length l. Let $\{0,1\}^* := \bigcup_{l\geq 0}\{0,1\}^l$. For binary sequences x, y, their concatenation is denoted by $x\|y$.

Let $\mathcal{S}$ be a set. For $\boldsymbol{v} := (v_1,\ldots,v_n) \in \{0,1\}^n$ and $\boldsymbol{s} := (s_1,\ldots,s_n) \in \mathcal{S}^n$, let $\boldsymbol{v} \boxdot \boldsymbol{s} := (s_{j_1},\ldots,s_{j_w})$, where $1 \leq j_1 < \cdots < j_w \leq n$ and $v_j = 1$ iff $j \in \{j_1,\ldots,j_w\}$. For $\boldsymbol{u}, \boldsymbol{u}' \in \{0,1\}^n$, let $\boldsymbol{u} \vee \boldsymbol{u}'$ be their component-wise disjunction. Let $s \twoheadleftarrow \mathcal{S}$ represent that s is sampled uniformly at random from $\mathcal{S}$.

2.2. MAC Function and Pseudorandom Function

Let $f : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a keyed function with its key space $\mathcal{K}$. $f(K,\cdot)$ is often denoted by $f_K(\cdot)$.

f is called a secure MAC function or unforgeable if it is intractable to predict unknown outputs of f_K, where $K \twoheadleftarrow \mathcal{K}$. An adversary $\mathbf{A}$ is given a tagging oracle T_K^f and a verification oracle V_K^f and is allowed to make queries adaptively to them. In response to a query $X \in \mathcal{X}$, T_K^f returns $f_K(X)$. In response to a query $(X,Y) \in \mathcal{X} \times \mathcal{Y}$, V_K^f returns 1 if $f_K(X) = Y$ and 0 otherwise. $\mathbf{A}$ is not allowed to ask (X,Y) to V_K^f after asking X to T_K^f. $\mathbf{A}^{\mathsf{T}_K^f,\mathsf{V}_K^f}$ is successful iff V_K^f returns 1 in response to at least one query. The advantage of $\mathbf{A}$ against f is

$$\mathrm{Adv}_f^{\mathrm{mac}}(\mathbf{A}) := \Pr\left[\mathbf{A}^{\mathsf{T}_K^f,\mathsf{V}_K^f} \text{ is successful}\right].$$

f is called a secure pseudorandom function (PRF) if it is intractable to distinguish f_K with $K \twoheadleftarrow \mathcal{K}$ from a uniform random function $\rho : \mathcal{X} \to \mathcal{Y}$. An adversary $\mathbf{A}$ is given either f_K or ρ as an oracle and is allowed to make adaptive queries in $\mathcal{X}$. $\mathbf{A}$ outputs 0 or 1. The advantage of $\mathbf{A}$ against f is

$$\mathrm{Adv}_f^{\mathrm{prf}}(\mathbf{A}) := \left|\Pr\left[\mathbf{A}^{f_K} = 1\right] - \Pr\left[\mathbf{A}^{\rho} = 1\right]\right|,$$

where $\mathbf{A}$ is regarded as a random variable which takes values in $\{0,1\}$. It is easy to see that a secure PRF is a secure MAC function, and that a secure MAC function is not necessarily a secure PRF.

2.3. Cryptographic Hash Function

A cryptographic hash function $H : \{0,1\}^* \to \{0,1\}^\tau$ is often simply called a hash function. Among its various security requirements, our work is concerned with the random oracle model and collision resistance.

The random oracle model [21] assumes that H is an ideal function such that, for any $X \in \{0,1\}^*$, $H(X)$ is chosen uniformly at random from $\{0,1\}^\tau$. H is called a random oracle.

H is said to satisfy collision resistance if it is intractable to find a pair of distinct inputs of H mapped to the same output. The advantage of an adversary $\mathbf{A}$ against H is

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

Notice that the above definition is not theoretically precise: H should be sampled from a sufficiently large number of hash functions at random.

2.4. Group Testing

Suppose that there exists a set of items, each of which is either positive or negative. It is assumed that a test can inspect multiple items simultaneously and that the result is positive iff one or more positive items exist among them. Then, it may be possible to identify positive items with fewer tests than by inspecting all the items individually.

A group-testing algorithm can be described as a sequence of sets of tests. Suppose that there are n items. Then, each test can be denoted by a vector in $\{0,1\}^n$ such that the j-th element equals 1 iff the test examines the j-th item. Let $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_u \subseteq \{0,1\}^n$ be a sequence of sets of tests, where u is the number of its stages. The sets of tests are conducted in this order, and the order of the tests in each stage is arbitrary. A group-testing algorithm is called non-adaptive if all the tests are determined beforehand. Thus, it has only a single stage. A group-testing algorithm is called adaptive if the tests in the next stage are determined after the tests in the current stage.

Let $\mathcal{G} := \mathcal{G}_1 \cup \mathcal{G}_2 \cup \cdots \cup \mathcal{G}_u$. It is reasonable to assume that each test examines at least one item and that the whole set of tests examines all items. Namely, $0^n \notin \mathcal{G}$ and $\bigvee_{\boldsymbol{g} \in \mathcal{G}} \boldsymbol{g} = 1^n$. The group-testing algorithm extracts candidates of positive items in the following way. For $1 \leq j \leq n$, let id_j denote the j-th item. For $1 \leq i \leq u$, let $\mathcal{G}_i := \{\boldsymbol{g}_{i,1}, \boldsymbol{g}_{i,2}, \ldots, \boldsymbol{g}_{i,|\mathcal{G}_i|}\}$. Let $\mathcal{O}(\boldsymbol{g}) := \{id_j \mid 1 \leq j \leq n \text{ and } g_j = 1\}$, where $\boldsymbol{g} := (g_1, g_2, \ldots, g_n) \in \{0,1\}^n$.

1. $\mathcal{J}_0 \leftarrow \{id_1, id_2, \ldots, id_n\}$.
2. For $1 \leq i \leq u$, do the followings: (a) $\mathcal{J}_i \leftarrow \mathcal{J}_{i-1}$; (b) For $1 \leq l \leq |\mathcal{G}_i|$, if the result of $\boldsymbol{g}_{i,l}$ is negative, then $\mathcal{J}_i \leftarrow \mathcal{J}_i \setminus \mathcal{O}(\boldsymbol{g}_{i,l})$.
3. Output $\mathcal{J}_u$.

We call the group-testing algorithm complete if $\mathcal{J}_u$ does not include any negative elements. We call it sound if $\mathcal{J}_u$ includes all the positive elements. It is sound if the results of the tests are always correct. On the other hand, it may not be complete in general.

For non-adaptive group testing, let us see the matrix $\boldsymbol{G}$ whose rows are the vectors in a set $\mathcal{G}$ of tests, which is called a group-testing matrix. We call $\boldsymbol{G}$ d-disjunct if the component-wise disjunction of any d columns in $\boldsymbol{G}$ does not equal the component-wise disjunction of itself and any other single column. Suppose that non-adaptive group testing is represented by a d-disjunct matrix. Then, it is complete if there are at most d positive items. Specifically, all the positive items are identified.

Suppose that there are at most d positive items. For non-adaptive group testing, it is known that there exists a complete algorithm with $O(d^2 \log n)$ tests [3,22–24]. In addition, a non-asymptotic lower bound was conjectured as $\min\{(d+1)^2, n\}$ [25] while it is true for $d \leq 5$, and actually derived as $\min\{(d+2)(d+1)/2, n\}$ [26] and $\min\{d^2(15+\sqrt{33})/24, n\}$ [27]. For adaptive group testing, it is known that there exists a complete algorithm with $O(d \log(n/d))$ tests [3,28,29]. A tight lower bound is shown as $d \log(n/d) + o(d \log(n/d))$ [3].

3. Aggregate MAC

3.1. Syntax

A tuple of algorithms $\mathsf{AM} := (\mathsf{KG}, \mathsf{Tag}, \mathsf{Agg}, \mathsf{Ver})$ formalizes an aggregate MAC scheme. It is associated with an ID space $\mathcal{I}$, a key space $\mathcal{K}$, a message space $\mathcal{M}$, a tag space $\mathcal{T}$, and an aggregate-tag space $\mathcal{A}$.

- KG is a key-generation algorithm such that $k \leftarrow \mathsf{KG}(1^\kappa)$, where κ is a security parameter and $k \in \mathcal{K}$. Each entity is assigned a secret key independently generated by KG.
- Tag is a tagging algorithm such that $t \leftarrow \mathsf{Tag}(k, m)$, where $(k, m) \in \mathcal{K} \times \mathcal{M}$ and $t \in \mathcal{T}$.
- Agg is an aggregate algorithm such that $T \leftarrow \mathsf{Agg}((id_1, m_1, t_1), \ldots, (id_p, m_p, t_p))$, where $(id_j, m_j, t_j) \in \mathcal{I} \times \mathcal{M} \times \mathcal{T}$ for $1 \leq j \leq p$ and $T \in \mathcal{A}$. (id_j, m_j)'s are required to be distinct from each other. It is often the case that T depends only on $t_1, t_2, \ldots, t_p$.
- Ver is a verification algorithm such that $d \leftarrow \mathsf{Ver}(((id_1, k_1), \ldots, (id_p, k_p)), ((id_1, m_1), \ldots, (id_p, m_p)), T)$, where $(id_j, k_j) \in \mathcal{I} \times \mathcal{K}$ and $(id_j, m_j) \in \mathcal{I} \times \mathcal{M}$ for $1 \leq j \leq p$, $T \in \mathcal{T}$ if $p = 1$ and $T \in \mathcal{A}$ otherwise, and $d \in \{0, 1\}$. (id_j, m_j)'s are required to be distinct from each other. With respect to $((id_1, k_1), \ldots, (id_p, k_p))$, the pair $((id_1, m_1), \ldots, (id_p, m_p))$ and T are valid if $d = 1$ and invalid otherwise.

AM satisfies correctness. For $(id_1, k_1), \ldots, (id_p, k_p)$ and $(id_1, m_1), \ldots, (id_p, m_p)$, let $t_j \leftarrow \mathsf{Tag}(k_j, m_j)$ for $1 \leq j \leq p$ and $T \leftarrow \mathsf{Agg}((id_1, m_1, t_1), \ldots, (id_p, m_p, t_p))$. Then, it holds that $\mathsf{Ver}(((id_1, k_1), \ldots, (id_p, k_p)), ((id_1, m_1), \ldots, (id_p, m_p)), T) = 1$. In particular, for $p = 1$, if $t \leftarrow \mathsf{Tag}(k, m)$, then $\mathsf{Ver}((id, k), (id, m), t) = 1$.

3.2. Security Requirement

Unforgeability and soundness are formalized as security requirements for aggregate MAC. Soundness is required for applying group testing to aggregate MAC [14].

3.2.1. Unforgeability

We introduce a game $\mathscr{G}^{\mathrm{uf}}_{\mathsf{AM},\mathbf{A}}$ to define unforgeability, where $\mathbf{A}$ is an adversary allowed to make queries adaptively to the oracles $\mathcal{TG}$, $\mathcal{KD}$, and $\mathcal{VR}$.

- $\mathcal{TG}$ is called a tagging oracle. It returns $t \leftarrow \mathsf{Tag}(k_{id}, m)$ in response to a query (id, m), where k_{id} is the key of the entity id.
- $\mathcal{KD}$ is called a key-disclosure oracle. It accepts a query id and returns k_{id}.
- $\mathcal{VR}$ is called a verification oracle. It accepts a query $(((id_1, m_1), \ldots, (id_p, m_p)), T)$ and returns $d \leftarrow \mathsf{Ver}(((id_1, k_1), \ldots, (id_p, k_p)), ((id_1, m_1), \ldots, (id_p, m_p)), T)$.

For a query $(((id_1, m_1), \ldots, (id_p, m_p)), T)$ made by $\mathbf{A}$ to $\mathcal{VR}$, we call (id_j, m_j) a fresh pair if $\mathbf{A}$ does not ask it to $\mathcal{TG}$ and does not ask id_j to $\mathcal{KD}$ prior to the query. $\mathcal{VR}$ does not accept a query with no fresh pair. $\mathscr{G}^{\mathrm{uf}}_{\mathsf{AM},\mathbf{A}}$ outputs 1 iff $\mathbf{A}$ gets 1 from $\mathcal{VR}$ for at least one query. The advantage of $\mathbf{A}$ against AM for unforgeability is defined by

$$\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{AM}}(\mathbf{A}) := \Pr[\mathscr{G}^{\mathrm{uf}}_{\mathsf{AM},\mathbf{A}} = 1].$$

It is informally stated that AM is unforgeable or satisfies unforgeability if, for any efficient $\mathbf{A}$, $\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{AM}}(\mathbf{A})$ is negligible.

3.2.2. Soundness

To define soundness, we specify a game $\mathscr{G}^{\mathrm{snd}}_{\mathsf{AM},\mathbf{A}}$, where $\mathbf{A}$ is an adversary allowed to make queries adaptively to the aggregate-then-verify oracle $\mathcal{AVR}$ in addition to the oracles $\mathcal{TG}$, $\mathcal{KD}$, and $\mathcal{VR}$. $\mathcal{AVR}$ accepts a query $((id_1, m_1, t_1), \ldots, (id_p, m_p, t_p))$ and computes

1. $d_j \leftarrow \mathsf{Ver}((id_j, k_j), (id_j, m_j), t_j)$ for $1 \leq j \leq p$,
2. $T \leftarrow \mathsf{Agg}((id_1, m_1, t_1), \ldots, (id_p, m_p, t_p))$, and
3. $D \leftarrow \mathsf{Ver}(((id_1, k_1), \ldots, (id_p, k_p)), ((id_1, m_1), \ldots, (id_p, m_p)), T)$.

Then, it returns $D \wedge (\overline{d_1} \vee \overline{d_2} \vee \cdots \vee \overline{d_p})$. $\mathcal{G}^{\mathrm{snd}}_{\mathsf{AM},\mathbf{A}}$ outputs 1 iff $\mathbf{A}$ gets 1 from $\mathcal{AVR}$ for at least one query. The advantage of $\mathbf{A}$ against AM for soundness is defined by

$$\mathrm{Adv}^{\mathrm{snd}}_{\mathsf{AM}}(\mathbf{A}) := \Pr[\mathcal{G}^{\mathrm{snd}}_{\mathsf{AM},\mathbf{A}} = 1].$$

It is informally stated that AM is sound or satisfies soundness if, for any efficient $\mathbf{A}$, $\mathrm{Adv}^{\mathrm{snd}}_{\mathsf{AM}}(\mathbf{A})$ is negligible.

3.3. Aggregate MAC Scheme by Katz and Lindell

Let $F : \mathcal{K} \times \mathcal{M} \to \{0,1\}^\tau$ be a MAC function. The Katz-Lindell aggregate MAC scheme [1] using F is specified as follows:

- Each entity $id \in \mathcal{I}$ is given a secret key $k_{id} \twoheadleftarrow \mathcal{K}$.
- The tagging algorithm returns a tag $t \leftarrow F_k(m)$ in response to $(k, m) \in \mathcal{K} \times \mathcal{M}$.
- The aggregate algorithm returns an aggregate tag $T \leftarrow \bigoplus_{1 \le i \le p} t_i$ in response to $(id_1, m_1, t_1), \ldots, (id_p, m_p, t_p) \in \mathcal{I} \times \mathcal{M} \times \mathcal{T}$.
- Taking $(id_1, k_1), \ldots, (id_p, k_p)$ and $((id_1, m_1), \ldots, (id_p, m_p), T)$ as input, the verification algorithm outputs 1 iff $\bigoplus_{1 \le i \le p} F_{k_i}(m_i) = T$.

Let AM_X denote the Katz-Lindell aggregate MAC scheme. AM_X is shown to be unforgeable for any efficient adversary asking the verification oracle a single query [1]. It is also shown to be unforgeable even for any efficient adversary asking the verification oracle multiple queries:

Proposition 1 ([14]). *Let* $\mathbf{A}$ *be any adversary against* AM_X *with* ℓ *users. Suppose that* $\mathbf{A}$ *asks the tagging oracle* q_t *queries and the verification oracle* q_v *queries. Suppose that each verification query by* $\mathbf{A}$ *consists of at most* p *pairs of ID and message. Then, there exists some adversary* $\dot{\mathbf{A}}$ *satisfying*

$$\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{AM}_\mathrm{X}}(\mathbf{A}) \le \ell q_\mathrm{v} \cdot \mathrm{Adv}^{\mathrm{mac}}_F(\dot{\mathbf{A}}).$$

$\dot{\mathbf{A}}$ *asks the tagging oracle at most* $(q_\mathrm{t} + p)$ *queries and the verification oracle at most one query.* $\dot{\mathbf{A}}$*'s running time is at most about that of* $\mathcal{G}^{\mathrm{uf}}_{\mathsf{AM}_\mathrm{X},\mathbf{A}}$.

It is easy to see that AM_X is not sound. Let $\tilde{\mathbf{A}}$ be an adversary working as follows. $\tilde{\mathbf{A}}$ first asks (id_1, m_1) and (id_2, m_2) to the tagging oracle and gets $t_1 = F_{k_1}(m_1)$ and $t_2 = F_{k_2}(m_2)$. Then, $\tilde{\mathbf{A}}$ gets 1 from the aggregate-then-verify oracle by asking $((id_1, m_1, \tilde{t}_1), (id_2, m_2, \tilde{t}_2))$ such that $(\tilde{t}_1, \tilde{t}_2) \neq (t_1, t_2)$ and $\tilde{t}_1 \oplus \tilde{t}_2 = t_1 \oplus t_2$.

3.4. Aggregate MAC Scheme Using Hashing

We refer to the aggregate MAC scheme using a hash function $H : \{0,1\}^* \to \{0,1\}^\tau$ to aggregate tags [14] as AM_H. AM_H is specified as follows:

- The key generation and tagging algorithms are identical to those of AM_X.
- For $(id_1, m_1, t_1), \ldots, (id_p, m_p, t_p)$, the aggregate algorithm returns $T \leftarrow H(t_1 \| \cdots \| t_p)$. For the uniqueness of the aggregate tag T, $(id_1, m_1, t_1), \ldots, (id_p, m_p, t_p)$ are assumed to be ordered in a lexicographic order.
- Taking $(id_1, k_1), \ldots, (id_p, k_p)$ and $((id_1, m_1), \ldots, (id_p, m_p), T)$ as input, the verification algorithm outputs 1 if $H(F_{k_1}(m_1) \| \cdots \| F_{k_p}(m_p)) = T$ and 0 otherwise.

AM_H is shown to be unforgeable if F is unforgeable and H is a random oracle [14]:

Proposition 2. *Let* $\mathbf{A}$ *be any adversary against* AM_H *with* ℓ *users. Suppose that* $\mathbf{A}$ *asks the random oracle* H q_h *queries, the tagging oracle* q_t *queries, and the verification oracle* q_v *queries. Suppose that each verification query by* $\mathbf{A}$ *consists of at most* p *pairs of ID and message. Then, there exists some adversary* $\dot{\mathbf{A}}$ *satisfying*

$$\mathrm{Adv}^{\mathrm{uf}}_{\mathsf{AM}_\mathrm{H}}(\mathbf{A}) \le \ell q_\mathrm{v} \cdot \mathrm{Adv}^{\mathrm{mac}}_F(\dot{\mathbf{A}}) + q_\mathrm{v}/2^\tau.$$

$\dot{\mathbf{A}}$ *asks the random oracle at most* $(q_{\mathrm{h}} + q_{\mathrm{v}})$ *queries, the tagging oracle at most* $(q_{\mathrm{t}} + p)$ *queries, and the verification oracle at most one query.* $\dot{\mathbf{A}}$*'s running time is at most about that of* $\mathscr{G}^{\mathrm{uf}}_{\mathsf{AM}_{\mathrm{H}},\mathbf{A}}$.

The soundness of AM_{H} is reduced to the collision resistance of H [14]:

Proposition 3. *For any adversary* $\mathbf{A}$ *against* AM_{H} *concerning soundness, there exists some adversary* $\dot{\mathbf{A}}$ *satisfying*

$$\mathrm{Adv}^{\mathrm{snd}}_{\mathsf{AM}_{\mathrm{H}}}(\mathbf{A}) \leq \mathrm{Adv}^{\mathrm{col}}_{H}(\dot{\mathbf{A}}).$$

The running time of $\dot{\mathbf{A}}$ *is at most about that of* $\mathscr{G}^{\mathrm{snd}}_{\mathsf{AM}_{\mathrm{H}},\mathbf{A}}$.

4. Group-Testing Aggregate Entity Authentication

4.1. Scheme

We present a group-testing aggregate entity authentication scheme. It is a challenge-response protocol between a server and a set of entities, and they communicate through an aggregator (Figure 1). It consists of a group-testing algorithm GT and an aggregate MAC scheme $\mathsf{AM} := (\mathsf{KG}, \mathsf{Tag}, \mathsf{Agg}, \mathsf{Ver})$ and is denoted by $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}]$.

Let $\mathcal{P} := \{P_1, P_2, \ldots, P_n\}$ denote the set of entities. Each P_j has an ID id_j and shares a secret key $k_j \leftarrow \mathsf{KG}(1^\kappa)$ with the server. $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}]$ proceeds as follows:

Step 1: The server sends a challenge $c \twoheadleftarrow \{0,1\}^\nu$ to the aggregator, which broadcasts it to the entities.

Step 2: In response to c, each entity P_j returns (id_j, t_j) to the aggregator, where $t_j \leftarrow \mathsf{Tag}(k_j, c)$.

Step 3: The aggregator sends $(id_1, id_2, \ldots, id_n)$ to the server.

Step 4: With the help of the aggregator, the server identifies the valid entities using GT, Agg, and Ver in the following way:

1. $\mathcal{J}_0 \leftarrow \{id_1, id_2, \ldots, id_n\}$.
2. Let u be the number of stages of GT. For $1 \leq i \leq u$,
   (a) According to GT, both the server and the aggregator determine the set of tests $\mathcal{G}_i := \{\boldsymbol{g}_{i,1}, \ldots, \boldsymbol{g}_{i,|\mathcal{G}_i|}\}$.
   (b) The aggregator computes $T_{i,l} \leftarrow \mathsf{Agg}(\boldsymbol{g}_{i,l} \boxdot ((id_1, c, t_1), \ldots, (id_n, c, t_n)))$ for $1 \leq l \leq |\mathcal{G}_i|$ and sends $(T_{i,1}, T_{i,2}, \ldots, T_{i,|\mathcal{G}_i|})$ to the server.
   (c) The server first sets $\mathcal{J}_i \leftarrow \mathcal{J}_{i-1}$. Then, for $1 \leq l \leq |\mathcal{G}_i|$, it computes
   $$D_{i,l} \leftarrow \mathsf{Ver}(\boldsymbol{g}_{i,l} \boxdot ((id_1, k_1), \ldots, (id_n, k_n)), \boldsymbol{g}_{i,l} \boxdot ((id_1, c), \ldots, (id_n, c)), T_{i,l})$$
   and $\mathcal{J}_i \leftarrow \mathcal{J}_i \setminus \mathcal{O}(\boldsymbol{g}_{i,l})$ if $D_{i,l} = 1$. Finally, it sends $(D_{i,1}, D_{i,2}, \ldots, D_{i,|\mathcal{G}_i|})$ to the aggregator.
3. Output $\mathcal{J}_u$.

The communication among the server, the aggregator, and the entities in $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}]$ is depicted in Figure 2.

For the description above, Step 3 can be merged with the first move of 2(b) in Step 4 if the server knows the number of entities to be authenticated in advance. If GT is non-adaptive, then $u = 1$, and both the server and the aggregator know all the tests in advance. In addition, the server does not have to send the results of the tests to the aggregator in Step 4(c). If GT is adaptive, then the results of $\mathcal{G}_1, \ldots, \mathcal{G}_j$ determine $\mathcal{G}_{j+1}$. Since the server sends the results of the current tests to the aggregator, the aggregator can also determine the new set of tests.

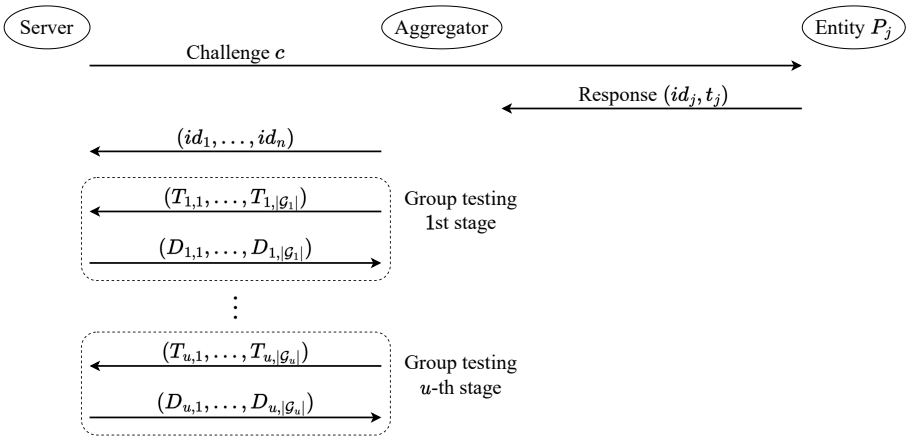

Figure 2. The communication among the server, the aggregator, and the entities in $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}]$.

4.2. Security Requirement

The security requirements of $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}]$ are impersonation resistance, completeness, and soundness.

4.2.1. Impersonation Resistance

We introduce a game $\mathcal{G}^{\mathrm{im}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}],\mathbf{A}}$ to formalize impersonation resistance. In this game, the adversary $\mathbf{A}$ is supplied with oracles $\{S^{(i)} \mid i \in \mathbb{N}\}$ working as the server. For the i-th run of $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}]$, $\mathbf{A}$ triggers $S^{(i)}$, which starts the protocol by returning a challenge $c^{(i)}$ to $\mathbf{A}$. $\mathbf{A}$ is also supplied with oracles $\{P_j^{(i)} \mid i \in \mathbb{N} \text{ and } 1 \leq j \leq n\}$ working as entities.

The i-th run of $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}]$ proceeds with the communication between $S^{(i)}$ and $\mathbf{A}$. Multiple runs may proceed concurrently in general. Each $P_j^{(i)}$ accepts two kinds of queries. For a tagging query $(\mathtt{tag}, c)$, $P_j^{(i)}$ returns $\mathsf{Tag}(k_j, c)$. For a corrupt query `corrupt`, it returns k_j. Once $\mathbf{A}$ gets $c^{(i)}$, $\mathbf{A}$ is allowed to ask it only to $P_j^{(i)}$. At the end of the run, $S^{(i)}$ outputs the set $\mathcal{J}^{(i)}$ of IDs of invalid entities. $S^{(i)}$ may abort the run if $\mathbf{A}$ does not follow the protocol.

$\mathcal{G}^{\mathrm{im}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}],\mathbf{A}}$ outputs 1 iff there exist some i^* and j^* such that $id_{j^*} \notin \mathcal{J}^{(i^*)}$, $\mathbf{A}$ does not ask $c^{(i^*)}$ to $P_{j^*}^{(i^*)}$, and $\mathbf{A}$ does not ask `corrupt` to $P_{j^*}^{(i)}$ for any i. The advantage of $\mathbf{A}$ against $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}]$ for impersonation resistance is

$$\mathrm{Adv}^{\mathrm{im}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}]}(\mathbf{A}) := \Pr[\mathcal{G}^{\mathrm{im}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}],\mathbf{A}} = 1].$$

4.2.2. Completeness and Soundness

Completeness and soundness are security requirements for the identifiability of (in)valid responses to a challenge. We introduce games $\mathcal{G}^{\mathrm{cmp}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}],\mathbf{A}}$ and $\mathcal{G}^{\mathrm{snd}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}],\mathbf{A}}$. In both games, the adversary $\mathbf{A}$ is not allowed to corrupt the server and the aggregator, and the communication channel between them is authenticated. Notice that, if $\mathbf{A}$ is allowed to tamper aggregate tags, then any valid response by an entity can be judged invalid by the server.

In both of the games, the adversary $\mathbf{A}$ is supplied with oracles $\{SA^{(i)} \mid i \in \mathbb{N}\}$ playing the roles of the server and the aggregator. $\mathbf{A}$ is also supplied with oracles $\{P_j^{(i)} \mid i \in \mathbb{N}, 1 \leq j \leq n\}$ working as entities, which are specified in Section 4.2.1. For the i-th run of $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}]$, $\mathbf{A}$ triggers $SA^{(i)}$, which starts the protocol by returning a challenge $c^{(i)}$ to $\mathbf{A}$. Once $\mathbf{A}$ gets $c^{(i)}$, $\mathbf{A}$ is allowed to ask it only to $P_j^{(i)}$. In response to $c^{(i)}$, $\mathbf{A}$ returns $(id_1, t_1^{(i)}), (id_2, t_2^{(i)}), \ldots, (id_n, t_n^{(i)})$ to $SA^{(i)}$. $SA^{(i)}$ runs the protocol step by step. Each step is triggered by $\mathbf{A}$. $\mathbf{A}$ can also see the messages communicated during the protocol. At the end of the run, $SA^{(i)}$ outputs the set $\mathcal{J}^{(i)}$ of IDs of invalid entities. Multiple runs may proceed concurrently in general.

$\mathcal{G}^{\mathrm{cmp}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}],\mathbf{A}}$ outputs 1 iff there exists some i^* such that

$$\mathcal{J}^{(i^*)} \cap \{id_j \mid t_j^{(i^*)} = \mathsf{Tag}(k_j, c^{(i^*)})\} \neq \emptyset.$$

$\mathcal{G}^{\mathrm{snd}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}],\mathbf{A}}$ outputs 1 iff there exists some i^* such that

$$\{id_j \mid t_j^{(i^*)} \neq \mathsf{Tag}(k_j, c^{(i^*)})\} \setminus \mathcal{J}^{(i^*)} \neq \emptyset.$$

The advantage of $\mathbf{A}$ for completeness and soundness of $\mathsf{EA}[\mathsf{GT},\mathsf{AM}]$ is

$$\mathrm{Adv}^{\mathrm{cmp}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}]}(\mathbf{A}) := \Pr[\mathcal{G}^{\mathrm{cmp}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}],\mathbf{A}} = 1], \text{ and}$$
$$\mathrm{Adv}^{\mathrm{snd}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}]}(\mathbf{A}) := \Pr[\mathcal{G}^{\mathrm{snd}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}],\mathbf{A}} = 1],$$

respectively.

Remark 1. *The unforgeability of the tagging algorithm is irrelevant to soundness. This is because, for soundness,* $\mathbf{A}$ *is allowed to ask* $(\mathtt{tag}, c^{(i)})$ *and* `corrupt` *to* $P_j^{(i)}$ *for any i and j. If the tagging algorithm is unforgeable and* $\mathbf{A}$ *is not allowed to ask them to* $P_j^{(i)}$*, then it cannot return a valid tag to* $c^{(i)}$*. Thus, impersonation resistance can be regarded as weak soundness in that*

$$\{id_j \mid t_j^{(i)} \neq \mathsf{Tag}(k_j, c^{(i)}) \text{ and } \mathbf{A} \text{ neither gets } \mathsf{Tag}(k_j, c^{(i)}) \text{ nor corrupts } P_j\} \setminus \mathcal{J}^{(i)} = \emptyset.$$

All in all, impersonation resistance is sufficient to identify invalid entities. Soundness is required to achieve the same function as individual verification of each response, that is, to identify invalid responses.

5. Discussion on Security

5.1. Impersonation Resistance

The impersonation resistance of $\mathsf{EA}[\mathsf{GT},\mathsf{AM}]$ is reduced to the unforgeability of AM:

Theorem 1. *For any adversary* $\mathbf{A}$ *against* $\mathsf{EA}[\mathsf{GT},\mathsf{AM}]$ *for impersonation resistance, triggering at most* q_{r} *runs of* $\mathsf{EA}[\mathsf{GT},\mathsf{AM}]$ *and making at most* q_{t} *tagging queries and* q_{c} *corrupt queries, there exists some adversary* $\dot{\mathbf{A}}$ *satisfying*

$$\mathrm{Adv}^{\mathrm{im}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}]}(\mathbf{A}) \leq \mathrm{Adv}^{\mathrm{uf}}_{\mathsf{AM}}(\dot{\mathbf{A}}) + q_{\mathrm{r}}(q_{\mathrm{r}} + 2q_{\mathrm{t}})/2^{\nu+1}.$$

The number of queries made by $\dot{\mathbf{A}}$ *to* $\mathcal{TG}$ *is at most* q_{t}*. The number of queries made by* $\dot{\mathbf{A}}$ *to* $\mathcal{KD}$ *is at most* q_{c}*. The number of queries made by* $\dot{\mathbf{A}}$ *to* $\mathcal{VR}$ *is at most the total number of tests completed by* $S^{(i)}$*'s in* $\mathcal{G}^{\mathrm{im}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}],\mathbf{A}}$*. The running time of* $\dot{\mathbf{A}}$ *is at most about that of* $\mathcal{G}^{\mathrm{im}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}],\mathbf{A}}$*.*

Proof. In $\mathcal{G}^{\mathrm{uf}}_{\mathsf{AM},\dot{\mathbf{A}}}$, $\dot{\mathbf{A}}$ runs $\mathcal{G}^{\mathrm{im}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}],\mathbf{A}}$. If $\mathbf{A}$ makes a tagging query $(\mathtt{tag}, c)$ to $P_j^{(i)}$, then $\dot{\mathbf{A}}$ asks (id_j, c) to $\mathcal{TG}$ and gets $t_j \leftarrow \mathsf{Tag}(k_j, c)$, which is returned to $\mathbf{A}$. If $\mathbf{A}$ makes a corrupt query to $P_j^{(i)}$, then $\dot{\mathbf{A}}$ asks id_j to $\mathcal{KD}$ and gets k_j, which is returned to $\mathbf{A}$. $\dot{\mathbf{A}}$ simulates $S^{(1)}, S^{(2)}, \ldots, S^{(q_{\mathrm{r}})}$ by making use of $\mathcal{VR}$.

Suppose that $\mathcal{G}^{\mathrm{im}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}],\mathbf{A}}$ outputs 1. Then, there are two cases:

1. There exists some i^* such that the challenge $c^{(i^*)}$ of the i^*-th run of $\mathsf{EA}[\mathsf{GT},\mathsf{AM}]$ collides with some previous challenge $c^{(i')}$ $(i' < i^*)$ or c in a previous tagging query.
2. There exists some i^* and j^* such that, for some test $\boldsymbol{g} := (g_1, \ldots, g_n) \in \{0,1\}^n$ with $g_{j^*} = 1$ during the i^*-th run of $\mathsf{EA}[\mathsf{GT},\mathsf{AM}]$, $\mathsf{Ver}(\boldsymbol{g} \boxdot ((id_1, k_1), \ldots, (id_n, k_n)), \boldsymbol{g} \boxdot$

$((id_1, c^{(i^*)}), \ldots, (id_n, c^{(i^*)})), T^*) = 1$, and $\mathbf{A}$ does not ask $(\mathtt{tag}, c^{(i^*)})$ to $P_{j^*}^{(i^*)}$ and `corrupt` to any $P_{j^*}^{(i)}$.

For the first case, notice that $\mathbf{A}$ triggers at most q_r runs of $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}]$ and makes at most q_t tagging queries. Thus, the probability of the first case is at most

$$(q_\mathrm{r}(q_\mathrm{r} - 1)/2)/2^\nu + q_\mathrm{r} q_\mathrm{t}/2^\nu \leq q_\mathrm{r}(q_\mathrm{r} + 2q_\mathrm{t})/2^{\nu+1}.$$

For the second case, $\dot{\mathbf{A}}$ gets 1 from $\mathcal{VR}$ in response to the query $(\boldsymbol{g} \boxdot ((id_1, c^{(i^*)}), \ldots, (id_n, c^{(i^*)})), T^*)$, and $(id_{j^*}, c^{(i^*)})$ is a fresh pair. Thus, $\mathscr{G}^{\mathrm{uf}}_{\mathsf{AM},\mathbf{A}}$ outputs 1. □

5.2. Completeness and Soundness

It is easy to see that the completeness of $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}]$ is reduced to the completeness of GT since AM satisfies correctness:

Theorem 2. *If* GT *satisfies completeness, then, for any adversary* $\mathbf{A}$ *against* $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}]$,

$$\mathrm{Adv}^{\mathrm{cmp}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}]}(\mathbf{A}) = 0.$$

Proof. Since GT satisfies completeness, for any valid tag, there exists some test such that it examines the tag and all the other tags it examines are valid. Since AM satisfies correctness, any aggregate tag generated only from valid tags is judged valid. □

The soundness of $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}]$ is reduced to the soundness of AM:

Theorem 3. *For any adversary* $\mathbf{A}$ *against* $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}]$ *for soundness, triggering at most* q_r *runs of* $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}]$ *and making at most* q_t *tagging queries and* q_c *corrupt queries, there exists some adversary* $\dot{\mathbf{A}}$ *satisfying*

$$\mathrm{Adv}^{\mathrm{snd}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}]}(\mathbf{A}) \leq \mathrm{Adv}^{\mathrm{snd}}_{\mathsf{AM}}(\dot{\mathbf{A}}).$$

The number of queries made by $\dot{\mathbf{A}}$ *to* $\mathcal{TG}$ *is at most* q_t. *The number of queries made by* $\dot{\mathbf{A}}$ *to* $\mathcal{KD}$ *is at most* q_c. *The number of queries made by* $\dot{\mathbf{A}}$ *to* $\mathcal{VR}$ *is at most the total number of tests during the runs of* $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}]$. *The number of queries made by* $\dot{\mathbf{A}}$ *to* $\mathcal{AVR}$ *is also at most the total number of tests during the runs of* $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}]$. *The running time of* $\dot{\mathbf{A}}$ *is at most about that of* $\mathscr{G}^{\mathrm{snd}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}],\mathbf{A}}$.

Proof. In $\mathscr{G}^{\mathrm{snd}}_{\mathsf{AM},\dot{\mathbf{A}}}$, $\dot{\mathbf{A}}$ runs $\mathscr{G}^{\mathrm{snd}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}],\mathbf{A}}$ in the similar way described in the proof of Theorem 1. $\dot{\mathbf{A}}$ simulates $SA^{(1)}, SA^{(2)}, \ldots, SA^{(q_\mathrm{r})}$ by making use of $\mathcal{VR}$. Suppose that $\mathbf{A}$ returns $(id_1, t_1^{(i)}), (id_2, t_2^{(i)}), \ldots, (id_n, t_n^{(i)})$ to $SA^{(i)}$ in response to the challenge $c^{(i)}$. Then, for each test $\boldsymbol{g} := (g_1, \ldots, g_n) \in \{0,1\}^n$ during the i-th run of $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}]$, $\dot{\mathbf{A}}$ also makes a query $\boldsymbol{g} \boxdot ((id_1, c^{(i)}, t_1^{(i)}), \ldots, (id_n, c^{(i)}), t_n^{(i)})$ to $\mathcal{AVR}$.

Suppose that $\mathscr{G}^{\mathrm{snd}}_{\mathsf{EA}[\mathsf{GT},\mathsf{AM}],\mathbf{A}}$ outputs 1 in $\mathscr{G}^{\mathrm{snd}}_{\mathsf{AM},\dot{\mathbf{A}}}$. Then, there exists some i^* and j^* such that $\{id_{j^*} \mid t_{j^*}^{(i^*)} \neq \mathsf{Tag}(k_{j^*}, c^{(i^*)})\} \setminus \mathcal{J}^{(i^*)} \neq \emptyset$. Thus, during the i^*-th run of $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}]$, there exists some test $\boldsymbol{g}^* := (g_1^*, \ldots, g_n^*)$ with $g_{j^*}^* = 1$ such that $\mathsf{Ver}(\boldsymbol{g}^* \boxdot ((id_1, k_1), \ldots, (id_n, k_n)), \boldsymbol{g}^* \boxdot ((id_1, c^{(i^*)}), \ldots, (id_n, c^{(i^*)})), T^*) = 1$ and $t_{j^*}^{(i^*)} \neq \mathsf{Tag}(k_{j^*}, c^{(i^*)})$, where $T^* := \mathsf{Agg}(\boldsymbol{g}^* \boxdot ((id_1, c^{(i^*)}, t_1^{(i^*)}), \ldots, (id_n, c^{(i^*)}, t_n^{(i^*)})))$. Thus, $\dot{\mathbf{A}}$ gets 1 from $\mathcal{AVR}$ in response to $\boldsymbol{g}^* \boxdot ((id_1, c^{(i^*)}, t_1^{(i^*)}), \ldots, (id_n, c^{(i^*)}, t_n^{(i^*)}))$. □

5.3. Enhancing the Generic Construction

From the results so far, we confirm that $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}_\mathrm{X}]$ and $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}_\mathrm{H}]$ satisfy impersonation resistance and satisfy completeness if GT satisfies completeness. On the other hand, $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}_\mathrm{X}]$ does not satisfy soundness, while $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}_\mathrm{H}]$ satisfies soundness.

We enhance the proposed scheme and present $\mathsf{EEA}[\mathsf{GT},\mathsf{AM}]$, which achieves soundness even with AM_χ.

$\mathsf{EEA}[\mathsf{GT},\mathsf{AM}]$ is equipped with a PRF $R : \mathcal{R} \times \mathcal{I} \times \{0,1\}^{\nu+\tau} \to \{0,1\}^\tau$, where $\mathcal{R}$ is its key space. A shared secret key $r \in \mathcal{R}$ is given to the server and the aggregator. Notice that, for soundness, the communication channel between the server and the aggregator is assumed to be authenticated. Thus, the assumption is not critical that the server and the aggregator share a secret key. $\mathsf{EEA}[\mathsf{GT},\mathsf{AM}]$ is specified as follows:

Steps 1 to 3: Identical to those of $\mathsf{EA}[\mathsf{GT},\mathsf{AM}]$.

Step 4: $t_j \leftarrow R_r(id_j, c \| t_j)$ for $1 \le j \le n$.

Step 5: Identical to Step 4 of $\mathsf{EA}[\mathsf{GT},\mathsf{AM}]$.

The sole difference between $\mathsf{EEA}[\mathsf{GT},\mathsf{AM}]$ and $\mathsf{EA}[\mathsf{GT},\mathsf{AM}]$ is that the former utilizes R to randomize the tags from the entities. Thus, Theorems 1 and 2 hold for $\mathsf{EEA}[\mathsf{GT},\mathsf{AM}]$ as well as for $\mathsf{EA}[\mathsf{GT},\mathsf{AM}]$. In addition, $\mathsf{EEA}[\mathsf{GT},\mathsf{AM}_\chi]$ satisfies soundness if R is a secure PRF:

Theorem 4. *Let* $\mathbf{A}$ *be any adversary against* $\mathsf{EEA}[\mathsf{GT},\mathsf{AM}_\chi]$ *for soundness. Suppose that* $\mathbf{A}$ *triggers at most* q_r *runs of* $\mathsf{EEA}[\mathsf{GT},\mathsf{AM}_\chi]$ *and makes at most* q_t *tagging queries. Suppose that the runs of* $\mathsf{EEA}[\mathsf{GT},\mathsf{AM}_\chi]$ *conduct at most* q_v *tests in total and* R *is called at most* q_p *times in total. Then, there exists some adversary* $\dot{\mathbf{A}}$ *such that*

$$\mathrm{Adv}^{\mathrm{snd}}_{\mathsf{EEA}[\mathsf{GT},\mathsf{AM}_\chi]}(\mathbf{A}) \le \mathrm{Adv}^{\mathrm{prf}}_R(\dot{\mathbf{A}}) + q_\mathrm{r}^2/2^{\nu+1} + q_\mathrm{v}/2^\tau.$$

$\dot{\mathbf{A}}$ *makes at most* q_p *queries to its oracle, and its running time is at most about that of* $\mathscr{G}^{\mathrm{snd}}_{\mathsf{EEA}[\mathsf{GT},\mathsf{AM}_\chi],\mathbf{A}}$.

Proof. Let $\mathsf{EEA}^\rho[\mathsf{GT},\mathsf{AM}_\chi]$ be identical to $\mathsf{EEA}[\mathsf{GT},\mathsf{AM}_\chi]$ except that the former uses $\rho : \mathcal{I} \times \{0,1\}^{\nu+\tau} \to \{0,1\}^\tau$ chosen uniformly at random instead of R_r with $r \twoheadleftarrow \mathcal{R}$. The adversary $\dot{\mathbf{A}}$ against R is given access to either R_r or ρ. $\dot{\mathbf{A}}$ runs $\mathscr{G}^{\mathrm{snd}}_{\mathsf{EEA}[\mathsf{GT},\mathsf{AM}_\chi],\mathbf{A}}$ or $\mathscr{G}^{\mathrm{snd}}_{\mathsf{EEA}^\rho[\mathsf{GT},\mathsf{AM}_\chi],\mathbf{A}}$ with the use of R_r or ρ, respectively. $\dot{\mathbf{A}}$ outputs 1 iff $\mathbf{A}$ is successful for soundness. Then,

$$\mathrm{Adv}^{\mathrm{snd}}_{\mathsf{EEA}[\mathsf{GT},\mathsf{AM}_\chi]}(\mathbf{A}) \le \mathrm{Adv}^{\mathrm{snd}}_{\mathsf{EEA}^\rho[\mathsf{GT},\mathsf{AM}_\chi]}(\mathbf{A}) + \mathrm{Adv}^{\mathrm{prf}}_R(\dot{\mathbf{A}})$$

since

$$\begin{aligned}\mathrm{Adv}^{\mathrm{prf}}_R(\dot{\mathbf{A}}) &= \left|\Pr[\dot{\mathbf{A}}^{R_r} = 1] - \Pr[\dot{\mathbf{A}}^{\rho} = 1]\right| \\ &= \left|\mathrm{Adv}^{\mathrm{snd}}_{\mathsf{EEA}[\mathsf{GT},\mathsf{AM}_\chi]}(\mathbf{A}) - \mathrm{Adv}^{\mathrm{snd}}_{\mathsf{EEA}^\rho[\mathsf{GT},\mathsf{AM}_\chi]}(\mathbf{A})\right|.\end{aligned}$$

For $\mathscr{G}^{\mathrm{snd}}_{\mathsf{EEA}^\rho[\mathsf{GT},\mathsf{AM}_\chi],\mathbf{A}}$, let $\mathtt{Col}$ be the event that there exists a collision among the challenges generated in the runs of $\mathsf{EEA}^\rho[\mathsf{GT},\mathsf{AM}_\chi]$. Then,

$$\mathrm{Adv}^{\mathrm{snd}}_{\mathsf{EEA}^\rho[\mathsf{GT},\mathsf{AM}_\chi]}(\mathbf{A}) \le \Pr[\mathtt{Col}] + \Pr[\mathscr{G}^{\mathrm{snd}}_{\mathsf{EEA}^\rho[\mathsf{GT},\mathsf{AM}_\chi],\mathbf{A}} = 1 \mid \overline{\mathtt{Col}}\,].$$

Since $\mathbf{A}$ triggers at most q_r runs of $\mathsf{EEA}[\mathsf{GT},\mathsf{AM}_\chi]$,

$$\Pr[\mathtt{Col}] \le (q_\mathrm{r}(q_\mathrm{r}-1)/2)/2^\nu \le q_\mathrm{r}^2/2^{\nu+1}.$$

Finally, let us see that

$$\Pr[\mathscr{G}^{\mathrm{snd}}_{\mathsf{EEA}^\rho[\mathsf{GT},\mathsf{AM}_\chi],\mathbf{A}} = 1 \mid \overline{\mathtt{Col}}\,] \le q_\mathrm{v}/2^\tau.$$

Let $c^{(i)}$ be the challenge in the i-th run of $\mathsf{EEA}^\rho[\mathsf{GT},\mathsf{AM}_\chi]$ and $t_{i,j} := F_{k_j}(c^{(i)})$. If $\mathtt{Col}$ does not occur, then $\rho(id_j, c^{(i)} \| t_{i,j})$ is chosen uniformly at random. Thus, the probability that the result of a test involving $(id_j, t'_{i,j})$ such that $t'_{i,j} \ne t_{i,j}$ happens to be valid is at most $1/2^\tau$. □

6. Performance Evaluation

We implemented the verification algorithms of group-testing aggregate entity authentication for $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}_{\mathrm{X}}]$, $\mathsf{EEA}[\mathsf{GT}, \mathsf{AM}_{\mathrm{X}}]$, and $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}_{\mathrm{H}}]$. We used the MAC function HMAC-SHA-256 for tagging and SHA-256 to aggregate tags for AM_{H}. For GT, we adopted non-adaptive group testing and used d-disjunct matrices generated by the shifted transversal design (STD) [30], where d is the upper bound on the number of invalid entities.

We implemented the algorithms in Python 3.10.9 and utilized the modules `hmac` and `hashlib` for SHA-256 and HMAC-SHA-256. We evaluated the performance of our implementations on a MacBook Pro with Apple M1, 16 GB of memory, and macOS Ventura 13.3.1.

The results are summarized in Table 1. For the numbers of the entities 100, 1000, and 10,000, the sizes of the matrices are 66×100, 666×1000, and $6969 \times$ 10,000, respectively. They are 5-, 17-, and 68-disjunct matrices, respectively.

Each time presented in Table 1 is the smallest of ten measurements. The "Tagging" column shows the time required to generate all the tags for the entities. Thus, they almost equal the time to verify all the tags of the entities one by one. For the same number of entities, there is no significant difference in the times required for verification by $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}_{\mathrm{X}}]$, $\mathsf{EEA}[\mathsf{GT}, \mathsf{AM}_{\mathrm{X}}]$, and $\mathsf{EA}[\mathsf{GT}, \mathsf{AM}_{\mathrm{H}}]$. They depend on the numbers of 1's in the group-testing matrices, which are 600, 18,000, and 690,000 for 100, 1000, and 10,000 entities, respectively.

Table 1. Runtime (milliseconds).

Number of Entities	Tagging	Verification		
		$\mathsf{EA}[\mathsf{GT}, \mathsf{AM}_{\mathrm{X}}]$	$\mathsf{EEA}[\mathsf{GT}, \mathsf{AM}_{\mathrm{X}}]$	$\mathsf{EA}[\mathsf{GT}, \mathsf{AM}_{\mathrm{H}}]$
100	1.37×10^{-1}	2.65×10^{-1}	4.28×10^{-1}	3.05×10^{-1}
1000	7.83×10^{-1}	3.61	4.18	3.66
10,000	7.31	8.88×10^{1}	1.03×10^{2}	1.24×10^{2}

Figure 3 presents more details on the runtime for verification of $\mathsf{EEA}[\mathsf{GT}, \mathsf{AM}_{\mathrm{X}}]$ with 1000 entities and $2 \leq d \leq 26$. Table 2 presents the number of rows and the number of 1's in the group-testing matrices used for the experiments. If $d \geq 27$, then $\mathsf{EEA}[\mathsf{GT}, \mathsf{AM}_{\mathrm{X}}]$ cannot reduce the amount of communication between the server and the aggregator.

In Figure 3, the orange dots represent the times. For reference, we also give the blue dots representing the values of (the number of 1's in the group-testing matrix)/5000. As shown in Table 2, for group-testing matrices based on STD, the number of rows increases with the value of d. On the other hand, this is not necessarily the case for the number of 1's.

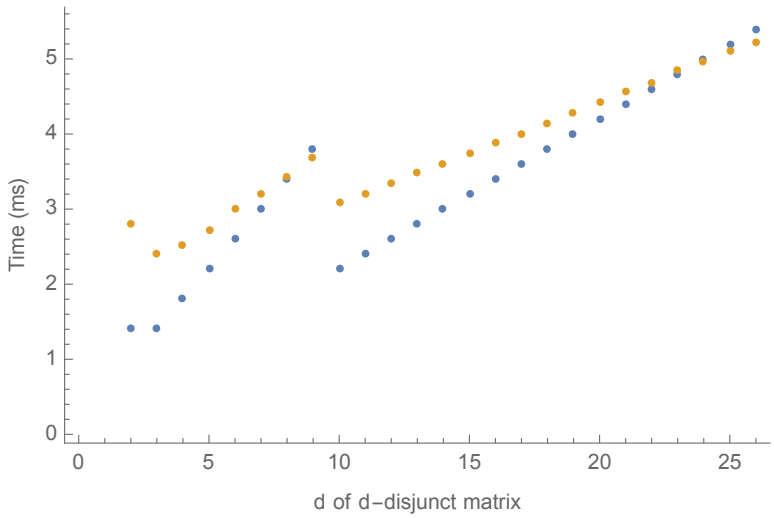

Figure 3. Runtime for verification of $\mathsf{EEA}[\mathsf{GT}, \mathsf{AM}_{\mathrm{X}}]$ with 1000 entities.

Table 2. The number of rows and the number of 1's of d-disjunct matrices used for the experiments on Figure 3.

d	2	3	4	5	6	7	8	9	10
# rows	49	77	99	121	169	255	289	361	407
# 1's	7000	7000	9000	11,000	13,000	15,000	17,000	19,000	11,000
d	11	12	13	14	15	16	17	18	19
# rows	444	481	518	555	592	629	666	703	740
# 1's	12,000	13,000	14,000	15,000	16,000	17,000	18,000	19,000	20,000
d	20	21	22	23	24	25	26		
# rows	777	814	851	888	925	962	999		
# 1's	21,000	22,000	23,000	24,000	25,000	26,000	27,000		

7. Concluding Remark

We have introduced and explored group-testing aggregate authentication. We have first formalized the scheme and security requirements. Then, we have presented a general construction utilizing a group-testing scheme and an aggregate MAC scheme. We have reduced the security properties of the generic construction and its enhancement to those of the underlying group testing and aggregate MAC. Finally, we have shown results on the performance evaluation of the proposed construction instantiated with SHA-256 and HMAC.

The proposed construction can easily be deployed due to its simplicity. In addition, any progress in group testing and aggregate MAC will benefit it. Future work is to improve the performance further. It is interesting to see if the idea of Minematsu and Kamiya [12] is effective for our proposed construction.

Author Contributions: Conceptualization, S.H. and J.S.; methodology, S.H. and J.S.; software, S.H.; validation, S.H. and J.S.; formal analysis, S.H. and J.S.; investigation, S.H. and J.S.; resources, S.H.; data curation, S.H. and J.S.; writing—original draft preparation, S.H.; writing—review and editing, S.H. and J.S.; visualization, S.H.; supervision, J.S.; project administration, J.S.; funding acquisition, J.S. All authors have read and agreed to the published version of the manuscript.

Funding: These research results were obtained from the commissioned research (No.03901) by National Institute of Information and Communications Technology (NICT), Japan.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing is not applicable to this article.

Acknowledgments: We would like to thank Kazuhiko Minematsu for providing us disjunct matrices generated by using the STD method.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MAC	Message authentication code
IoT	Internet of things
XOR	Exclusive or
PRF	Pseudorandom function
STD	Shifted transversal design

References

1. Katz, J.; Lindell, A.Y. Aggregate Message Authentication Codes. In Proceedings of the Topics in Cryptology—CT-RSA 2008, The Cryptographers' Track at the RSA Conference 2008, San Francisco, CA, USA, 8–11 April 2008; Malkin, T., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2008; Volume 4964, pp. 155–169. [CrossRef]
2. Dorfman, R. The Detection of Defective Members of Large Populations. *Ann. Math. Stat.* **1943**, *14*, 436–440. [CrossRef]
3. Du, D.Z.; Hwang, F.K. *Combinatorial Group Testing and Its Applications*, 2nd ed.; Series on Applied Mathematics; World Scientific: Singapore, 2000; Volume 12.
4. *FIPS PUB 180-4*; Secure Hash Standard (SHS). National Institute of Standards and Technology: Gaithersburg, MD, USA, 2015.
5. *FIPS PUB 198-1*; The Keyed-Hash Message Authentication Code (HMAC). National Institute of Standards and Technology: Gaithersburg, MD, USA, 2008.
6. Eikemeier, O.; Fischlin, M.; Götzmann, J.; Lehmann, A.; Schröder, D.; Schröder, P.; Wagner, D. History-Free Aggregate Message Authentication Codes. In Proceedings of the Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, 13–15 September 2010; Garay, J.A., Prisco, R.D., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6280, pp. 309–328. [CrossRef]
7. Sato, S.; Hirose, S.; Shikata, J. Sequential Aggregate MACs from Any MACs: Aggregation and Detecting Functionality. *J. Internet Serv. Inf. Secur.* **2019**, *9*, 2–23. [CrossRef]
8. Ishii, Y.; Tada, M. Structurally aggregate message authentication codes. In Proceedings of the International Symposium on Information Theory and Its Applications, ISITA 2020, Kapolei, HI, USA, 24–27 October 2020; pp. 339–343.
9. Goodrich, M.T.; Atallah, M.J.; Tamassia, R. Indexing Information for Data Forensics. In Proceedings of the Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, 7–10 June 2005; Ioannidis, J., Keromytis, A.D., Yung, M., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3531, pp. 206–221. [CrossRef]
10. Minematsu, K. Efficient Message Authentication Codes with Combinatorial Group Testing. In Proceedings of the Computer Security—ESORICS 2015—20th European Symposium on Research in Computer Security, Vienna, Austria, 21–25 September 2015; Proceedings, Part I; Pernul, G., Ryan, P.Y.A., Weippl, E.R., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2015; Volume 9326, pp. 185–202. [CrossRef]
11. Black, J.; Rogaway, P. A Block-Cipher Mode of Operation for Parallelizable Message Authentication. In Proceedings of the Advances in Cryptology—EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, 28 April–2 May 2002; Knudsen, L.R., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2332, pp. 384–397. [CrossRef]
12. Minematsu, K.; Kamiya, N. Symmetric-Key Corruption Detection: When XOR-MACs Meet Combinatorial Group Testing. In Proceedings of the Computer Security—ESORICS 2019—24th European Symposium on Research in Computer Security, Luxembourg, 23–27 September 2019; Proceedings, Part I; Sako, K., Schneider, S., Ryan, P.Y.A., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2019; Volume 11735, pp. 595–615. [CrossRef]
13. Hirose, S.; Shikata, J. Non-adaptive Group-Testing Aggregate MAC Scheme. In Proceedings of the Information Security Practice and Experience—14th International Conference, ISPEC 2018, Tokyo, Japan, 25–27 September 2018; Su, C., Kikuchi, H., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2018; Volume 11125, pp. 357–372. [CrossRef]
14. Hirose, S.; Shikata, J. Aggregate Message Authentication Code Capable of Non-Adaptive Group-Testing. *IEEE Access* **2020**, *8*, 216116–216126. [CrossRef]
15. Sato, S.; Shikata, J. Interactive Aggregate Message Authentication Scheme with Detecting Functionality. In *Advanced Information Networking and Applications, Proceedings of the 33rd International Conference on Advanced Information Networking and Applications, AINA 2019, Matsue, Japan, 27–29 March 2019*; Barolli, L., Takizawa, M., Xhafa, F., Enokido, T., Eds.; Advances in Intelligent Systems and Computing; Springer: Berlin/Heidelberg, Germany, 2019; Volume 926, pp. 1316–1328. [CrossRef]
16. Anada, H.; Kamibayashi, D. Quantum Security and Implementation Evaluation of Non-adaptive Group-Testing Aggregate Message Authentication Codes. In Proceedings of the Eighth International Symposium on Computing and Networking Workshops, CANDAR 2020 Workshops, Naha, Japan, 24–27 November 2020; pp. 307–313. [CrossRef]
17. Sato, S.; Shikata, J. Quantum-Secure (Non-)Sequential Aggregate Message Authentication Codes. In Proceedings of the Cryptography and Coding—17th IMA International Conference, IMACC 2019, Oxford, UK, 16–18 December 2019; Albrecht, M., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2019; Volume 11929, pp. 295–316. [CrossRef]
18. Ogawa, Y.; Sato, S.; Shikata, J.; Imai, H. Aggregate Message Authentication Codes with Detecting Functionality from Biorthogonal Codes. In Proceedings of the IEEE International Symposium on Information Theory, ISIT 2020, Los Angeles, CA, USA, 21–26 June 2020; pp. 868–873. [CrossRef]
19. Hirose, S.; Shikata, J. Group-Testing Aggregate Entity Authentication. In Proceedings of the IEEE Information Theory Workshop, ITW 2023, Saint-Malo, France, 23–24 April 2023.
20. Bellare, M.; Rogaway, P. Entity Authentication and Key Distribution. In Proceedings of the Advances in Cryptology—CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, CA, USA, 22–26 August 1993; Stinson, D.R., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1993; Volume 773, pp. 232–249. [CrossRef]

21. Bellare, M.; Rogaway, P. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In Proceedings of the CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, VA, USA, 3–5 November 1993; Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V., Eds.; ACM: New York, NY, USA, 1993; pp. 62–73. [CrossRef]
22. Dyachkov, A.G.; Rashad, A.M.; Rykov, V.V. Superimposed distance codes. *Probl. Control Inf. Theory* **1989**, *18*, 237–250.
23. Porat, E.; Rothschild, A. Explicit Non-adaptive Combinatorial Group Testing Schemes. In Proceedings of the Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, 7–11 July 2008; Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games; Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5125, pp. 748–759. [CrossRef]
24. Aldridge, M.; Johnson, O.; Scarlett, J. Group Testing: An Information Theory Perspective. *Found. Trends Commun. Inf. Theory* **2019**, *15*, 196–392. [CrossRef]
25. Erdös, P.; Frankl, P.; Füredi, Z. Families of Finite Sets in Which No Set Is Covered by the Union of *r* Others. *Isr. J. Math.* **1985**, *51*, 79–89. [CrossRef]
26. Dyachkov, A.G.; Rykov, V.V. Bounds on the Length of Disjunctive Codes. *Probl. Inf. Transm.* **1982**, *18*, 7–13.
27. Shangguan, C.; Ge, G. New Bounds on the Number of Tests for Disjunct Matrices. *IEEE Trans. Inf. Theory* **2016**, *62*, 7518–7521. [CrossRef]
28. Li, C.H. A Sequential Method for Screening Experimental Variables. *J. Am. Stat. Assoc.* **1962**, *57*, 455–477. [CrossRef]
29. Eppstein, D.; Goodrich, M.T.; Hirschberg, D.S. Improved Combinatorial Group Testing Algorithms for Real-World Problem Sizes. *SIAM J. Comput.* **2007**, *36*, 1360–1375. [CrossRef]
30. Thierry-Mieg, N. A new pooling strategy for high-throughput screening: The Shifted Transversal Design. *BMC Bioinform.* **2006**, *7*, 28. [CrossRef] [PubMed]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.