PAPER *Special Section on Information and Communication Technologies for Safe and Secure Life*

Constraints and Evaluations on Signature Transmission Interval for Aggregate Signatures with Interactive Tracing Functionality*

Ryu ISHII[†,††a)], **Kyosuke YAMASHITA**[††,†††], **Zihao SONG**[††††], **Yusuke SAKAI**[††], ***Nonmembers***, **Tadanori TERUYA**[††], ***Member***, **Takahiro MATSUDA**[††], ***Nonmember***, **Goichiro HANAOKA**[††], ***Member***, **Kanta MATSUURA**[†], ***Senior Member***, ***and*** **Tsutomu MATSUMOTO**[††,††††], ***Member***

SUMMARY Fault-tolerant aggregate signature (FT-AS) is a special type of aggregate signature that is equipped with the functionality for tracing signers who generated invalid signatures in the case an aggregate signature is detected as invalid. In existing FT-AS schemes (whose tracing functionality requires multi-rounds), a verifier needs to send a feedback to an aggregator for efficiently tracing the invalid signer(s). However, in practice, if this feedback is not responded to the aggregator in a sufficiently fast and timely manner, the tracing process will fail. Therefore, it is important to estimate whether this feedback can be responded and received in time on a real system. In this work, we measure the total processing time required for the feedback by implementing an existing FT-AS scheme, and evaluate whether the scheme works without problems in real systems. Our experimental results show that the time required for the feedback is 605.3 ms for a typical parameter setting, which indicates that if the acceptable feedback time is significantly larger than a few hundred ms, the existing FT-AS scheme would effectively work in such systems. However, there are situations where such feedback time is not acceptable, in which case the existing FT-AS scheme cannot be used. Therefore, we further propose a novel FT-AS scheme that does not require any feedback. We also implement our new scheme and show that a feedback in this scheme is completely eliminated but the size of its aggregate signature (affecting the communication cost from the aggregator to the verifier) is 144.9 times larger than that of the existing FT-AS scheme (with feedbacks) for a typical parameter setting, and thus has a trade-off between the feedback waiting time and the communication cost from the verifier to the aggregator with the existing FT-AS scheme.

key words: *sensor networks, aggregate signature*

1. Introduction

An aggregate signature scheme [2]–[5] is a signature scheme in which multiple signatures can be aggregated into a single aggregate signature. Aggregate signature schemes potentially allow us to significantly reduce the communication cost in information systems where a large number of signed data is transmitted. For instance, an aggregate signature scheme can be used for alive monitoring of devices in a factory. Suppose that there are thousands of devices that send their own signatures as survival reports. Thanks to an aggregate signature scheme, the thousands of reports can be aggregated into a single aggregate signature. Thus, the factory can pass the entire reports to the monitoring center by sending the aggregate signature, which will drastically reduce the bandwidth consumption.

Here, however, a potential problem is that, if invalid signatures are contained in the aggregate signature, then the entire reports would be rejected. In this paper, we call signers who generate invalid signatures *rogue* signers. Rogue signers can be seen to model signers who generate invalid signature probabilistically (due to, e.g., the failure of devices), or signers that are corrupted by a malicious party**. At first glance, it seems trivial to detect invalid signatures if an aggregator verifies individual signatures before aggregation. In practice, however, aggregators are assumed to be relatively small devices on the communication path, such as IoT devices, which have small memory size and would be difficult to verify many signatures sufficiently fast. Therefore, we assume aggregators are inexpensive small devices that can aggregate fast but cannot verify individual signatures sufficiently fast before aggregation or store them all.

For resolving this problem, Hartung et al. [6] proposed Fault-Tolerant Aggregate Signature (FT-AS for short) which is equipped with the functionality for identifying rogue signers. In FT-AS schemes, the aggregator temporarily stores all signatures which have not been aggregated in its memory, and when an invalid aggregate signature is detected, it identifies the rogue signer(s) by using combinatorial methods such as cover-free family [7]–[9], or group testing [10], [11]. Note that these schemes are more efficient than the trivial scheme because the combinatorial methods require light computation and less times of verification than the trivial method. A potential problem in these schemes is that the aggregator is required to have a relatively large memory which can store all individual (non-aggregated) signatures. Since the aggregator is assumed to be a non-powerful device, it is not always the case that such a sufficiently large memory is available to

Manuscript received April 10, 2023.
Manuscript revised August 19, 2023.
Manuscript publicized October 10, 2023.

†The authors are with the University of Tokyo, Tokyo, 153-8505 Japan.
††The authors are with the National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, 135-0064 Japan.
†††The author is with Osaka University, Suita-shi, 565-0871 Japan.
††††The authors are with the Yokohama National University, Yokohama-shi, 240-8501 Japan.

*This work was partially supported by the Cabinet Office (CAO), Cross-ministerial Strategic Innovation Promotion Program (SIP), "Cyber Physical Security for IoT Society", JPNP18015 (funding agency: NEDO). A preliminary version of this paper was presented at ESORICS 2022 ADIoT Workshop [1]. We have added a new section discussing implications of our results in Sect. 1.2.

a) E-mail: ryuishii@iis.u-tokyo.ac.jp

DOI: 10.1587/transfun.2023SSP0002

**In this paper, we mainly focus on signers that probabilistically generate invalid signatures. However, we can think of them as corrupted by a malicious party.

the aggregator. Therefore, following [6], there have been proposed some attempts to reduce the required memory size for the aggregator in the context of FT-AS. Especially, [12] investigated an approach for reducing the aggregator's memory size by introducing a *multi-round tracing* [13]. This special type of FT-AS was called *aggregate signature with interactive tracing functionality (ASIT)* in [12]. As mentioned above, one possible application of FT-AS is alive monitoring of edge devices in a factory, where each edge device periodically sends a signature and a detecting process for the rogue signers is carried out by using multiple aggregated signatures at each time period. In such a scenario, in contrast to the single-round setting, FT-AS schemes in the multi-round setting do not identify rogue signers in a single execution of the tracing functionality. Typically, it works as follows: Once the aggregator receives individual signatures, it aggregates them by following some predetermined way (specified by the underlying combinatorial method such as Dynamic Traitor Tracing (DTT) [14]), and then sends the aggregate signatures to the verifier. The verifier, on receiving the aggregate signatures, verifies them to determine the next way of aggregation (by running the combinatorial method), and sends it to the aggregator as a *feedback*. When the aggregator receives individual signatures in the next time period, it aggregates them based on the way that is determined by the feedback. This process is repeated until all rogue signers are identified.

As mentioned above, in existing multi-round FT-AS schemes, the verifier sends a feedback to the aggregator for efficiently tracing rogue signers. Thus, in real systems, this feedback must be transmitted in a sufficiently fast and timely manner to the aggregator so that the aggregator can proceed to the next step of the tracing procedure in time. Therefore, it is important to investigate whether this feedback can be sent and received sufficiently fast on a real system, and might be necessary to consider an alternative scheme if the feedback waiting time is not acceptable.

1.1 Our Contribution

To the best of our knowledge, the most efficient multi-round FT-AS scheme is the one proposed by Ishii et al. [12]. (In the following, we refer to this scheme as AS-FT-2, following the description in [12].) The first contribution of our work is that we measure the total processing time required for the feedback by implementing AS-FT-2. In order to realize an environment in cyberspace that is as close to a real system as possible, we implement this scheme in a simulation environment built on Amazon Web Services (AWS). We measure the performance of the algorithms, and find out that the feedback waiting time is 605.3 ms on average in the setting with the number of signers $N = 3000$ including $d = 5$ rogue signers (for which we have the above mentioned example of a factory in mind). The feedback waiting time is asymptotically proportional to N. This indicates that in applications whose acceptable feedback time is significantly larger than a few hundred ms, the existing FT-AS scheme can be used without problems. However, these schemes cannot be used if applications require faster feedback time. For instance, the average requirement for the transmission interval is considered to be 5 seconds to an hour [15], whereas in some real-time communication systems, the data transmission interval for cooperative intelligent transportation systems is considered to be 200 ms [16]. The existing scheme would not be suitable for the latter case.

Our second contribution is that we propose a new variant of an ASIT scheme [12] that does not require a feedback, in anticipation of applications where the feedback waiting time is shorter so that existing schemes cannot be used. The idea behind our proposed scheme is that we let the aggregator decide how to generate aggregate signatures beforehand. More specifically, our scheme employs Sequential Traitor Tracing (STT) [17] as a tracing functionality, instead of DTT employed in the existing ASIT scheme (see Sect. 3 for the technical details). We also implement our new scheme, and compare the performances with AS-FT-2. The experiment reveals that the new scheme runs faster than the existing scheme, but requires more communication cost (i.e., the bandwidth for sending aggregate signatures). See Sect. 5 for the discussion.

1.2 Implications of Our Results: Suitable Schemes for Individual Applications

In this subsection, we discuss the implications of our results. In particular, we provide specific recommended multi-round FT-AS schemes for some applications. We illustrate this in Fig. 1. Note that this is a list of recommended schemes that seem to be most suitable, and other schemes can also be applied to the same applications. In addition, when an aggregator is sufficiently *powerful* in terms of computation or memory size, we can use the trivial method that the aggregator verifies all individual signatures before aggregation or store all individual signatures to trace invalid signatures. If the aggregator is sufficiently powerful then we can use this trivial method. Therefore, we only consider the case where such a powerful aggregator is not available.

Our results show that for applications where high communication rate (more than about 1 Mbps) is available but we need to transmit a new message in real time with an interval of less than 1 second (the left-upper region of Fig. 1),

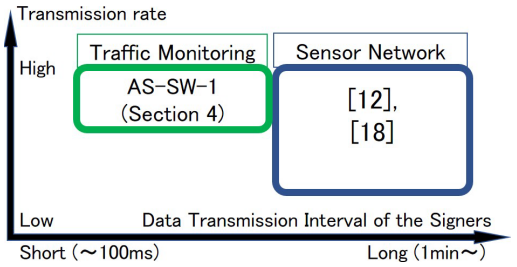

Fig. 1 A map of the most suitable scheme in existing multi-round FT-AS schemes for a network where aggregators have small memory and low computational power.

our proposed scheme (presented in Sect. 4) is recommended. Such applications include, for example, traffic monitoring.

For applications where the transmission rate is low (about 128 kbps) but it is sufficient to send the latest information at a relatively long interval of a few seconds or more (the right-lower region of Fig. 1), the schemes in [12] and [18] are recommended[†]. Such applications include sensor networks.

For applications where high communication rate is available and where it is sufficient to transmit a new message at relatively long intervals of a few seconds or more (the right-upper region of Fig. 1), a variety of FT-AS schemes can naturally be applied. However, among those schemes, we recommend [12] and [18] as the most suitable schemes because of their very small size of data transmission. Of course, AS-SW-1 (presented in Sect. 4) is also applicable, but since the communication data size of AS-SW-1 is much larger than that of [12] and [18], there is no particular merit except for being able to accept a short interval.

For applications where only low transmission rate is available and the latest message must be transmitted in real time at intervals of less than 1 second (the left-lower region of Fig. 1), there are currently no recommended schemes. In such a situation, if it is necessary to use a FT-AS scheme, the only way is to use the trivial methods using a *powerful* aggregator as described above.

Note that our proposed scheme is not always the best for such time-constrained network settings. Our claim is that only our scheme is applicable to some networks where existing schemes have not been assumed.

1.3 Related Work

Boneh et al. [2] proposed the first aggregate signature scheme (together with its concept itself), which is secure in the random oracle model and based on the BLS signature [19] in groups with efficiently computable bilinear maps. Hohenberger et al. [20] gave an aggregate signature scheme using multilinear maps in the standard model. These schemes can aggregate individual signatures as well as already aggregated signatures in any order.

Hartung et al. [6] proposed fault-tolerant aggregate signature schemes. In their schemes, a cover-free family [7]–[9], which is a combinatorial scheme, is used to determine sets of individual signatures to be aggregated. Several works [13], [18], [21] proposed efficient aggregate signatures with a tracing functionality based on group testing [10], [11]. These schemes were shown to be secure against static attackers.

There are other types of aggregate signature schemes. One is sequential aggregate signature, which was first proposed by Lysyanskaya et al. [3] and shown to be secure in the random oracle model. Since then, a number of schemes have been proposed both in the random oracle model [3], [22], [23] and in the standard model [24], [25]. Another type is aggregate signature with synchronized aggregation, which was first proposed by Gentry and Ramzan [4] (in the identity-based setting) and shown to be secure in the random oracle model. Again, since then, several constructions have been proposed both in the random oracle model [4], [5] and in the standard model [5].

1.4 Paper Organization

In Sect. 2, we introduce notations and recall the definitions of ASIT. In Sect. 3, we review the ASIT scheme AS-FT-2 by Ishii et al. [12], observe the feedback waiting time of this scheme, and give a discussion on it. In Sect. 4, we propose a new ASIT scheme, called AS-SW-1, that does not require a feedback, based on a STT. In Sect. 5, we show the experimental results for the proposed scheme and discuss which of AS-FT-2 and AS-SW-1 is more suitable for some network systems. Finally, Sect. 6 concludes this work.

2. Preliminaries

2.1 Notations

We let $[n] := \{1, \ldots, n\}$. We denote the empty string by ϵ, the empty set by $\emptyset$, the message space (of a signature scheme) by $\mathcal{M}$, an unspecified polynomial by $\mathrm{poly}(\cdot)$, an unspecified negligible function by $\mathrm{negl}(\cdot)$, and the security parameter by λ. PPT stands for probalistic polynomial time. We say that P is a partition of a set $U(\subseteq [n])$ if it satisfies the following conditions: $P = (S_1, \ldots, S_p)$, $p \in [|U|]$, $S_1, \ldots, S_p \in 2^U \setminus \{\emptyset\}$, $\bigcup_{i\in[p]} S_i = U$, and $S_i \cap S_j = \emptyset$ for all $i \neq j$ $(i, j \in [p])$.

2.2 Aggregate Signatures

Here, we recall an ordinary aggregate signature and its security definition. In this paper, for simplicity, we deal with aggregate signatures which aggregate only one message and signature pair under one verification key[††].

An aggregate signature scheme consists of the five PPT algorithms (KeyGen, Sign, Verify, Agg, AggVerify).

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$: KeyGen is the key generation algorithm that takes 1^λ as input and outputs a pair $(\mathsf{pk}, \mathsf{sk})$ of public and secret keys.
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$: Sign is the signing algorithm that takes a secret key sk and a message $m \in \mathcal{M}$ as input, and outputs a signature σ.
- $1 \,/\, 0 \leftarrow \mathsf{Verify}(\mathsf{pk}, m, \sigma)$: Verify is the verification algorithm (for an individual signature) that takes a public key pk, a message m, and a signature σ as input, and outputs either 1 (valid) or 0 (invalid).

[†]AS-FT-2 [12] and the Construction I in [18] have essentially the same tracing procedure, binary splitting.

[††]In general, aggregate signatures can aggregate multiple signatures even if they are generated under the same key.

- $\tau \leftarrow \mathsf{Agg}(\{(\mathsf{pk}_i, m_i, \sigma_i)\}_i)$: Agg is the aggregation algorithm that takes as input tuples of a public key, a message, and a signature, $\{(\mathsf{pk}_i, m_i, \sigma_i)\}_i$. It then outputs an aggregate signature τ.
- $1 \,/\, 0 \leftarrow \mathsf{AggVerify}(\{(\mathsf{pk}_i, m_i)\}_i, \tau)$: AggVerify is the verification algorithm (for an aggregate signature) that takes as input pairs of a public key and a message $\{(\mathsf{pk}_i, m_i)\}_i$, and an aggregate signature τ. It then outputs either 1 (valid) or 0 (invalid).

Definition 2.1 (Correctness): An aggregate signature scheme $\Sigma_{\mathrm{AS}} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Agg}, \mathsf{AggVerify})$ satisfies correctness if for any $\lambda \in \mathbb{N}$, any $n = \mathrm{poly}(\lambda)$, and any $m_1, \ldots, m_n \in \mathcal{M}$, it holds that

$$\Pr\left[\begin{array}{l| l} & 1 \leftarrow \mathsf{AggVerify}(\{(\mathsf{pk}_i, m_i)\}_{i\in[n]}, \tau) \\ \forall i \in [n],\ (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(1^\lambda), & \\ \text{and } \sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m_i); & \\ \tau \leftarrow \mathsf{Agg}(\{(\mathsf{pk}_i, m_i, \sigma_i)\}_{i\in[n]}) & \end{array}\right] = 1.$$

For security aganist forgery, we consider EUF-CMA (Existential UnForgeability against Chosen Message Attacks) security in the model where all key pairs are generated honestly (honest-key model). The definition captures a situation where adversaries, including an aggregator, cannot forge a valid and fresh message-signature pair even if they acquired valid signatures of honest signers.

We define security using an experiment in which all signers and an aggregator except the first signer are considered as adversaries, and the adversary wins if it successfully forges a valid aggregate signature containing the first signer. The adversaries' acquisition of valid signatures is represented as oracle access.

Definition 2.2 (EUF-CMA security): An aggregate signature scheme $\Sigma_{\mathrm{AS}} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Agg}, \mathsf{AggVerify})$ satisfies EUF-CMA security if for any $\lambda \in \mathbb{N}$, any $n = \mathrm{poly}(\lambda)$ and any PPT adversary $\mathcal{A}$, it holds that $\Pr\left[\mathsf{ExpAS}^{EUF\text{-}CMA}_{\Sigma_{\mathrm{AS}},\mathcal{A}}(\lambda, n) = 1\right] = \mathrm{negl}(\lambda)$ where $\mathsf{ExpAS}^{EUF\text{-}CMA}_{\Sigma_{\mathrm{AS}},\mathcal{A}}(\lambda, n)$ is the following experiment:

$\mathsf{ExpAS}^{EUF\text{-}CMA}_{\Sigma_{\mathrm{AS}},\mathcal{A}}(\lambda, n)$
$\forall i \in [n],\ (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \Sigma_{\mathrm{AS}}.\mathsf{KeyGen}(1^\lambda);$
$Q := \emptyset;$
$(\{m_i\}_{i\in S}, \tau, S) \leftarrow$
$\quad \mathcal{A}^{\Sigma_{\mathrm{AS}}.\mathsf{Sign}(\mathsf{sk}_1,\cdot)}(\mathsf{pk}_1, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i\in[n]\setminus\{1\}})$:
Output 1 if $S \subseteq [n]$, $1 \in S$, $m_1 \notin Q$ and
$\Sigma_{\mathrm{AS}}.\mathsf{AggVerify}(\{(\mathsf{pk}_i, m_i)\}_{i\in S}, \tau) = 1$,
else output 0

where when $\mathcal{A}$ makes a query $m \in \mathcal{M}$ to the signing oracle $\Sigma_{\mathrm{AS}}.\mathsf{Sign}(\mathsf{sk}_1, \cdot)$, it computes $\sigma \leftarrow \Sigma_{\mathrm{AS}}.\mathsf{Sign}(\mathsf{sk}_1, m)$, sends σ to $\mathcal{A}$, and sets $Q \leftarrow Q \cup \{m\}$.

Note that in the experiment, the user index 1 is used as the challenge user, whose secret key is unknown to an adversary, and the remaining keys $(\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[n]\setminus\{1\}}$ are directly given to $\mathcal{A}$. Thus, the signing oracle is provided only for the index 1.

2.3 Aggregate Signatures with Interactive Tracing Functionality

Here we recall the formal definitions for an aggregate signature scheme with interactive tracing functionality (ASIT) in the form given in [12]. In Definition 2.3, Trace is an algorithm to trace adversaries from invalid aggregate signatures and generate partitions of the signer set, and PartVerify is an algorithm to verify an aggregate signature containing a specified signer.

Definition 2.3 (ASIT): An ASIT scheme consists of the PPT algorithms (KeyGen, Sign, Agg, Verify, PartVerify, Trace) that work as follows.

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$: KeyGen is the key generation algorithm that takes 1^λ as input and outputs a pair $(\mathsf{pk}, \mathsf{sk})$ of public and secret keys.
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$: Sign is the signing algorithm that takes a secret key sk and a message $m \in \mathcal{M}$ as input, and outputs a signature σ.
- $1 \,/\, 0 \leftarrow \mathsf{Verify}(\mathsf{pk}, m, \sigma)$: Verify is the verification algorithm (for an individual signature) that takes a public key pk, a message m, and a signature σ as input, and outputs either 1 (valid) or 0 (invalid).
- $\tau \leftarrow \mathsf{Agg}(f, \{(\mathsf{pk}_i, m_i, \sigma_i)\}_i)$: Agg is the aggretation algorithm that takes as input a feedback f from the verifier and tuples of a public key, a message, and a signature $\{(\mathsf{pk}_i, m_i, \sigma_i)\}_i$ as input. It then outputs an aggregate signature τ.
- $1 \,/\, 0 \leftarrow \mathsf{PartVerify}(\beta, \{(\mathsf{pk}_i, m_i)\}_i, \tau, j)$: PartVerify is the partial verification algoritm that takes as input the verifier's internal state β, pairs of a public key and a message $\{(\mathsf{pk}_i, m_i)\}_i$, an aggregate signature τ, and a target user index j. It then outputs either 1 (valid) or 0 (invalid).
- $(\beta', f, V) \leftarrow \mathsf{Trace}(\beta, \{(\mathsf{pk}_i, m_i)\}_i, \tau)$: Trace is the tracing algorithm that takes as input the verifier's internal state β, pairs of a public key and a message $\{(\mathsf{pk}_i, m_i)\}_i$, and an aggregate signature τ. It then outputs a tuple (β', f, V) where β' is the verifier's internal state in the next round, f is a feedback, and V is the traced user set. It is required that the feedback f and traced user set V can be uniquely retrieved from the internal state β'.

Note that PartVerify is not part of the actual operation of an aggregator or a verifier but is introduced for the brevity of the EUF-CMA security of ASIT.

We also present the correctness of Trace and PartVerify as functional requirements of the ASIT algorithms. The correctness of Trace represents that any signer is not traced

$\mathsf{ExpASIT}^{\mathrm{Trace}}_{\Sigma_{\mathrm{ASIT}}}(\lambda, n, \{m_i\}_{i\in[n]})$:
for $i \coloneqq 1$ to n **do**
$(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \Sigma_{\mathrm{ASIT}}.\mathsf{KeyGen}(1^\lambda)$
$t \coloneqq 1; V_1 \coloneqq \epsilon; \beta_1 \coloneqq \epsilon; f_0 \coloneqq \epsilon$
while true
for $i \coloneqq 1$ to n **do**
$\sigma_{i,t} \leftarrow \Sigma_{\mathrm{ASIT}}.\mathsf{Sign}(\mathsf{sk}_i, m_{i,t})$
$\tau_t \leftarrow \Sigma_{\mathrm{ASIT}}.\mathsf{Agg}(f_{t-1}, \{(\mathsf{pk}_i, m_{i,t}, \sigma_{i,t})\}_{i\in[n]})$
$(\beta_{t+1}, f_t, V_t) \leftarrow \Sigma_{\mathrm{ASIT}}.\mathsf{Trace}(\beta_t, \{(\mathsf{pk}_i, m_{i,t})\}_{i\in[n]}, \tau_t)$
$t \leftarrow t + 1$

$\mathsf{ExpASIT}^{\mathrm{PartVrf}}_{\Sigma_{\mathrm{ASIT}}}(\lambda, n, \beta, j, \{m_i\}_{i\in[n]})$:
for $i \coloneqq 1$ to n **do**
$(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \Sigma_{\mathrm{ASIT}}.\mathsf{KeyGen}(1^\lambda)$
$\sigma_i \leftarrow \Sigma_{\mathrm{ASIT}}.\mathsf{Sign}(\mathsf{sk}_i, m_i)$
Let f be the feedback and V be the set of traced users that are determined by β
$\tau \leftarrow \Sigma_{\mathrm{ASIT}}.\mathsf{Agg}(f, \{(\mathsf{pk}_i, m_i, \sigma_i)\}_{i\in[n]\setminus V})$
$v \leftarrow \Sigma_{\mathrm{ASIT}}.\mathsf{PartVerify}(\beta, \{(\mathsf{pk}_i, m_i)\}_{i\in[n]\setminus V}, \tau, j)$
return v

Fig. 2 The experiments used for defining the correctness of an ASIT scheme.

by Trace algorithm in any round if all users follow the ASIT algorithm.

Definition 2.4 (Correctness of Trace): Let Σ_{ASIT} be an ASIT scheme. The algorithm $\Sigma_{\mathrm{ASIT}}.\mathsf{Trace}$ satisfies correctness if for any $\lambda \in \mathbb{N}$, any $n = \mathrm{poly}(\lambda)$, any $t \in \mathbb{N}$, and any $m_1, \ldots, m_n \in \mathcal{M}$, it holds that $\Pr[V_t = \emptyset] = 1$ where V_t is the value in the experiment $\mathsf{ExpASIT}^{\mathrm{Trace}}_{\Sigma_{\mathrm{ASIT}}}(\lambda, n, \{m_i\}_{i\in[n]})$ described in Fig. 2 (left).

The correctness of PartVerify represents that if all users follow the ASIT algorithm, the PartVerify algorithm can show the correctness of an aggregate signature including a specified user.

Definition 2.5 (Correctness of PartVerify): Let Σ_{ASIT} be an ASIT scheme. The algorithm $\Sigma_{\mathrm{ASIT}}.\mathsf{PartVerify}$ satisfies correctness if for any $\lambda \in \mathbb{N}$, any $n = \mathrm{poly}(\lambda)$, any possible form of internal state β, any $j \in [n]$, and any $m_1, \ldots, m_n \in \mathcal{M}$, it holds that $\Pr[v = 1] = 1$ where v is an output of the experiment $\mathsf{ExpASIT}^{\mathrm{PartVrf}}_{\Sigma_{\mathrm{ASIT}}}(\lambda, n, \beta, j, \{m_i\}_{i\in[n]})$ described in Fig. 2 (right).

2.3.1 EUF-CMA Security

We recall EUF-CMA security for ASIT [12]. The EUF-CMA security for ASIT is also defined as well as the EUF-CMA security for standard aggregate signatures we presented in Sect. 2.2. The EUF-CMA security for ASIT captures a situation where adversaries, including an aggregator, cannot forge a valid and fresh message-signature pair even if they acquired valid signatures of honest signers. In the experiment of Definition 2.6, O_S is the signature oracle, and O_V is the verification oracle, which verifies whether the signature generated by an untraced adversary is a valid signature containing the first signer, i.e., whether the forgery of a signature is successful. If successful, the adversary wins, and the game stops; otherwise, the tracing algorithm is executed.

Definition 2.6: An ASIT scheme Σ_{ASIT} satisfies EUF-CMA security if for any $\lambda \in \mathbb{N}$, any $n = \mathrm{poly}(\lambda)$, and any PPT adversary $\mathcal{A}$, it holds that $\Pr[\mathsf{ExpASIT}^{\text{EUF-CMA}}_{\Sigma_{\mathrm{ASIT}},\mathcal{A}}(\lambda, n) = 1] = \mathrm{negl}(\lambda)$ where $\mathsf{ExpASIT}^{\text{EUF-CMA}}_{\Sigma_{\mathrm{ASIT}},\mathcal{A}}$ is the following experiment.

$\mathsf{ExpASIT}^{\text{EUF-CMA}}_{\Sigma_{\mathrm{ASIT}},\mathcal{A}}(\lambda, n)$
$\forall i \in [n], (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \Sigma_{\mathrm{ASIT}}.\mathsf{KeyGen}(1^\lambda)$;
$t \coloneqq 1; Q \coloneqq \emptyset; W_1 \coloneqq \emptyset$;
run $\mathcal{A}^{O_S(\cdot), O_V(\cdot)}(\mathsf{pk}_1, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i\in[n]\setminus\{1\}})$:
Output 0 when $\mathcal{A}$ halts

where $\mathcal{A}$ can halt at an arbitrary point, $\mathcal{A}$ is allowed to make arbitrarily (polynomially) many queries to the signing oracle O_S and the verification oracle O_V, which work as follows:

O_S**:** Given a query $m \in \mathcal{M}$ from $\mathcal{A}$, O_S runs $\sigma \leftarrow \Sigma_{\mathrm{ASIT}}.\mathsf{Sign}(\mathsf{sk}_1, m)$, returns σ to $\mathcal{A}$, and updates $Q \leftarrow Q \cup \{m\}$.

O_V**:** Given pairs of an index and a message $\{(i, m_{i,t})\}_{i\in I_t}$ and an aggregate signature τ from $\mathcal{A}$, O_V outputs 1 (indicating that $\mathcal{A}$ wins) and terminates the experiment if it holds that $\Sigma_{\mathrm{ASIT}}.\mathsf{PartVerify}(\beta_t, \{(\mathsf{pk}_i, m_{i,t})\}_{i\in I_t}, \tau_t, 1) = 1$, $1 \notin W_t$, and $m_{1,t} \notin Q$. Otherwise, O_V executes $(\beta_{t+1}, f_t, V_t) \leftarrow \Sigma_{\mathrm{ASIT}}.\mathsf{Trace}(\beta_t, \{(\mathsf{pk}_i, m_{i,t})\}_{i\in I_t}, \tau_t)$, returns (f_t, V_t) to $\mathcal{A}$, and updates $W_t = W_t \cup V_t$ and $t \leftarrow t + 1$.

Note that the user index 1 is treated as the challenge user, and an adversary is given the secret keys for the remaining users with index 2 to n, and thus the signing oracle is necessary only for the user index 1. Note also that the experiment can output 1 only if $\mathcal{A}$ makes an O_V-query that contains a forged signature with respect to the user index 1 (judged using the algorithm PartVerify).

2.3.2 R-Identifiability and Correctness

We recall R-identifiability and correctness of an ASIT scheme Σ_{ASIT}. R-identifiability guarantees that a verifier can identify all rogue signers within R rounds of executions of the tracing procedure. A potential adversary in these security notions is a set of users $C \subseteq [n]$ that may generate invalid signatures. On the other hand, correctness guarantees that no honest signers will be traced. Note that an aggregator and the verifier behave honestly. These security notions are defined based on the following experiment $\mathsf{ExpASIT}_{\Sigma_{\mathrm{ASIT}},\mathcal{A}}(\lambda, n)$ in which a stateful adversary $\mathcal{A}$ is executed:

Initialize$(1^\lambda, 1^n)$:
$I := \{1, \cdots, n\}$; $P_1 := I$; $\alpha_1 := P_1$
return (α_1, P_1)

Halve(I) :
$A :=$ the first $\lfloor |I|/2 \rfloor$ elements of I; $B := I \setminus A$
return (A, B)

Trace(α_t, i) :
$P_t := \alpha_t$; $p := (|P_t| - 1)/2$; $V := \emptyset$; parse $(Q_j)_{j \in [|P|]} \leftarrow P_t$
if $Q_i = I$ **then**
$(L_{p+1}, R_{p+1}) \leftarrow$ Halve(I); $I \leftarrow \emptyset$; $P \leftarrow P \cup (\{L_{p+1}\} \cup \{R_{p+1}\})$; $p \leftarrow p + 1$
else if $\exists j \in [p] : Q_i = L_j$ **then**
$\star$ $I \leftarrow I \cup R_j$
$\star$ **if** $|L_j| = 1$ **then**
$\star$ $V \leftarrow L_j$; $p \leftarrow p - 1$; $P \leftarrow P \setminus (\{L_j\} \cup \{R_j\})$
$\star$ **else** $(L_j, R_j) \leftarrow$ Halve(L_j)
else if $\exists j \in [p] : Q_i = R_j$ **then**
Execute the lines marked with $\star$ with switching L_j and R_j.
$P_{t+1} := P$; $V_t := V$; $\alpha_{t+1} := P$
return $(\alpha_{t+1}, P_{t+1}, V_t)$

Fig. 3 The descriptions of the DTT (FT-2) by Fiat and Tassa [14] and the subroutine Halve used in the scheme.

$\mathsf{ExpASIT}_{\Sigma_{\mathrm{ASIT}},\mathcal{A}}(\lambda, n)$
$\forall i \in [n], (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \Sigma_{\mathrm{ASIT}}.\mathsf{KeyGen}(1^\lambda)$;
$C \leftarrow \mathcal{A}(\{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i \in [n]})$;
$t := 1$; $r := 0$; $W_1 := \emptyset$; $\beta_1 := \epsilon$; $f_0 := \epsilon$; $I_1 := [n]$; $J_1 := C$;
run $\mathcal{A}^{O_T(\cdot)}(\{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i \in [n]})$:
Output $(W := \bigcup_{t'=1}^{t} V_{t'}, C, r)$ when $\mathcal{A}$ halts

where $\mathcal{A}$ can halt at an arbitrary point, and $\mathcal{A}$ is allowed to make arbitrarily (polynomially) many queries to the tracing oracle O_T. Let $W_t := \bigcup_{t'=1}^{t} V_t$, $I_t := [n] \setminus W_t$, and $J_t := C \setminus W_t$. Given a query $(\{m_{i,t}\}_{i \in I_t}, \{(m_{j,t}, \sigma_{j,t})\}_{j \in J_t})$ from $\mathcal{A}$, O_T operates as follows:

1. If there exists $j \in J_t$ such that $\Sigma_{\mathrm{ASIT}}.\mathsf{Verify}(\mathsf{pk}_j, m_{j,t}, \sigma_{j,t}) = 0$, then set $r \leftarrow r + 1$.
2. For every $i \in I_t$, compute $\sigma_{i,t} \leftarrow \Sigma_{\mathrm{ASIT}}.\mathsf{Sign}(\mathsf{sk}_i, m_{i,t})$.
3. Compute $\tau_t \leftarrow \Sigma_{\mathrm{ASIT}}.\mathsf{Agg}(f_{t-1}, \{(\mathsf{pk}_i, m_{i,t}, \sigma_{i,t})\}_{i \in I_t \cup J_t})$.
4. Compute $(\beta_{t+1}, f_t, V_t) \leftarrow \Sigma_{\mathrm{ASIT}}.\mathsf{Trace}(\beta_t, \{(\mathsf{pk}_i, m_{i,t})\}_{i \in I_t \cup J_t}, \tau_t)$.
5. Return (f_t, V_t) to $\mathcal{A}$ and set $t \leftarrow t + 1$.

We define R-identifiability and correctness of ASIT as follows.

Definition 2.7 (R-Identifiability): An ASIT scheme Σ_{ASIT} satisfies R-identifiability if for any $\lambda \in \mathbb{N}$, any $n = \mathrm{poly}(\lambda)$, and any PPT adversary $\mathcal{A}$, we have

$$\Pr[(C \not\subseteq W) \mid (W, C, r) \leftarrow \mathsf{ExpASIT}_{\Sigma_{\mathrm{ASIT}},\mathcal{A}}(\lambda, n) \wedge (r \geq R)] = \mathrm{negl}(\lambda).$$

Definition 2.8 (Correctness): An ASIT scheme Σ_{ASIT} satisfies correctness if both $\Sigma_{\mathrm{ASIT}}.\mathsf{Trace}$ and $\Sigma_{\mathrm{ASIT}}.\mathsf{PartVerify}$ satisfy correctness, and for any $\lambda \in \mathbb{N}$, any $n = \mathrm{poly}(\lambda)$, and any PPT adversary $\mathcal{A}$, we have

$$\Pr[([n] \setminus C) \cap W \neq \emptyset \mid (W, C, r) \leftarrow \mathsf{ExpASIT}_{\Sigma_{\mathrm{ASIT}},\mathcal{A}}(\lambda, n)] = \mathrm{negl}(\lambda).$$

3. Feedback Waiting Time in ASIT

This section presents our first contribution. Recall that, as discussed in Sect. 1, the feedback waiting time of a multi-round FT-AS scheme could be critical if we use it in practice. Therefore, here we evaluate the feedback waiting time of a concrete instantiation of ASIT. More specifically, we evaluate the instantiation of ASIT in [12], which is constructed from an ordinary aggregate signature scheme and DTT. In Sect. 3.1, we introduce the concrete instantiation FT-2 of DTT that is proposed by Fiat and Tassa [14], and an ASIT scheme proposed by Ishii et al. [12] that is based on an aggregate signature scheme and FT-2. We review the formal definition of DTT formalized by Ishii et al. [12] in Appendix A. In Sect. 3.2, we evaluate the feedback waiting time of the ASIT scheme. We remark that this is the first experimental result for ASIT schemes, because [12] only proposed the scheme.

3.1 Existing Instantiations of DTT and ASIT

DTT is a method to trace piracy in a contents distributing service, which works as follows: Users are divided into several subgroups, and for each subgroup a content with a unique watermark is distributed. Once a piracy is found, the contents distributor checks the watermark, divides the corresponding subgroup into smaller subgroups, and repeats this procedure until it traces a piracy. We refer to the interval between the executions of tracing as 1 *round*. Fiat and Tassa [14] proposed two DTTs, and we recall one of them.

Figure 3 illustrates the instantiation of DTT proposed in [14], named FT-2. FT-2 is a method to trace traitors in a set of users similar to a binary search. Initially, all users are included in the set I (representing *innocent*), which is bisected into L_j and R_j when a piracy is detected. Then in the next round, when piracy is found from L_j (or R_j), L_j (or R_j) is divided in two. The process is repeated until the adversary is isolated. It is known that FT-2 traces all traitors

$\mathsf{KeyGen}(1^\lambda)$:
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow \Sigma_{\mathrm{AS}}.\mathsf{KeyGen}(1^\lambda)$
  return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Sign}(\mathsf{sk}, m)$:
  $\sigma \leftarrow \Sigma_{\mathrm{AS}}.\mathsf{Sign}(\mathsf{sk}, m)$
  return σ

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$:
  $v \leftarrow \Sigma_{\mathrm{AS}}.\mathsf{Verify}(\mathsf{pk}, m, \sigma)$
  return v

$\mathsf{Agg}(f, \{(\mathsf{pk}_i, m_i, \sigma_i)\}_{i\in[n]})$:
  if $f = \epsilon$ **then**
    $(\alpha, P) \leftarrow \Sigma_{\mathrm{DTT}}.\mathsf{Initialize}(1^\lambda, 1^n)$
  else $P := f$
  parse $(S_i)_{i\in[|P|]} \leftarrow P$
  for $j := 1$ to $|P|$ **do**
    $\tau_j \leftarrow \Sigma_{\mathrm{AS}}.\mathsf{Agg}(\{(\mathsf{pk}_i, m_i, \sigma_i)\}_{i\in S_j})$
  return $\tau := (\tau_j)_{j\in[|P|]}$

$\mathsf{PartVerify}(\beta, \{(\mathsf{pk}_j, m_j)\}_{j\in[n]}, \tau, i)$:
  if $\beta = \epsilon$ **then**
    $(\alpha, P) \leftarrow \Sigma_{\mathrm{DTT}}.\mathsf{Initialize}(1^\lambda, 1^n)$
  else parse $(\alpha, P) \leftarrow \beta$
  parse $(S_i)_{i\in[|P|]} \leftarrow P$ and $(\tau_i)_{i\in[|\tau|]} \leftarrow \tau$
  find i' s.t. $i \in S_{i'}$
  $v \leftarrow \Sigma_{\mathrm{AS}}.\mathsf{AggVerify}(\{(\mathsf{pk}_j, m_j)\}_{j\in S_{i'}}, \tau_{i'})$
  return v

$\mathsf{Trace}(\beta, \{(\mathsf{pk}_i, m_i)\}_{i\in[n]}, \tau)$:
  if $\beta = \epsilon$ **then**
    $(\alpha, P) \leftarrow \Sigma_{\mathrm{DTT}}.\mathsf{Initialize}(1^\lambda, 1^n)$
  else parse $(\alpha, P) \leftarrow \beta$
  parse $(S_i)_{i\in[|P|]} \leftarrow P$ and $(\tau_i)_{i\in[|\tau|]} \leftarrow \tau$
  $f' := P$; $\beta' := (\alpha, P)$; $V := \emptyset$
  for $i := 1$ to $|\tau|$ **do**
    $v \leftarrow \Sigma_{\mathrm{AS}}.\mathsf{AggVerify}(\{(\mathsf{pk}_j, m_j)\}_{j\in S_i}, \tau_i)$
    if $v = 0$ **then**
      $(\alpha', P', V) \leftarrow \Sigma_{\mathrm{DTT}}.\mathsf{Trace}(\alpha, i)$
      $f' := P'$; $\beta' := (\alpha', P')$
      break (†)
  return (β', f', V)

Fig. 4 The instantiation Σ_{ASIT} of an ASIT scheme based on an aggregate signature scheme Σ_{AS} and FT-2 Σ_{DTT} by [12]. (†) This command exits **for** and moves to the next line.

in $d(\log_2 N + 1)$ rounds, where the number of signers is N and the number of traitors is d.

We recall the ASIT scheme proposed by Ishii et al. [12]. The idea behind the tracing functionality of this scheme is to regard rogue signers as pirates, and run FT-2 to trace them. Figure 4 illustrates the ASIT scheme Σ_{ASIT} based on an aggregate signature scheme Σ_{AS} and the DTT FT-2 Σ_{DTT}. We call this instantiation as AS-FT-2. For AS-FT-2, an aggregator aggregates all individual signatures into an aggregate signature in the first round. The verifier verifies it, and if it is invalid, the Trace algorithm of FT-2 is run to generate the next partition. This partition is sent as feedback to the aggregator. In the next round, aggregation is performed in part-by-part according to the partition in the feedback. Specifically, for each individual subset divided by the partition, only signatures in the same subset are aggregated (i.e., the number of aggregate signatures generated by the aggregator is as many as the number of subsets). Then the verifier verifies the signatures and if it finds an invalid one, it runs the Trace algorithm. These processes are repeated until all invalid signers are traced.

3.2 Evaluation of the Feedback Waiting Time of AS-FT-2

We evaluate the feedback waiting time of AS-FT-2, describe the implementation experiment of AS-FT-2 in detail and present the results. We also discuss whether the feedback waiting time meets the requirement for some applications.

3.2.1 Experimental Environment

We use a simulator implemented using Amazon Web Service (AWS) shown in [26]. For the devices, we use four Amazon EC2 instances, which are virtual devices provided by AWS. The EC2 instance types are all t3.micro (3.1 GHz Intel Xeon Scalable processor, baseline performance per virtual CPU: 10%, virtual CPU: 2, memory: 1 GiB, maximum bandwidth: 5 Gbps, baseline bandwidth: 87 Mbps). These four instances are managed by Amazon Elastic Kubernetes Service (EKS) as shown in Fig. 5 and each of them has a container image, which consists of alpine Linux 3.15 (latest) as the OS and g++ 10.3.1 as the C++ version.

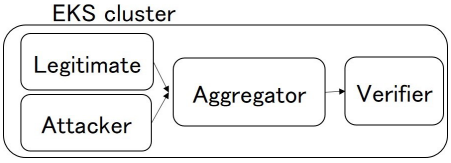

Fig. 5 Device configuration in a cluster. The legitimate node sends $N-d$ valid signatures and the rogue signer node sends d invalid signatures in each round, where N is the number of all signers and d is the number of rogue signers.

3.2.2 Experimental Setting

We capture a situation where some fixed rogue signers among many signers always send invalid signatures. The number N of all signers and the number d of rogue signers are $N = 1000, 3000$, and $d = 5, 10, 40$, respectively. These parameter settings are based on [27], which deals with identifying sensor nodes that have failed to transmit in a sensor network. It assumes a network setting similar to ours where a large number (1000–3000) of sensor nodes transmit data and some faulty nodes are included. Each honest/rogue signer sends a pair of a 128-Byte message and a signature to the aggregator. For the underlying aggregate signature scheme, we implement the BGLS aggregate signature scheme [2] by using a pairing cryptography library mcl [28] and BN254 [29] for elliptic curves. The experiment is conducted 10 times and the average is taken.

Table 1 Simulation results of AS-FT-2. **Trace** is the average execution time of the tracing algorithm, and **Feedback** is the average time from the time that the aggregator generates an aggregate signature to the time that the verifier returns a feedback. The feedback size of each round when $N = 1000, 3000$ are 3.9 kilobytes and 11.9 kilobytes respectively.

N	d	**Trace** [ms]	**Feedback** [ms]
1000	5	227.0	233.3
1000	10	171.9	179.6
1000	40	81.2	96.9
3000	5	594.4	605.3
3000	10	452.2	462.1
3000	40	205.2	221.6

3.2.3 Experimental Results

Table 1 shows the simulation results of AS-FT-2 under the experimental conditions described above. Observe that the execution time of tracing occupies a large ratio of the feedback waiting time, which further increases when N becomes large. This is because the verifier requires $N + 1$ pairing operations and a pairing operation takes longer time than other group operations in our instantiation[†]. We observe that the execution time of tracing becomes less as d increases. In this experiment, each rogue signer always sends an invalid signature, so at least one of the splits out of two includes a rogue signer and an invalid signature can be found in two times of verifications. In addition, when d is large, the number of set partitions in tracing is also large and the number of signatures in each set becomes small, then the number of pairing operations executed to find an invalid aggregate signature becomes small[††].

For the column **Feedback**, it becomes clear that when $(N, d) = (1000, 5), (3000, 5)$, the aggregator needs to wait on average of 233.3 ms, 605.3 ms, respectively, and each signer needs to have a transmission interval longer than this. The time difference of the columns **Trace** and **Feedback** is due to the communication delay to the aggregator in addition to the time of tracing.

3.2.4 Acceptable Feedback Waiting Time

We consider whether or not the feedback waiting time of AS-FT-2 is acceptable for the data transmission interval requirement of less time-constrained or real-time communication systems. We also consider whether AS-FT-2 is applicable or not in these networks systems.

In less time-constrained communication systems such as monitoring sensor networks, according to [15], to reduce power consumption and increase lifetime of nodes, the average requirement for the transmission interval is considered to be 5 seconds to an hour. From these requirements and Table 1, the feedback waiting time of AS-FT-2 is sufficiently short when AS-FT-2 is used in these networks. Therefore, AS-FT-2 is applicable in monitoring sensor networks and industrial networks.

On the other hand, in some real-time communication systems, according to [16], to guarantee traffic safety under bandwidth constraints, the data transmission interval for a cooperative intelligent transportation system is considered to be 200 ms, which is to optimize the overall efficiency of a system in features of safety indicators. From these requirements and Table 1, the feedback waiting time of AS-FT-2 is too long. Therefore, AS-FT-2 is *not* applicable for such real-time communication systems.

[†]It takes 0.679 microseconds per group operation and 286 microseconds per pairing operation in our setting.

[††]In the tracing algorithm of AS-FT-2, when the verifier finds an invalid aggregate signature, it outputs a feedback and the rest of signatures are not verified.

4. An ASIT Scheme without Feedback

In this section, we propose a new ASIT scheme that does not require a feedback, in anticipation of applications where the feedback waiting time is short so that the ASIT scheme AS-FT-2 cannot be used. The idea behind our proposed scheme is that we let the aggregator decide how to create aggregate signatures (i.e., the partitions) beforehand, in contrast to the construction of AS-FT-2, in which the verifier (the tracing algorithm) decides it adaptively during the execution. Similarly to AS-FT-2, we construct the new ASIT scheme based on an aggregate signature scheme and a piracy tracing algorithm, but in our scheme we use a Sequential Traitor Tracing (STT) [17]. STT predetermines the assignment of watermarks beforehand, and this feature allows us to remove a feedback from an ASIT scheme.

We first review the concept of STT, and then the concrete instantiation of STT, named c-SW-1 [17]. Then we provide our new construction of ASIT named AS-SW-1 based on an aggregate signature scheme and c-SW-1. As explained earlier, [12] proposed a generic construction of ASIT based on an aggregate signature scheme and a DTT. Our ASIT scheme AS-SW-1 is constructed just by replacing FT-2 used in AS-FT-2 with c-SW-1. Therefore, to prove AS-SW-1 satisfies the requirements of an ASIT scheme, it is sufficient to prove that c-SW-1 is a DTT.

4.1 Sequential Traitor Tracing

As already mentioned, STT is a variant of a piracy tracing algorithm that predetermines the watermark assignment. The assignment is actually determined by an allocation matrix M, whose entry is decided by a function family Φ. The column of M corresponds to the allocation of watermarks for each distribution. The time interval between each distribution is called *segment*. In this subsection, we introduce the allocation matrix and the function family along with several lemmas and an implementation of the function family. Then, we provide an implementation of the STT.

Let $W := [q]$ be the set of watermarks, and b ($\leq q$) and l be integers. We let $\Phi = \{\phi_{i,j} : 1 \leq i \leq b, 1 \leq j \leq l\}$, where $\phi_{i,j} : W \to W$ is a function family that satisfies the following conditions:

Condition 1: For all $x \in W$, any fixed j and a pair of the first indices (i_1, i_2) where $i_1 \neq i_2$, it holds that $\phi_{i_1,j}(x) \neq \phi_{i_2,j}(x)$.

Condition 2: For any (i_1, i_2) and (j_1, j_2) where $j_1 \neq j_2$, and any $x_1, x_2 \in W$ such that $x_1 \neq x_2$, if $\phi_{i_1,j_1}(x_1) = \phi_{i_2,j_1}(x_2)$ then $\phi_{i_1,j_2}(x_1) \neq \phi_{i_2,j_2}(x_2)$.

Now, we describe an allocation matrix M. Let Φ be a function family that satisfies the above conditions. Let M_0 and $\phi_{i,j}(M_0)$ $(i \in [b], j \in [l])$ be the following $q \times 1$ matrices:

$$M_0 = \begin{pmatrix} 1 \\ 2 \\ \vdots \\ q \end{pmatrix}, \phi_{i,j}(M_0) = \begin{pmatrix} \phi_{i,j}(1) \\ \phi_{i,j}(2) \\ \vdots \\ \phi_{i,j}(q) \end{pmatrix}.$$

Then, the allocation matrix M is described as follows:

$$M = \begin{pmatrix} M_0 & \phi_{1,1}(M_0) & \phi_{1,2}(M_0) & \cdots & \phi_{1,l}(M_0) \\ M_0 & \phi_{2,1}(M_0) & \phi_{2,2}(M_0) & \cdots & \phi_{2,l}(M_0) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ M_0 & \phi_{b,1}(M_0) & \phi_{b,2}(M_0) & \cdots & \phi_{b,l}(M_0) \end{pmatrix}.$$

Note that M has b block rows and each block consists of q rows. The i-th row of M represents the watermark assigned to user i in order (thus, it is implicitly assumed that the number of users is bq), and j-th column represents the assignment of watermarks for each user at segment j. Let (r, k) denote the k-th row of the r-th block.

Here we explain how M is used to detect piracy in a content distribution service. The content distributor distributes a watermark of a content $M(i, j)$ to the user i at the segment j, with monitoring piracy. For ease of discussion, we assume that the distributor detects exactly one piracy in every segment. Let $F_j = (f_1, \ldots, f_j)$ be the index sequence of the watermarks on the content detected by the distributor from segment 1 to segment j. Furthermore, let

$$\rho(F_j, i, s) = \begin{cases} 1 & (\text{if } f_s = M(i, j)) \\ 0 & (\text{otherwise}) \end{cases},$$

$$\rho(F_j, i) = \sum_{s=1}^{j} \rho(F_j, i, s).$$

If there exists a user i such that $\rho(F_j, i) \geq t$ for a threshold t, then the distributor regards i as a pirate and eliminates it. Regarding the threshold, Safavi-Naini and Wang [17] showed the following lemmas.

Lemma 4.1 ([17]): Let C be the set of pirates where $|C| = c$, and $F_j = (f_1, \ldots, f_j)$ be the index sequence of the watermarks on the content detected by the distributor from segment 1 to segment j. If there exists a user i such that $\rho(F_j, i) \geq c + 1$, then $i \in C$.

Regarding how many segments are necessary to trace all pirates, they also showed the following lemma.

Lemma 4.2 ([17]): Let C be the set of pirates where $|C| = c$. All pirates can be traced with at most $c^2 + c$ segments.

Lemma 4.2 tells us the relation between the number of segments and the number of pirates.

Corollary 4.1 ([17]): Let M be an $n \times (l + 1)$ allocation matrix. The STT that employs this matrix can trace at most

$$c = \left\lfloor \frac{-1 + \sqrt{5 + 4l}}{2} \right\rfloor \tag{1}$$

pirates.

4.1.1 Implementation of Φ

Safavi-Naini and Wang [17] proposed the following implementation Φ of the function family.

Theorem 4.1 ([17]): Let p be a prime number. Let $\Phi = \{\phi_{i,j} : i, j \in [(p-1)/2]\}$ $(\phi_{i,j} : Z_p^* \to Z_p^*)$ be the function family defined by $\phi_{i,j}(x) = (i+j)x \mod p$. Then, Φ satisfies Condition 1 and Condition 2.

4.1.2 Observation

When we use Φ in Theorem 4.1 in a STT, the allocation matrix should be the $n \times l$ matrix where $n = (p-1)^2/2$ and $l = (p-1)/2$. Also, the maximum number of pirates that can be traced is $c = \lfloor \frac{-1+\sqrt{2p+3}}{2} \rfloor$. Given n and c, the condition for p is expressed as $p \geq \max(1 + \sqrt{2n}, 2c^2 + 2c - 1)$ from $(p-1)^2/2 \geq n$ and $c^2 + c \leq (p-1)/2 + 1$ (note that the second inequality is due to Lemma 4.2).

4.1.3 An Instantiation of STT

Here we provide an instantiation of STT based on the function family in Theorem 4.1. Let $n := bq$ be the number of signers, $C \subseteq [n]$ be a set of pirates where $|C| = c \leq n$, and $W := [q]$ be a set of watermarks that are assumed to be given to the distributor in advance. Let $Q_{j,x}$ be the set of users to which the watermark $x \in W$ is assigned at segment j, i.e., $Q_{j,x} = \{i \in [n] : M[i, j] = x\}$.

Figure 6 illustrates the construction c-SW-1 of the STT that uses the function family Φ in Theorem 4.1. Note that in the subroutine GenMatrix, the prime p is chosen based on the above observation. We show the following lemma.

Lemma 4.3: The algorithm c-SW-1 is a DTT.

The proof of 4.3 is presented in Appendix B.

4.2 The Construction of an ASIT Scheme without a Feedback

We demonstrate an ASIT scheme that does not require a feedback based on an aggregate signature and the STT c-SW-1 (hereinafter referred to as AS-SW-1). Figure 7 illustrates AS-SW-1, where c satisfies the Eq. (1). We assume without loss of generality that the number of users n and the number

GenMatrix(n, c):
select a minimum prime p s.t.
$p \geq \max(1+\sqrt{2n}, 2c^2+2c+1)$;
$l := (p-1)/2$; $q = p-1$; $b := \lceil n/q \rceil$
for $j = 1$ to l **do**
for $i = 1$ to b **do**
for $x = 1$ to q **do**
$M[b(i-1)+x, j] = \phi_{i,j}(x)$
return M

Initialize$(1^\lambda, 1^n)$:
$j := 1$; $F := ()$;
$M \leftarrow$ GenMatrix(n, c) based on Φ
for $x := 1$ to q **do**
$Q_{1,x} := \{i \in [n] \ : \ M[i, 1] = x\}$
$P_1 := (Q_{1,1}, \ldots, Q_{1,q})$; $\alpha_1 := (j, F, P_1, M)$
return (α_1, P_1)

Trace(α, i):
parse $(j, F, P, M) \leftarrow \alpha$; $V := \emptyset$;
append i to F
for $u = 1$ to n **do**
if $\rho(F, u) = c+1$ **then**
$V \leftarrow u$
for $x := 1$ to q **do**
$Q_{j+1,x} := \{i \in [n] \ : \ M[i, j+1] = x\}$
$P_{j+1} := (Q_{j+1,1}, \ldots, Q_{j+1,q})$; $j \leftarrow j+1$
$\alpha_j := (j, F, P_j, M)$
return (α_j, P_j, V)

Fig. 6 The construction c-SW-1 of STT. GenMatrix is a subroutine that is used to generate the allocation matrix M based on the function family Φ in Theorem 4.1.

KeyGen(1^λ) :
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \Sigma_{\mathrm{AS}}.\mathsf{KeyGen}(1^\lambda)$
return $(\mathsf{pk}, \mathsf{sk})$

Sign(sk, m) :
$\sigma \leftarrow \Sigma_{\mathrm{AS}}.\mathsf{Sign}(\mathsf{sk}, m)$
return σ

Verify(pk, m, σ) :
$v \leftarrow \Sigma_{\mathrm{AS}}.\mathsf{Verify}(\mathsf{pk}, m, \sigma)$
return v

Agg$(\alpha, \{(\mathsf{pk}_i, m_i, \sigma_i)\}_{i\in[n]})$:
if $\alpha = \epsilon$ **then**
$j := 1$; $V := \emptyset$
$(\alpha, P) \leftarrow c$-SW-1.Initialize$(1^\lambda, 1^n)$
parse $(S_i)_{i\in[|P|]} \leftarrow P$
parse $(j, F, P, M) \leftarrow \alpha$
for $p := 1$ to $|P|$ **do**
$\tau_p \leftarrow \Sigma_{\mathrm{AS}}.\mathsf{Agg}(\{(\mathsf{pk}_i, m_i, \sigma_i)\}_{i\in S_p \setminus V})$
for $x := 1$ to q **do**
$Q_{j+1,x} := \{i \in [n] \ : \ M[i, j] = x\}$
$P \leftarrow (Q_{j+1,x})_{x\in[q]}$
$j \leftarrow j+1$; $\alpha \leftarrow (j, F, P, M)$
return $\tau := (\tau_p)_{p\in[|P|]}$

PartVerify$(\beta, \{(\mathsf{pk}_j, m_j)\}_{j\in[n]}, \tau, i)$:
if $\beta = \epsilon$ **then**
$j := 1$; $V := \emptyset$; $(\alpha, P) \leftarrow c$-SW-1.Initialize$(1^\lambda, 1^n)$
else parse $(\alpha, P) \leftarrow \beta$
parse $(S_i)_{i\in[|P|]} \leftarrow P$ and $(\tau_i)_{i\in[|\tau|]} \leftarrow \tau$
find i' s.t. $i \in S_{i'}$
$v \leftarrow \Sigma_{\mathrm{AS}}.\mathsf{AggVerify}(\{(\mathsf{pk}_j, m_j)\}_{j\in S_{i'}\setminus V}, \tau_{i'})$
return v

Trace$(\beta, \{(\mathsf{pk}_i, m_i)\}_{i\in[n]}, \tau)$:
if $\beta = \epsilon$ **then**
$j := 1$; $V := \emptyset$
$(\alpha, P) \leftarrow c$-SW-1.Initialize$(1^\lambda, 1^n)$
else parse $(\alpha, P) \leftarrow \beta$
parse $(S_i)_{i\in[|P|]} \leftarrow P$ and $(\tau_i)_{i\in[|\tau|]} \leftarrow \tau$
$f := \epsilon$; $\beta' := (\alpha, P)$
for $i := 1$ to $|\tau|$ **do**
$v \leftarrow \Sigma_{\mathrm{AS}}.\mathsf{AggVerify}(\{(\mathsf{pk}_j, m_j)\}_{j\in S_i\setminus V}, \tau_i)$
if $v = 0$ **then**
$(\alpha', P', V) \leftarrow c$-SW-1.Trace$(\alpha, i)$
$\beta' := (\alpha', P')$
break (†)
return (β', f, V)

Fig. 7 The proposed ASIT scheme AS-SW-1. (†) This command exits **for** and moves to the next line.

of rogue signers c are given to the algorithms. AS-SW-1 performs similarly to AS-FT-2 in that aggregation is performed in part-by-part according to the partition, but unlike AS-FT-2, AS-SW-1 avoids the feedback communication between an aggregator and a verifier. Specifically, the aggregator and the verifier both execute c-SW-1 to share how the signer set is divided in each round, and the partition P held by the aggregator and the verifier are stored in the internal states α and β, respectively.

Observe that both Agg and Trace run c-SW-1.Initialize when they are initiated. Because c-SW-1, including the subroutine GenMatrix, is deterministic, these algorithms share the same partition P (or the same allocation matrix). Note that the output f by Trace is an empty string (this feedback is put just to follow the syntax of ASIT, and thus Agg does not receive this f in an actual execution). Also, when the verifier identifies rogue signers, it sends the traced attacker set V to the aggregator.

4.2.1 Security

Observe that AS-SW-1 is constructed by just replacing the DTT FT-2 used in AS-FT-2 with c-SW-1. Furthermore, as shown in Lemma 4.3, c-SW-1 is a DTT. Therefore, AS-SW-1 follows the generic construction of ASIT proposed by Ishii et al. [12], which implies the following lemma.

Lemma 4.4: Assume the underlying aggregate signature scheme Σ_{AS} is EUF-CMA secure, and the underlying DTT is c-SW-1. Then, AS-SW-1 is an ASIT scheme satisfying

Table 2 Theoretical evaluations. Trivial denotes a trivial scheme. N and d are the numbers of all signers and rogue signers, respectively, and p is a prime number that satisfies $p > \max(2d^2 + 2d, \lceil \frac{\sqrt{2N}}{2} \rceil)$. $R_{\max}$ is the maximum number of rounds required to trace all the rogue signers. $\mathrm{Sig}_{\max}$ is the maximum number of signatures sent by the aggregator per round. TotalSig is the total number of signatures to trace all colluded rogue signers when there is a round to send $\mathrm{Sig}_{\max}$ signatures.

	$R_{\max}$	$\mathrm{Sig}_{\max}$	TotalSig
Trivial	d	N	$d(2N - d + 1)/2$
AS-FT-2	$d \log_2 N + d$	$2d + 1$	$3d(d + 1)$
AS-SW-1	$d^2 + d$	$p - 1$	$(d^2 + d)(p - 1)$

EUF-CMA security, $(c^2 + c)$-identifiabilty, and correctness.

5. Comparison of AS-SW-1 with AS-FT-2

In this section, we theoretically and experimentally evaluate the efficiency of AS-SW-1 compared to that of AS-FT-2. Then, we discuss which of the two schemes is more suitable in some situations.

5.1 Theoretical Evaluations

We theoretically evaluate a trivial scheme (i.e, no aggregation), an existing scheme AS-FT-2 and our proposed scheme AS-SW-1 with respect to the efficiency of the tracing functionality and the communication bandwidth of AS-FT-2 and AS-SW-1. To compare these metrics, we evaluate the maximum number of rounds required to trace all the rogue signers, the maximum number of signatures sent by the aggregator per round and the total number of signatures until all rogue signers are traced.

We firstly compare a trivial method and the other schemes. From Table 2, Trivial requires the least number of tracing rounds, while TotalSig is linear to N. When $d(2N - d + 1)/2 < 3d(d + 1)$, i.e, $N < 4d + 2$ or $d(2N - d + 1)/2 < (d^2 + d)(p - 1)$, i.e, $N < (d + 1)(p - 1) + (d - 1)/2$, Trivial is the best scheme in terms of both tracing and bandwidth, otherwise, it requires huge communication cost in the whole tracing.

Next, we compare AS-FT-2 and AS-SW-1. For the number of rounds $R_{\max}$, when $d < \log_2 N$, AS-SW-1 requires fewer rounds than AS-FT-2, otherwise AS-FT-2 requires fewer rounds than AS-SW-1. For $\mathrm{Sig}_{\max}$, when $2(d^2 + d) < \frac{\sqrt{2N}}{2}$ and $\frac{\sqrt{2N}}{2} - 1 < 2d + 1$, i.e., $8(d^2 + d)^2 < N < 8(d + 1)^2$, the maximum number of signatures sent by the aggregator per round of AS-SW-1 is less than that of AS-FT-2, otherwise AS-FT-2 sends less signatures than AS-SW-1. Therefore, AS-SW-1 traces more efficiently when the number of rogue signers is quite much smaller compared to the number of all signers but requires more bandwidth than AS-FT-2.

5.2 Implementation Evaluations

Based on the theoretical evaluations, we concretely set the number of all signers and that of the rogue signers, and implement our scheme, and evaluate which of AS-FT-2 or AS-SW-1 is more effective based on experiments. The details of the experimental environment are the same as in Sect. 3.

Table 3 Simulation results for AS-FT-2 and AS-SW-1. **non-FB** stands for no feedback ("✓" means it does not require a feedback, and "-" means it does), **Round** is the total number of rounds to trace all attackers, **Time** is the total time from the start of the first round to the completion of tracing, and **Sig** is the average size of transmitted signatures per round.

	N	d	p	non-FB	Round	Time [s]	Sig [kB]
AS-FT-2	1000	5	-	-	49.5	87.3	0.42
	1000	10	-	-	89.6	150.5	0.61
	1000	40	-	-	295.5	490.3	1.54
	3000	5	-	-	58.7	270.9	0.42
	3000	10	-	-	108.1	498.4	0.59
	3000	40	-	-	355.9	1676.7	1.62
AS-SW-1	1000	5	61	✓	28.7	58.4	4.70
	1000	10	223	✓	107.5	230.4	17.41
	1000	40	3299	✓	1640.0	3332.3	78.03
	3000	5	79	✓	29.1	138.7	6.11
	3000	10	223	✓	107.7	560.8	17.39
	3000	40	3299	✓	1640.0	6912.5	234.72

5.2.1 Simulation Settings

In the experiment setting, we capture a situation where a small number of fixed attackers always send invalid signatures to show the tracing efficiency and bandwidth consumption of AS-SW-1 by comparing with AS-FT-2. As in Sect. 3, the number N of all signers and the number d of rogue signers are $N = 1000, 3000$, and $d = 5, 10, 40$, respectively. The experiment is performed 10 times and the average is taken.

5.2.2 Simulation Results

Table 3 shows our result. In this table, the prime p is the smallest one satisfying the condition in Table 2 for AS-SW-1. The most important part is the column **non-FB**, which indicates the necessity of a feedback. We confirm that AS-SW-1, which does not require a feedback, indeed works well. For tracing efficiency, the total time of tracing is almost proportional to d^2 and N/p for AS-SW-1 when $p < N$, while that of AS-FT-2 is almost proportional to $d \log N$. This result is consistent with the theoretical evaluation shown in Table 2. From Table 3, the total time of tracing of AS-SW-1 is less than that of AS-FT-2 when $200d < N$, otherwise that of AS-FT-2 is less. On the other hand, for bandwidth consumption, from Table 2, the column **Sig** is almost proportional to d^2 for AS-SW-1 when $p < N$ and otherwise it is the same as individual signature transmission, while AS-FT-2 is almost proportional to d. Specifically, AS-SW-1 requires 144.9 times more signatures to trace rogue signers than AS-FT-2 when $(N, d) = (3000, 40)$. Observe that this result indicates that AS-SW-1 requires more bandwidth consumption than AS-FT-2.

5.2.3 Suitable Schemes for Applications

We consider which of AS-FT-2 and AS-SW-1 is more suitable for some applications, which are less time-constrained or real-time communication systems. In less time-constrained

communication systems, e.g., in monitoring sensor networks [15], a low-bandwidth protocol, such as LoRa, which has a speed of 37.5 kbps [30] is used. In addition, a system could have many rogue devices, say, from 40 to 300 rogue devices out of 1000 devices [27]. From the above results, AS-FT-2 is more efficient than AS-SW-1 in terms of the bandwidth and the time to trace all rogue signers when d is large. Although the aggregator of AS-FT-2 needs to wait for a feedback, as mentioned in Sect. 3, the waiting time is not too long for these applications, and the bandwidth is sufficient for AS-FT-2. Therefore, AS-FT-2 is more suitable for less time-constrained communication systems to take advantage of its bandwidth efficiency and short tracing time when d is large. In some real-time communication systems, e.g., cooperative intelligent transportation systems, some high bandwidth communication protocols, such as LTE (23.6 Mbps), are used. In addition, to optimize the overall efficiency of a system in features of safety indicators, many devices send data in short interval, which is about 200 ms [16]. Therefore, AS-FT-2 cannot be used because of the feedback waiting time. On the other hand, the above results show that AS-SW-1 is more efficient in terms of the time to trace all rogue signers when d is small ($d \leq 10$) due to its no feedback feature. However, when d is large, AS-SW-1 needs more tracing time and large bandwidth. Therefore, AS-SW-1 is more suitable for real-time communication systems to take advantage of its no feedback feature, acceptable bandwidth consumption and short tracing time when d is small.

6. Conclusion

In this paper, to examine whether an existing FT-AS scheme is capable of transmitting a feedback sufficiently fast on a real system, we evaluated the feedback waiting time of the implementation AS-FT-2 of an ASIT scheme proposed in [12]. The results of the implementation experiment indicates that if the acceptable feedback time of a system is significantly larger than a few hundred ms, e.g., industrial sensor systems [31], AS-FT-2 can be used without problems. On the other hand, there may be some applications where such sufficiently large feedback time is not acceptable, e.g., cooperative intelligent transportation systems [16]. In anticipation of such applications, we also proposed an ASIT scheme AS-SW-1 that does not require a feedback. The proposed scheme eliminates a feedback by using the STT proposed by Safavi-Naini et al. [17] in Ishii et al.'s generic construction of ASIT [12]. From our implementation results, we found that although a feedback is completely eliminated in our scheme, its communication cost is significantly larger, for example 144.9 times larger when $(N, d) = (3000, 40)$. Therefore, AS-SW-1 is more suitable for applications in which it is highly time-constrained and high bandwidth is available, while for applications where bandwidth is constrained but time is not, AS-FT-2 is more suitable.

We leave it as a future work to construct a concrete system that uses ASIT schemes. Towards this goal, we will first need to consider an execution environment such as the network topology and the message format for ASIT schemes.

References

[1] R. Ishii, K. Yamashita, Z. Song, Y. Sakai, T. Teruya, G. Hanaoka, K. Matsuura, and T. Matsumoto, "Constraints and evaluations on signature transmission interval for aggregate signatures with interactive tracing functionality," ESORICS 2022 Workshop on ADIoT, pp.51–71, 2022.

[2] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," EUROCRYPT 2003, pp.416–432, 2003.

[3] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham, "Sequential aggregate signatures from trapdoor permutations," EUROCRYPT 2004, pp.74–90, 2004.

[4] C. Gentry and Z. Ramzan, "Identity-based aggregate signatures," PKC 2006, Lecture Notes in Computer Science, vol.3958, pp.257–273, Springer, 2006.

[5] J.H. Ahn, M. Green, and S. Hohenberger, "Synchronized aggregate signatures: New definitions, constructions and applications," CCS 2010, pp.473–484, ACM, 2010.

[6] G. Hartung, B. Kaidel, A. Koch, J. Koch, and A. Rupp, "Fault-tolerant aggregate signatures," PKC 2016, pp.331–356, 2016.

[7] W.H. Kautz and R.C. Singleton, "Nonrandom binary superimposed codes," IEEE Trans. Inf. Theory, vol.10, no.4, pp.363–377, 1964.

[8] R. Kumar, S. Rajagopalan, and A. Sahai, "Coding constructions for blacklisting problems without computational assumptions," CRYPTO'99, pp.609–623, 1999.

[9] G.M. Zaverucha and D.R. Stinson, "Group testing and batch verification," ICITS 2009, pp.140–157, 2009.

[10] D. Du, F.K. Hwang, and F. Hwang, Combinatorial Group Testing and Its Applications, World Scientific, 2000.

[11] D. Eppstein, M.T. Goodrich, and D.S. Hirschberg, "Improved combinatorial group testing algorithms for real-world problem sizes," SIAM J. Comput., vol.36, no.5, pp.1360–1375, 2007.

[12] R. Ishii, K. Yamashita, Y. Sakai, T. Matsuda, T. Teruya, G. Hanaoka, K. Matsuura, and T. Matsumoto, "Aggregate signature with traceability of devices dynamically generating invalid signatures," ACNS 2021 Satellite Workshop on SCI, pp.378–396, 2021.

[13] J. Shikata, T. Matsumoto, and ECSEC, "Digital signature system and digital signature method," 2021. JP, 2021-077961, A, 2021-5-20 (In Japanese).

[14] A. Fiat and T. Tassa, "Dynamic traitor tracing," CRYPTO'99, pp.354–371, 1999.

[15] S. Suryavansh, A. Benna, C. Guest, and S. Chaterji, "A data-driven approach to increasing the lifetime of iot sensor nodes," Sci. Rep., vol.11, no.1, pp.1–12, 2021.

[16] S. Tak and S. Choi, "Safety monitoring system of CAVs considering the trade-off between sampling interval and data reliability," Sensors, vol.22, no.10, 3611, 2022.

[17] R. Safavi-Naini and Y. Wang, "Sequential traitor tracing," IEEE Trans. Inf. Theory, vol.49, no.5, pp.1319–1326, 2003.

[18] S. Sato, J. Shikata, and T. Matsumoto, "Aggregate signature with detecting functionality from group testing," IACR Cryptol. ePrint Arch., vol.2020, p.1219, 2020.

[19] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," ASIACRYPT 2001, pp.514–532, 2001.

[20] S. Hohenberger, A. Sahai, and B. Waters, "Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures," CRYPTO 2013, LNCS, vol.8042, pp.494–512, Springer, 2013.

[21] S. Sato and J. Shikata, "Interactive aggregate message authentication equipped with detecting functionality from adaptive group testing," IACR Cryptol. ePrint Arch., vol.2020, p.1218, 2020.

[22] G. Neven, "Efficient sequential aggregate signed data," EUROCRYPT 2008, N.P. Smart, ed., LNCS, vol.4965, pp.52–69, Springer, 2008.

[23] M. Gerbush, A.B. Lewko, A. O'Neill, and B. Waters, "Dual form sig-

natures: An approach for proving security from static assumptions," ASIACRYPT 2012, pp.25–42, 2012.
[24] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters, "Sequential aggregate signatures and multisignatures without random oracles," EUROCRYPT 2006, pp.465–485, 2006.
[25] K. Lee, D.H. Lee, and M. Yung, "Sequential aggregate signatures with short public keys without random oracles," Theor. Comput. Sci., vol.579, pp.100–125, 2015.
[26] Z. Song, R. Anzai, J. Sakamoto, N. Yoshida, and T. Matsumoto, "Proposal and prototype implementation of a cloud-based simulator for traceable aggregate signature protocol," SCIS 2022, 2022 (in Japanese).
[27] M. Pandey, S. Dhanoriya, and A. Bhagat, "Fast and efficient data acquisition in radiation affected large wsn by predicting transfaulty nodes," International Conference on Next Generation Computing Technologies, pp.246–262, Springer, 2017.
[28] S. Mitsunari, "mcl - a portable and fast pairing-based cryptography library," https://github.com/herumi/mcl, 2016.
[29] P.S.L.M. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order," SAC 2005, LNCS, vol.3897, pp.319–331, Springer, 2005.
[30] A. Lavric and V. Popa, "Performance evaluation of lorawan communication scalability in large-scale wireless sensor networks," Wireless Communications and Mobile Computing, vol.2018, 2018.
[31] M.P.R.S. Kiran and P. Rajalakshmi, "Performance analysis of CSMA/CA and PCA for time critical industrial iot applications," IEEE Trans. Ind. Inform., vol.14, no.5, pp.2281–2293, 2018.

Appendix A: Definition of DTT

Recall that DTT is a method for detecting piracy in digital content distribution services. Two security requirements are defined for DTT. One is R-identifiability, which means that the distributor can identify all the traitors within R rounds of executions of the tracing procedure. The other is completeness, which means that the distributor does not trace legitimate users. In DTT, it is assumed that at least one traitor redistributes in each segment. We first recall the syntax of DTT, then recall the security requirements.

Definition Appendix A.1 (Dynamic Traitor Tracing): A dynamic traitor tracing (DTT) consists of the two PPT algorithms (Initialize, Trace) that work as follows:

- $(\alpha, P) \leftarrow \mathsf{Initialize}(1^\lambda, 1^n)$: Algorithm Initialize takes 1^λ, 1^n as input and outputs a pair (α, P) of an internal state and a partition of the user set.
- $(\alpha', P', V) \leftarrow \mathsf{Trace}(\alpha, i)$: Algorithm Trace takes as input an internal state α and an index i, and outputs the tuple of an internal state of the next round, a partition of the user set, and a traced user set, where the input index is the index of the watermark assigned to the redistributed content from the traitor.

This syntax captures the following scenario. At first, the distributor executes Initialize to create an initial internal state and an initial partition. Note that a partition is a subset of the user set, and users who are in the same partition receive a content with the same watermark. Once the distributer detects a piracy, it checks the watermark i of the redistributed content, and runs Trace on i to create a new partition. The distributor repeats this until it detects a redistribution of a content whose watermark corresponds to a partition that contains only a single user. Then, the distributor regards the user as a pirate, and eliminates it. We count the interval between the executions of Trace as 1 *round*.

A.1 Security Requirements

Here, we recall the security requirements of DTT. In the security model, an adversary declares a set C (where $|C| \leq d$) of pirates at first. Furthermore, we assume that a pirate rebroadcasts the content that it receives. In other words, we do not consider the case where a pirate eavesdrops other user's content and rebroadcasts it. These are the restrictions implicitly put in [14].

As mentioned earlier, two security notions are considered for DTT: *R-identifiability*, which ensures that a distributor can identify all the pirates within R (or less) rounds, and *completeness*, which ensures that a distributor does not trace legitimate users as pirates. These are defined using the following experiment $\mathsf{ExpDTT}_{\Sigma_{\mathrm{DTT}},\mathcal{A}}(\lambda, n)$ in which a stateful adversary $\mathcal{A}$ is executed.

$\mathsf{ExpDTT}_{\Sigma_{\mathrm{DTT}},\mathcal{A}}(\lambda, n)$

$(\alpha_1, P_1) \leftarrow \Sigma_{\mathrm{DTT}}.\mathsf{Initialize}(1^\lambda, 1^n)$;
$C \leftarrow \mathcal{A}(\alpha_1, P_1); t := 1; W := \emptyset$; run $\mathcal{A}^{O_T(\cdot)}(\alpha_1, P_1)$:
Output (W, C)

where $\mathcal{A}$ may adaptively make multiple queries i_t to the tracing oracle O_T. However, $\mathcal{A}$'s t-th O_T query i_t must satisfy $i_t \in [|P_t|]$ and $S_{i_t,t} \cap C \neq \emptyset$, where α_t is the internal state and $P_t = (S_{1,t}, \ldots, S_{p_t,t})$ (for some natural number p_t) denotes the partition after $\mathcal{A}$'s $(t-1)$-th O_T query is answered. Given the t-th O_T query i_t from $\mathcal{A}$, O_T runs $(\alpha_{t+1}, P_{t+1}, V_t) \leftarrow \Sigma_{\mathrm{DTT}}.\mathsf{Trace}(\alpha_t, i_t)$, and returns $(\alpha_{t+1}, P_{t+1}, V_t)$ to $\mathcal{A}$. Then, O_T updates $W \leftarrow W \cup V_t$ and $t \leftarrow t+1$. We remark that W in the output of the experiment is that at the point $\mathcal{A}$ halts.

Definition Appendix A.2 (R-Identifiability): A DTT Σ_{DTT} satisfies R-identifiability if for any $\lambda \in \mathbb{N}$, any $n = \mathrm{poly}(\lambda)$, and any PPT adversary $\mathcal{A}$, it holds that

$$\Pr\left[C \not\subseteq W \mid (W, C) \leftarrow \mathsf{ExpDTT}_{\Sigma_{\mathrm{DTT}},\mathcal{A}}(\lambda, n) \wedge t \geq R\right] = \mathrm{negl}(\lambda).$$

where t is the value of the counter when $\mathcal{A}$ stops.

Definition Appendix A.3 (Completeness): A DTT Σ_{DTT} satisfies completeness if for any $\lambda \in \mathbb{N}$, any $n = \mathrm{poly}(\lambda)$, and any PPT adversary $\mathcal{A}$, it holds that

$$\Pr\left[([n] \setminus C) \cap W \neq \emptyset \mid (W, C) \leftarrow \mathsf{ExpDTT}_{\Sigma_{\mathrm{DTT}},\mathcal{A}}(\lambda, n)\right] = \mathrm{negl}(\lambda).$$

Appendix B: Proof of Lemma 4.3

Let Σ_{cSW1} be the algorithm described in Fig. 6. Observe that Σ_{cSW1} meets the syntax of DTT. Therefore, it remains to show the completeness and the R-identifiability, which can

be trivially obtained from Lemma 4.1 and Lemma 4.2.

Claim Appendix B.1: The algorithm Σ_{cSW1} satisfies the completeness of DTT.

Proof. Let (W, C) be the output of the experiment $\mathsf{ExpDTT}_{\Sigma_{\mathrm{cSW1}},\mathcal{A}}(\lambda, n)$ where $|C| = c$ and $\mathcal{A}$ is a PPT adversary, and let $u \in W$ (recall that W is the set of users that are detected as pirates). Observe that $u \in W$ if and only if $c + 1$ partitions[†] that u belongs to are detected as partitions that contain a piracy. Due to Lemma 4.1, $u \in W$ implies that $u \in C$, which concludes the claim. □

Claim Appendix B.2: Suppose that piracy occurs every round. Then, the algorithm Σ_{cSW1} satisfies $(c^2 + c)$-identifiability where c is the number of pirates.

Proof. Because piracy occurs every round, the tracing procedure of the underlying STT proceeds every round. Therefore, Lemma 4.2 concludes the claim. □

Ryu Ishii received the BE degree from Keio University and the ME degree from the University of Tokyo, in 2019 and 2021, respectively. Currently, he is a Ph.D. student at the University of Tokyo and a research assistant at the National Institute of Advanced Industrial Science and Technology (AIST), Japan.

Kyosuke Yamashita received the BE, ME and Ph.D. degrees from Kyoto University, in 2013, 2015 and 2021, respectively. Currently, he is an assistant professor at Osaka University, Japan. He received SCIS Paper Prize from IEICE in 2019.

Zihao Song received the BE degree from Okayama University and the ME degree from the Yokohama National University, in 2020 and 2022, respectively.

[†]Because we are now considering DTT, we interpret watermarks as partitions.

Yusuke Sakai received his B.E., M.E., and Ph.D. degrees from the University of Electro-Communications, Tokyo, Japan, in 2009, 2011, and 2014, respectively. From 2012 to 2014, and from 2014 to 2017, he was a research fellow of Japan Society for the Promotion of Science (JSPS). In 2017, he joined the National Institute of Advanced Industrial Science and Technology (AIST), Japan. He is presently engaged in research on cryptography. He received SCIS Paper Prize from IEICE in 2011 and the Best Student Award in IWSEC 2010.

Tadanori Teruya received the ME degree and the Ph.D. degree in engineering from the University of Tsukuba, Japan, in 2009 and 2012, respectively. He worked as a postdoctoral researcher in the Faculty of Engineering, Information and Systems, University of Tsukuba, Japan (2012–2013), and in the National Institute of Advanced Industrial Science and Technology (AIST), Japan (2013–2016), and then he worked as a researcher in AIST (2016–2018). He is currently a senior researcher in AIST since 2018. His research interests include cryptography, information security, and privacy-enhancing technologies, in particular, practical aspects of cryptography based on elliptic curves, lattices, and secure computation and their applications.

Takahiro Matsuda received his bachelors, masters, and Ph.D. degrees in Information and Communication Engineering from the University of Tokyo in 2006, 2008, and 2011, respectively. From 2009 to 2011 and from 2011 to 2013, he had been a Research Fellow of Japan Society for the Promotion of Science (JSPS). From 2011, he has been with the National Institute of Advanced Industrial Science and Technology (AIST), Japan, where he currently works as a Team Leader. His research interests are in the areas of public key cryptography and theory of cryptography.

Goichiro Hanaoka graduated from the Department of Engineering, the University of Tokyo in 1997. He received the Ph.D. degree from the University of Tokyo in 2002. Goichiro joined AIST in 2005. Currently, he is a Prime Senior Researcher, Cyber Physical Security Research Center, AIST. He engages in the R&D for encryption and information security technologies including the efficient design and security evaluation of public key cryptosystems. He has received numerous awards including the DoCoMo Mobile Science Award (2016), Mobile Communication Fund; the Wilkes Award (2007), British Computer Society; Best Paper Award (2008, 2019), The Institute of Electronics, Information and Communication Engineers (IEICE); and Innovative Paper Awards (2012, 2014), Symposium on Cryptography & Information Security (SCIS), IEICE.

Kanta Matsuura received his Ph.D. degree in electronics from the University of Tokyo in 1997. He is currently a Professor of Institute of Industrial Science at the University of Tokyo. His research interests include cryptography, cybersecurity, and security management such as security economics. He was an Associated Editor of IPSJ Journal (2001–2005) and IEICE Transactions on Communications (2005–2008), and won Distinguished-Service Award from the IEICE Communications Society in 2008. He was Editor-in-Chief of Security Management (2008–2012), and is an Editorial-Board member of Design, Codes, and Cryptography (2010–present). He is a fellow of IPSJ, and a senior member of IEEE, ACM, and IEICE. He is President of JSSM (Japan Society of Security Management) (2021–present). He is a member of Science Council of Japan (2017–present).

Tsutomu Matsumoto is a professor of the Faculty of Environment and Information Sciences, Yokohama National University. He also serves as the Director of the Cyber Physical Security Research Center (CPSEC) at the National Institute of Advanced Industrial Science and Technology (AIST). Starting from Cryptography in the early '80s, Prof. Matsumoto has opened up the field of security measuring for logical and physical security mechanisms. He received a Doctor of Engineering degree from the University of Tokyo in 1986. He serves as the chair of the Japanese National Body for ISO/TC68 (Financial Services) and the Cryptography Research and Evaluation Committees (CRYPTREC) and as an associate member of the Science Council of Japan (SCJ). He received the IEICE Achievement Award, the DoCoMo Mobile Science Award, the Culture of Information Security Award, the MEXT Prize for Science and Technology, and the Fuji Sankei Business Eye Award.