Lattice-based Multi-Entity Identification Protocols

Yohei Watanabe[1,2], Toi Tomita[3], and Junji Shikata[3,2]

[1] The University of Electro-Communications, Tokyo, Japan.
watanabe@uec.ac.jp
[2] Japan Datacom Co., Ltd., Tokyo, Japan.
[3] Yokohama National University, Yokohama, Japan.
{tomita-toi-sk, shikata-junji-rb}@ynu.ac.jp

Abstract

It is crucial for IoT networks to produce new methods to efficiently handle communications among multiple IoT devices. Aggregate MAC/signatures provide efficient multi-entity authentication protocols. However, a server cannot identify malicious entities, i.e., impersonated ones, though it can detect that there are some malicious ones. Recently, Hirose and Shikata introduced aggregate entity authentication protocols, which enable the server to simultaneously check the validity of multiple entities and identify malicious ones. Their aggregate entity authentication protocol is based on symmetric-key primitives, and hence it is lightweight. However, it requires key agreements between the server and entities beforehand. In this paper, we introduce multi-entity identification, which is a public-key analogy of aggregate entity authentication protocols, and propose two generic constructions. Since all the building blocks of our generic constructions can be instantiated from lattices, our constructions can be post-quantum ones.

Keywords: Aggregation, Entity authentication, Lattice-based cryptography

1 Introduction

1.1 Background

Internet-of-Things technologies have been spreading rapidly and enriching our lives. According to a Cisco report [1], tens of billions of IoT devices are expected to be deployed over the next few years. Therefore, it is crucial to figure out how we efficiently maintain and communicate with many IoT devices. Based on this motivation, Hirose and Shikata introduced aggregate entity authentication protocols [2, 3], which enable a server to authenticate many entities and identify invalid ones of them. It achieves more efficient communication costs than a naive solution, i.e., authenticating each entity individually. More specifically, Hirose and Shikata considered a scenario where an edge device plays the role of an *aggregator* and collects authentication information from each IoT device. Then, the aggregator performs efficient communications with the server. They proposed a generic construction of an aggregate entity authentication protocol from aggregate MACs [4] and a group-testing algorithm [5]. This construction only relies on symmetric-key cryptographic primitives and hence is lightweight, though the server needs to share secret keys with *all* entities.

1.2 Our Contributions

In this paper, we propose a *multi-entity identification protocol* as a public-key analogy of the aggregate entity authentication protocol [2, 3]. Specifically, we consider the same setting as in Hirose and Shikata's work and extend standard, canonical identification protocols [6, 7, 8] to define a model and security notions of multi-entity identification protocols. We propose two generic constructions of multi-entity identification protocols: one is aggregate signatures [9] and group-testing algorithms; the other is from digital signatures, non-interactive batch arguments (BARGs) [10], and group-testing algorithms. These primitives used in our constructions can be instantiated from lattice-based assumptions. Hence, we obtain post-quantum multi-entity identification protocols.

2 Preliminaries

2.1 Notations

For any integer $a \in \mathbb{Z}$, let $[a] := \{1, 2, \ldots, a\}$. For a finite set $\mathcal{X}$, we use $x \xleftarrow{\$} \mathcal{X}$ to represent processes of choosing an element x from X uniformly at random. For a finite set $\mathcal{X}$, we denote by $x \leftarrow \mathcal{X}$ and $|\mathcal{X}|$ the addition x to $\mathcal{X}$ and cardinality of $\mathcal{X}$, respectively. Concatenation is denoted by $\|$. In the description of the algorithm, all arrays, strings, and sets are initialized to empty ones. For any non-interactive algorithm A, $\mathsf{out} \leftarrow \mathsf{A}(\mathsf{in})$ means that A takes in as input and outputs out. We denote by $\mathsf{A}^{\mathsf{O}}(\cdot)$ A allowed access to an oracle O. In this paper, we consider two-party interactive algorithms between a sender and a receiver, and it is denoted by $(\mathsf{out}_\mathsf{S}; \mathsf{out}_\mathsf{R}) \leftarrow \mathsf{A}(\mathsf{in}_\mathsf{S}; \mathsf{in}_\mathsf{R})$, where in_S and in_R are input of the sender and receiver, respectively, and out_S and out_R are output of the sender and receiver, respectively. Throughout the paper, we denote by κ a security parameter and consider probabilistic polynomial-time algorithms (PPTAs). We say a function $\mathsf{negl}(\cdot)$ is negligible if for any polynomial $\mathsf{poly}(\cdot)$, there exists some constant $\kappa_0 \in \mathbb{N}$ such that $\mathsf{negl}(\kappa) < 1/\mathsf{poly}(\kappa)$ for all $\kappa \geq \kappa_0$.

2.2 Digital Signatures

A digital signature $\Pi_{\text{DS}} = (\mathsf{SSetup}, \mathsf{SKGen}, \mathsf{Sign}, \mathsf{SigVer})$ is defined as follows.

- $\mathsf{SSetup}(1^\kappa) \to \mathsf{pp}$: it takes a security parameter κ as input and outputs a public parameter pp. Suppose pp includes a message sapce $\mathcal{M}$.
- $\mathsf{SKGen}(\mathsf{pp}) \to (\mathsf{sigk}, \mathsf{verk})$: it takes the public parameter pp as input and outputs a pair of a signing key and verification key $(\mathsf{sigk}, \mathsf{verk})$.
- $\mathsf{Sign}(\mathsf{pp}, \mathsf{sigk}, \mathsf{m}) \to \sigma$: it takes the public parameter pp, a signing key sigk, and a message $\mathsf{m} \in \mathcal{M}$ as input and outputs a signature σ.
- $\mathsf{SigVer}(\mathsf{pp}, \mathsf{verk}, (\mathsf{m}, \sigma)) \to \mathsf{ans}$: it takes the public parameter pp, a verification key verk, and a pair of a message and a signature (m, σ) as input and outputs $\mathsf{ans} \in \{0, 1\}$.

Definition 1 (Correctness)**.** Let Π_{DS} be a digital signature scheme. Π_{DS} is said to meet correctness if for all $\kappa \in \mathbb{N}$, all $\mathsf{pp} \leftarrow \mathsf{SSetup}(1^\kappa)$, all $(\mathsf{sigk}, \mathsf{verk}) \leftarrow \mathsf{SKGen}(\mathsf{pp})$, all $\mathsf{m} \in \mathcal{M}$, $\mathsf{SigVer}(\mathsf{pp}, \mathsf{verk}, (\mathsf{m}, \mathsf{Sign}(\mathsf{pp}, \mathsf{sigk}, \mathsf{m}))) = 1$ holds with overwhelming probability.

We consider a standard security notion for digital signatures, called unforgeability against chosen message attacks ($\mathsf{UF\text{-}CMA}$). Specifically, we consider a $\mathsf{UF\text{-}CMA}$ game against a PPTA A in Fig. 1, and define A's advantage in the security game as $\mathsf{Adv}^{\mathsf{UF}}_{\Pi_{\text{DS}},\mathsf{A}}(\kappa) := \Pr[\mathsf{Exp}^{\mathsf{UF}}_{\Pi_{\text{DS}},\mathsf{A}}(\kappa) = 1]$.

Definition 2 ($\mathsf{UF\text{-}CMA}$)**.** Let Π_{DS} be a digital signature scheme. Π_{DS} is said to be $\mathsf{UF\text{-}CMA}$ secure if for sufficiently large $\kappa \in \mathbb{N}$ and any PPTA A, it holds $\mathsf{Adv}^{\mathsf{UF}}_{\Pi_{\text{DS}},\mathsf{A}}(\kappa) < \mathsf{negl}(\kappa)$.

Experiment: $\mathsf{Exp}^{\mathsf{UF}}_{\Pi_{\mathrm{DS}},\mathsf{A}}(1^\kappa)$

```
1: $\mathsf{pp} \leftarrow \mathsf{SSetup}(1^\kappa)$
2: $(\mathsf{sigk}^\star, \mathsf{verk}^\star) \leftarrow \mathsf{SKGen}(\mathsf{pp})$
3: $(\mathsf{m}^\star, \sigma^\star) \leftarrow \mathsf{A}^{\mathsf{O}_{\mathsf{Sign}}}(\mathsf{pp}, \mathsf{verk}^\star)$
4: if $\mathsf{m}^\star \notin \mathsf{SigList} \ \wedge\ \mathsf{SigVer}(\mathsf{pp}, \mathsf{verk}^\star, \mathsf{m}^\star, \sigma^\star) \to 1$ then
5:     return 1
6: else
7:     return 0
```

Figure 1: A UF-CMA game for a digital signature Π_{DS}. $\mathsf{O}_{\mathsf{Sign}}$ is a signing oracle that returns $\mathsf{Sign}(\mathsf{pp}, \mathsf{sigk}^\star, \mathsf{m})$ for any query $\mathsf{m} \in \mathcal{M}$ and adds m to $\mathsf{SigList}$.

Experiment: $\mathsf{Exp}^{\mathsf{UF}}_{\Pi_{\mathrm{AGGS}},\mathsf{A}}(1^\kappa)$

```
1: $\mathsf{pp} \leftarrow \mathsf{SSetup}(1^\kappa, 1^N)$
2: $(\vec{\mathsf{verk}}^\star = (\mathsf{verk}_i^\star)_{i\in[N^\star]}, \vec{\mathsf{m}}^\star = (\mathsf{m}_i^\star)_{i\in[N^\star]}, \sigma^\star_{\mathsf{agg}}) \leftarrow \mathsf{A}^{\mathsf{O}_{\mathsf{SKGen}},\mathsf{O}_{\mathsf{Crpt}},\mathsf{O}_{\mathsf{Sign}}}(\mathsf{pp})$
3: if $(N^\star \leq N) \wedge (\exists i \text{ s.t. } \mathsf{verk}_i^\star \notin \mathsf{CrList} \wedge \mathsf{m}_i^\star \notin \mathsf{SigList}) \wedge (\mathsf{AggVer}(\mathsf{pp}, \vec{\mathsf{verk}}^\star, \vec{\mathsf{m}}^\star, \sigma^\star_{\mathsf{agg}}) \to 1)$
   then
4:     return 1
5: else
6:     return 0
```

Figure 2: A UF-CAMA game for an aggregate signature Π_{AGGS}. $\mathsf{O}_{\mathsf{SKGen}}$ is a key-generation oracle that runs $(\mathsf{sigk}_i, \mathsf{verk}_i) \leftarrow \mathsf{SKGen}(\mathsf{pp})$ and returns verk_i for an i-th query. $\mathsf{O}_{\mathsf{Crpt}}$ is a corruption oracle that adds verk_i to CrList and returns sigk_i. $\mathsf{O}_{\mathsf{Sign}}$ is the same one as in Fig. 1.

2.3 Aggregate Signatures

Aggregate signatures [9, 11] enable the compression of multiple signatures to a single one only using public information. An aggregate signature scheme Π_{AGGS} consists of the same algorithms $(\mathsf{SSetup}, \mathsf{SKGen}, \mathsf{Sign}, \mathsf{SigVer})$ of a DS Π_{DS} (with a slight modification to SSetup) and the following algorithms $(\mathsf{Agg}, \mathsf{AggVer})$:

1) $\mathsf{SSetup}(1^\kappa, 1^N) \to \mathsf{pp}$: it takes a security parameter κ and an upper bound $N = \mathsf{poly}(\kappa) \in \mathbb{N}$ for compression as input and outputs a public parameter pp. Suppose pp includes a message sapce $\mathcal{M}$.
2) $\mathsf{Agg}(\mathsf{pp}, (\mathsf{verk}_i, \mathsf{m}_i, \sigma_i)_{i\in[N']}) \to \sigma_{\mathsf{agg}}$: it takes the public parameter pp, N' $(\leq N)$ verification keys verk_i, a message $\mathsf{m}_i \in \mathcal{M}$, a signature σ_i as input, and outputs an aggregated signature σ_{agg}.
3) $\mathsf{AggVer}(\mathsf{pp}, \vec{\mathsf{verk}}, \vec{\mathsf{m}}, \sigma_{\mathsf{agg}}) \to \mathsf{ans}$: it takes the public parameter pp, N' $(\leq N)$ verification keys verk, N' messages $\vec{\mathsf{m}} \in \mathcal{M}^{N'}$ and an aggregated signature σ_{agg} as input, and outputs $\mathsf{ans} \in \{0, 1\}$, where $\mathsf{ans} = 1$ and $\mathsf{ans} = 0$ mean acceptance and rejection, respectively.

The correctness of Π_{AGGS} is defined as follows.

Definition 3 (Correctness)**.** Let Π_{AGGS} be an aggregate signature scheme. Π_{AGGS} is said to meet correctness if for all $\kappa \in \mathbb{N}$, all $N = \mathsf{poly}(\kappa) \in \mathbb{N}$, all $\mathsf{pp} \leftarrow \mathsf{SSetup}(1^\kappa, 1^N)$, all $N' \leq N$, all

```
GT:  GTest($\mathcal{I}$)
1: $\mathcal{J}_0 \coloneqq \mathcal{I}$
2: for $i = 1, 2, \ldots, \ell$ do // Stage $i$
3:     $\mathcal{J}_i \coloneqq \mathcal{J}_{i-1}$
4:     for $j = 1, 2, \ldots, |\mathcal{G}_i|$ do
5:         if $\boldsymbol{g}_i^{(j)}$ is negative then
6:             $\mathcal{J}_i \leftarrow \mathcal{J}_i \setminus \mathcal{I}(\boldsymbol{g}_i^{(j)})$
7:     Determine $\mathcal{G}_{i+1}$ // determine tests in the next stage
8: return $\mathcal{J} \coloneqq \mathcal{J}_\ell$
```

Figure 3: A group-testing algorithm.

$i \in [N']$, $(\mathsf{sigk}_i, \mathsf{verk}_i) \leftarrow \mathsf{SKGen}(\mathsf{pp})$, all $\mathsf{m}_i \in \mathcal{M}$, all $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{sigk}_i, \mathsf{m}_i)$, it holds that

$$\Pr[\mathsf{SigVer}(\mathsf{pp}, \mathsf{verk}_i, \mathsf{m}_i, \sigma_i) \to 1] \geq 1 - \mathsf{negl}(\kappa),$$
$$\Pr[\mathsf{AggVer}(\mathsf{pp}, \vec{\mathsf{verk}}, \vec{\mathsf{m}}, \mathsf{Agg}(\mathsf{pp}, (\mathsf{verk}_i, \mathsf{m}_i, \sigma_i)_{i\in[N']})) \to 1] \geq 1 - \mathsf{negl}(\kappa),$$

where $\vec{\mathsf{verk}} \coloneqq (\mathsf{verk}_1, \ldots, \mathsf{verk}_{N'})$ and $\vec{\mathsf{m}} \coloneqq (\mathsf{m}_1, \ldots, \mathsf{m}_{N'})$.

We consider a security notion for aggregate signatures, called unforgeability against chosen aggregated message attacks (UF-CAMA). Specifically, we consider a UF-CAMA game against a PPTA A in Fig. 2, and define A's advantage in the security game as $\mathsf{Adv}^{\mathsf{UF}}_{\Pi_{\text{AGGS}},\mathsf{A}}(\kappa) \coloneqq \Pr[\mathsf{Exp}^{\mathsf{UF}}_{\Pi_{\text{AGGS}},\mathsf{A}}(\kappa) = 1]$.

Definition 4 (UF-CAMA)**.** Let Π_{AGGS} be an aggregate signature scheme. Π_{AGGS} is said to be UF-CAMA secure if for sufficiently large $\kappa \in \mathbb{N}$ and any PPTA A, it holds $\mathsf{Adv}^{\mathsf{UF}}_{\Pi_{\text{AGGS}},\mathsf{A}}(\kappa) < \mathsf{negl}(\kappa)$.

2.4 Group Testing

Suppose there are multiple items, say $\mathsf{id}_1, \mathsf{id}_2, \ldots, \mathsf{id}_n$, each of which is positive or negative. A *group testing* algorithm GT [5] examines more than one item and outputs negative if and only if all of them are negative, and enables one to identify all negative items with fewer tests than by examining one by one.

For n items $\mathcal{I} = \{\mathsf{id}_1, \mathsf{id}_2, \ldots, \mathsf{id}_n\}$, each test can be denoted by an n-bit vector $\boldsymbol{g} \in \{0,1\}^n$ such that $\boldsymbol{g}[j] = 1$ holds if and only if the test examines the j-th item. $\boldsymbol{g}$ outputs negative if and only if all items $\mathsf{id} \in \mathcal{I}(\boldsymbol{g})$ are negative, where $\mathcal{I}(\boldsymbol{g}) \coloneqq \{\mathsf{id}_j \mid \boldsymbol{g}[j] = 1\}$ is a set of items examined by a test $\boldsymbol{g}$. A group-testing algorithm is called non-adaptive if all the tests are determined beforehand. On the other hand, an adaptive group-testing algorithm allows one to determine the tests in the next stage after the tests in the current stage. A group testing algorithm GT can be denoted by a sequence of test sets $(\mathcal{G}_i \coloneqq \{\boldsymbol{g}_i^{(1)}, \boldsymbol{g}_i^{(2)}, \ldots, \boldsymbol{g}_i^{(|\mathcal{G}_i|)}\})_{i=1}^{\ell}$, and $\ell = 1$ if it is non-adaptive.

It is reasonable to assume $\mathbf{0}^n \notin \mathcal{G}$ and $\bigvee_{\boldsymbol{g}\in\mathcal{G}} \boldsymbol{g} = \mathbf{1}^n$ without loss of generality, where $\mathcal{G} \coloneqq \bigcup_{i=1}^{\ell} \mathcal{G}_i$ and $\vee$ denotes element-wise exclusive OR. We give a group-testing algorithm GT for $\mathcal{I}$ as an algorithm GTest with $(\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_\ell)$ in Fig. 3

Experiment: $\mathsf{Exp}^{\mathsf{CRS\text{-}IND}}_{\Pi_{\text{BARG}},\mathsf{A}}(1^\kappa, 1^N)$

```
1:  $(i^\star \in [N], \mathsf{state}) \leftarrow \mathsf{A}_1(1^\kappa, 1^N)$
2:  $b \xleftarrow{\$} \{0,1\}$
3:  if $b = 0$ then
4:      $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa, 1^N)$
5:  else
6:      $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{TrapSetup}(1^\kappa, 1^N, i^*)$
7:  $b' \leftarrow \mathsf{A}_2(\mathsf{state}, \mathsf{crs})$
8:  if $b' = b$ then
9:      return 1
10: else
11:     return 0
```

Experiment: $\mathsf{Exp}^{\mathsf{sExt}}_{\Pi_{\text{BARG}},\mathsf{A}}(1^\kappa, 1^N)$

```
1:  $(i^\star \in [N], \mathsf{state}) \leftarrow \mathsf{A}_1(1^\kappa, 1^N)$
2:  $(\mathsf{crs}^*, \mathsf{td}) \leftarrow \mathsf{TrapSetup}(1^\kappa, 1^N, i^*)$
3:  $(\vec{\mathsf{x}} := (\mathsf{x}_1, \ldots, \mathsf{x}_N), \pi) \leftarrow \mathsf{A}_2(\mathsf{crs}^*)$
4:  $\mathsf{w}^* \leftarrow \mathsf{Extract}(\mathsf{td}, \vec{\mathsf{x}}, \pi)$
5:  if $\mathsf{Verify}(\mathsf{crs}^*, \vec{\mathsf{x}}, \pi) = 1 \ \wedge \ (\mathsf{x}_{i^*}, \mathsf{w}^*) \notin \mathcal{R}$ then
6:      return 1
7:  else
8:      return 0
```

Figure 4: Gemes for CRS indistinguishability (left) and somewhere extractable in trapdoor mode (right).

We say GT is complete if the output $\mathcal{J}$ of GTest does not include any negative elements in $\mathcal{I}$. We say GT is sound if $\mathcal{J}$ includes all positive elements in $\mathcal{I}$. If each test $\boldsymbol{g}$ outputs the correct result, GT meets soundness but generally may not meet completeness.

Suppose that there are at most d positive items. There are several complete non-adaptive group-testing algorithms with $\mathcal{O}(d^2 \log n)$ tests [12, 13, 14] and complete adaptive algorithms with $\mathcal{O}(d \log (n/d))$ [15, 16].

2.5 Batch Arguments

For $\kappa \in \mathbb{N}$, let $x = \mathsf{poly}(\kappa) \in \mathbb{N}$ and $w = \mathsf{poly}'(\kappa) \in \mathbb{N}$. Let $\mathcal{R} \subset \{0,1\}^x \times \{0,1\}^w$ be an NP relation, and $\mathcal{L}_\mathcal{R} = \{\mathcal{L}_{\mathcal{R},\kappa} = \{\mathsf{x} \in \{0,1\}^x \mid \exists \mathsf{w} \in \{0,1\}^w : (\mathsf{x}, \mathsf{w}) \in \mathcal{R}\}\}$ be an NP language corresponding to $\mathcal{R}$ (we may simply write $\mathcal{L} = \{\mathcal{L}_\kappa\}$). For $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$, we call $\mathsf{x} \in \{0,1\}^x$ a statement and $\mathsf{w} \in \{0,1\}^w$ a witness of x.

A non-interactive batch argument (BARG) Π_{BARG} [10, 17] for an NP language $\mathcal{L}$ consists of three PPT algorithms $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ below.

1) $\mathsf{Setup}(1^\kappa, 1^N) \to \mathsf{crs}$: it takes a security parameter 1^κ and (the upper bound of) the number of instances $N = \mathsf{poly}(\kappa) \in \mathbb{N}$ as input and outputs a common reference string (CRS) crs.
2) $\mathsf{Prove}(\mathsf{crs}, \vec{\mathsf{x}}, \vec{\mathsf{w}}) \to \pi$: it takes the CRS crs, k $(\leq N)$ statements $\vec{\mathsf{x}} = (\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k)$, and their witnesses $\vec{\mathsf{w}} = (\mathsf{w}_1, \mathsf{w}_2, \ldots, \mathsf{w}_k)$ as input and outputs a proof π.
3) $\mathsf{Verify}(\mathsf{crs}, \vec{\mathsf{x}}, \pi) \to \mathsf{ans}$: it takes the CRS crs, k $(\leq N)$ statements $\vec{\mathsf{x}} = (\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k)$, and a proof π as input and outputs $\mathsf{ans} \in \{0,1\}$, where $\mathsf{ans} = 1$ and $\mathsf{ans} = 0$ indicate acceptance and rejection, respectively.

Completeness. We define the completeness property of BARG Π_{BARG} as follows.

Definition 5 (Completeness)**.** Let Π_{BARG} be a BARG for an NP language $\mathcal{L}$. Π_{BARG} is said to meet completeness if for $\kappa \in \mathbb{N}$, all $N = \mathsf{poly}(\kappa) \in \mathbb{N}$, $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa, 1^N)$, all $k \in [N]$, and all k pairs $(\mathsf{x}_i, \mathsf{w}_i)_{i \in [k]} \in \mathcal{R}^k$ of a statement and its witness, $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, (\mathsf{x}_i)_{i \in [k]}, (\mathsf{w}_i)_{i \in [k]})$, it holds $\mathsf{Verify}(\mathsf{crs}, (\mathsf{x}_i)_{i \in [k]}, \pi) = 1$ with overwhelming probability.

Somewhere argument of knowledge. We consider a special soundness property introduced by Choudhuri et al. [18], called *somewhere argument of knowledge*.

Definition 6 (Somewhere Argument of Knowledge [18])**.** Let Π_{BARG} be a BARG for an NP language $\mathcal{L}$. Π_{BARG} is said to be somewhere argument of knowledge if there exists a pair of PPT algorithms $(\mathsf{TrapSetup}, \mathsf{Extract})$ defined below:

4) $\mathsf{TrapSetup}(1^\kappa, 1^N, i^*) \to (\mathsf{crs}^*, \mathsf{td})$: it takes a security parameter 1^κ, (the upper bound of) the number of instances $N = \mathsf{poly}(\kappa) \in \mathbb{N}$, and an index $i^* \in [N]$ as input, and outputs a common reference string (CRS) crs and a trapdoor td;
5) $\mathsf{Extract}(\mathsf{td}, \vec{\mathsf{x}}, \pi) \to \mathsf{w}^*$: it takes the trapdoor td, k $(\leq N)$ statements $\vec{\mathsf{x}} = (\mathsf{x}_1, \mathsf{x}_2, \dots, \mathsf{x}_k)$, and a proof π as input, and outputs a witness w^*;

where $(\mathsf{TrapSetup}, \mathsf{Extract})$ satisifies the following two properties.

- *CRS indistinguishability*: Consider a PPT adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ and a left-side experiment in Fig. 4. BARG Π_{BARG} is said to meet CRS indistinguishability if for any PPT adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$, it holds $|\Pr[\mathsf{Exp}^{\mathsf{CRS\text{-}IND}}_{\Pi_{\text{BARG}},\mathsf{A}}(1^\kappa, 1^N) \to 1] - 1/2| \leq \mathsf{negl}(\kappa)$.
- *Somewhere extractable in trapdoor mode* Consider a PPT adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ and a right-side experiment in Fig. 4. BARG Π_{BARG} is said to be somewhere extractable in trapdoor mode if for any PPT adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$, it holds $\Pr[\mathsf{Exp}^{\mathsf{sExt}}_{\Pi_{\text{BARG}},\mathsf{A}}(1^\kappa, 1^N) \to 1] \leq \mathsf{negl}(\kappa)$.

3 Multi-Entity Identification Protocol

3.1 Model

We extend classical, canonical identification protocols [6, 7, 8] and introduce multi-entity identification protocol mID. We consider the same setting as the aggregate entity authentication protocol [2, 3]; there are a server, an aggregator, and multiple entities. Each entity sends information to be identified to the aggregator, and the aggregator and the server collaboratively identify *both valid and invalid entities.*

Formally, mID $\Sigma = (\mathsf{PG}, \mathsf{KG}, \mathsf{P}, \mathsf{V})$ is defined as follows.

1) $\mathsf{PG}(1^\kappa, 1^N) \to \mathsf{par}$: a probabilistic algorithm run by the server that takes a security parameter 1^κ and the maximum number $N = \mathsf{poly}(\kappa) \in \mathbb{N}$ of entities to be identified as unput and outputs a systems parameter par, which includes a challenge set $\mathcal{C}$.[1]
2) $\mathsf{KG}(\mathsf{id}) \to (\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}})$: a probabilistic algorithm run by entities that take an identifier id of an entity as input and outputs a public key $\mathsf{pk}_{\mathsf{id}}$ and secret key $\mathsf{sk}_{\mathsf{id}}$ for id.
3) $\mathsf{P} \coloneqq (\mathsf{P}_1, \mathsf{P}_2)$: an algorithm, which is run by entities, that consists of two sub-algorithms $\mathsf{P}_1, \mathsf{P}_2$ defined below:
   3-1) $\mathsf{P}_1(\mathsf{sk}_{\mathsf{id}}) \to (\mathsf{com}_{\mathsf{id}}, \mathsf{st}_{\mathsf{id}})$: a probabilistic algorithm that takes a secret key $\mathsf{sk}_{\mathsf{id}}$ as input and outputs a commitment $\mathsf{com}_{\mathsf{id}}$ and state information $\mathsf{st}_{\mathsf{id}}$.
   3-2) $\mathsf{P}_2(\mathsf{sk}_{\mathsf{id}}, \mathsf{com}_{\mathsf{id}}, \mathsf{ch}, \mathsf{st}_{\mathsf{id}}) \to \mathsf{res}_{\mathsf{id}}$: a deterministic algorithm that takes a secret key $\mathsf{sk}_{\mathsf{id}}$, a commitment $\mathsf{com}_{\mathsf{id}}$, a challenge $\mathsf{ch} \in \mathcal{C}$, and state information $\mathsf{st}_{\mathsf{id}}$ as input, and outputs a response $\mathsf{res}_{\mathsf{id}}$.
4) $\mathsf{V}(\{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}}, \mathsf{ch}; \{\mathsf{pk}_{\mathsf{id}}, \mathsf{res}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}}) \to (\widehat{\mathcal{I}}; \mathsf{ack})$: a deterministic interactive algorithm run between the server and the aggregator. The server-side algorithm takes public keys $\{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}}$

[1] All algorithms except for PG take par as input, so we omit it from the input of the algorithms.

Oracle: $\mathsf{Trans}_{\mathsf{par}}(\mathsf{id}, \mathsf{ch})$

```
1: if $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}})$ has not been generated then
2:    $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}}) \leftarrow \mathsf{KG}(\mathsf{par}, \mathsf{id})$
3: $(\mathsf{com}_{\mathsf{id}}, \mathsf{st}_{\mathsf{id}}) \leftarrow \mathsf{P}_1(\mathsf{sk}_{\mathsf{id}})$
4: if $\mathsf{ch} \notin \mathsf{ChList}$ then
5:    $\mathsf{ch} \xleftarrow{\$} \mathcal{C}$
6:    $\mathsf{ChList} \leftarrow \mathsf{ch}$
7: $\mathsf{res}_{\mathsf{id}} \leftarrow \mathsf{P}_2(\mathsf{sk}_{\mathsf{id}}, \mathsf{com}_{\mathsf{id}}, \mathsf{ch}, \mathsf{st}_{\mathsf{id}})$
8: return $(\mathsf{com}_{\mathsf{id}}, \mathsf{ch}, \mathsf{res}_{\mathsf{id}})$
```

Figure 5: $\mathsf{Trans}_{\mathsf{par}}$ oracle. ChList is a list of generated challenges.

Experiment: $\mathsf{Exp}^{\mathsf{cor}}_{\Sigma,\mathsf{A}}(1^\kappa, 1^N)$

```
1: $\mathsf{par} \leftarrow \mathsf{PG}(1^\kappa, 1^N)$
2: $\mathsf{id}^\star \leftarrow \mathsf{A}^{\mathsf{Trans}_{\mathsf{par}}, \mathsf{KG}_{\mathsf{par}}}(\mathsf{par})$
3: $(\mathsf{com}_{\mathsf{id}}, \mathsf{st}_{\mathsf{id}}) \leftarrow \mathsf{P}_1(\mathsf{sk}_{\mathsf{id}})$
4: $\mathsf{ch}^\star \xleftarrow{\$} \mathcal{C}$
5: $\mathsf{res}_{\mathsf{id}} \leftarrow \mathsf{P}_2(\mathsf{sk}_{\mathsf{id}}, \mathsf{com}_{\mathsf{id}}, \mathsf{ch}^\star, \mathsf{st}_{\mathsf{id}})$
6:
7: if $\mathsf{V}(\mathsf{pk}_{\mathsf{id}}, \mathsf{ch}^\star; \mathsf{pk}_{\mathsf{id}}, \mathsf{res}_{\mathsf{id}}) \rightarrow (\perp; \mathsf{ack})$ then
8:    return 1
9: else if $\mathsf{V}(\mathsf{pk}_{\mathsf{id}}, \mathsf{ch}^\star; \mathsf{pk}_{\mathsf{id}}, \mathsf{res}_{\mathsf{id}}) \rightarrow (\mathsf{id}; \mathsf{ack})$ then
10:    return 0
```

Figure 6: The correctness game. $\mathsf{KG}_{\mathsf{par}}$ is an oracle that takes id as input and returns $\mathsf{KG}(\mathsf{par}, \mathsf{id})$. Suppose $\mathsf{id}^\star$ is issued to O_{KG}.

of $\mathcal{I}$ and a challenge ch as input and the aggregator-side algorithm takes the public keys and responses $\{\mathsf{pk}_{\mathsf{id}}, \mathsf{res}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}}$ of $\mathcal{I}$ as input, and they interact with each other. Finally, the server-side algorithm outputs rejected identifiers $\widehat{\mathcal{I}} \subset \mathcal{I}$ of invalid entities; the aggregator-side algorithm outputs acknowledgment ack.

Suppose that a systems parameter par and each entity's key pair $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}})$ are correctly generated by PG and KG. Then, mID is executed as follows.

i. Each entity runs $\mathsf{P}_1(\mathsf{sk}_{\mathsf{id}})$ to generate a commitment $\mathsf{com}_{\mathsf{id}}$ and state information $\mathsf{st}_{\mathsf{id}}$, and sends the aggregator $\mathsf{com}_{\mathsf{id}}$.

ii. The aggregator broadcasts $\mathsf{ch} \in \mathcal{C}$ randomly chosen by the server to n $(\leq N)$ entities $(\mathsf{id}_1, \ldots, \mathsf{id}_n)$.

iii. Each entity receives ch and runs $\mathsf{P}_2(\mathsf{sk}_{\mathsf{id}}, \mathsf{com}_{\mathsf{id}}, \mathsf{ch}, \mathsf{st}_{\mathsf{id}})$ to obtain and send a response $\mathsf{res}_{\mathsf{id}}$ to the aggregator.

iv. The server and aggregator run $\mathsf{V}(\{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}}, \mathsf{ch}; \{\mathsf{pk}_{\mathsf{id}}, \mathsf{res}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}})$, and the server obtains a set of invalid entities' identifiers $\widehat{\mathcal{I}}$ (and accepts the rest of entities as valid ones).

To define the correctness property of the above model, we define an oracle $\mathsf{Trans}_{\mathsf{par}}$ that simulates interactions between valid entities and the server in Fig. 5. Trans takes as input entity's

Experiment: $\mathsf{Exp}^{\mathsf{imp\text{-}pa}}_{\Sigma,\mathsf{A}}(1^\kappa, 1^N)$

```
1:  $\mathsf{par} \leftarrow \mathsf{PG}(1^\kappa, 1^N)$
2:  $(\mathcal{I}^\star_\mathsf{h}, \mathcal{I}^\star_\mathsf{a}, \{\mathsf{com}_\mathsf{id}\}_{\mathsf{id}\in\mathcal{I}^\star_\mathsf{a}}, \mathsf{state}) \leftarrow \mathsf{A}_1^{\mathsf{Trans}_\mathsf{par}, \mathsf{O}_\mathsf{KG}, \mathsf{O}_\mathsf{Crpt}}(\mathsf{par})$
3:  for $\forall \mathsf{id} \in \mathcal{I}^\star_\mathsf{h}$ do
4:      $(\mathsf{com}_\mathsf{id}, \mathsf{st}_\mathsf{id}) \leftarrow \mathsf{P}_1(\mathsf{sk}_\mathsf{id})$
5:  $\mathsf{ch}^\star \overset{\$}{\leftarrow} \mathcal{C}$
6:  $(\widehat{\mathcal{I}}^\star_\mathsf{h}, \widehat{\mathcal{I}}^\star_\mathsf{a}, \{\mathsf{res}_\mathsf{id}\}_{\mathsf{id}\in\widehat{\mathcal{I}}^\star_\mathsf{a}}) \leftarrow \mathsf{A}_2^{\mathsf{Trans}_\mathsf{par}, \mathsf{O}_\mathsf{KG}, \mathsf{O}_\mathsf{Crpt}}(\mathsf{state}, \mathsf{ch}^\star, \{\mathsf{com}_\mathsf{id}\}_{\mathsf{id}\in\mathcal{I}^\star_\mathsf{h}})$
7:  for $\forall \mathsf{id} \in \widehat{\mathcal{I}}^\star_\mathsf{h}$ do
8:      $\mathsf{res}_\mathsf{id} \leftarrow \mathsf{P}_2(\mathsf{sk}_\mathsf{id}, \mathsf{com}_\mathsf{id}, \mathsf{ch}^\star, \mathsf{st}_\mathsf{id})$
9:  $\widetilde{\mathcal{I}}^\star := \emptyset$ // initialize a set of impersonated entities
10: for $\forall \mathsf{id} \in \widehat{\mathcal{I}}^\star_\mathsf{a} \setminus \mathcal{I}_\mathsf{Crpt}$ do
11:     if $(\mathsf{id}, \mathsf{ch}^\star) \notin \mathsf{TrList}$ then
12:         $\widetilde{\mathcal{I}}^\star \leftarrow \mathsf{id}$ // set id as an impersonated entity
13: $\mathsf{V}(\{\mathsf{pk}_\mathsf{id}\}_{\mathsf{id}\in\widehat{\mathcal{I}}^\star_\mathsf{h}\cup\widehat{\mathcal{I}}^\star_\mathsf{a}}, \mathsf{ch}^\star; \{\mathsf{pk}_\mathsf{id}, \mathsf{res}_\mathsf{id}\}_{\mathsf{id}\in\widehat{\mathcal{I}}^\star_\mathsf{h}\cup\widehat{\mathcal{I}}^\star_\mathsf{a}}) \rightarrow (\widehat{\mathcal{I}}^\star; \mathsf{ack})$
14: if $\widetilde{\mathcal{I}}^\star \not\subset \widehat{\mathcal{I}}^\star$ then // at least one entity is not identified
15:     return 1
16: else
17:     return 0
```

Figure 7: A security game against impersonation under passive attacks. It holds that $\widehat{\mathcal{I}}^\star_\mathsf{h} \subset \mathcal{I}^\star_\mathsf{h}$, $\mathcal{I}^\star_\mathsf{a} \subset \widehat{\mathcal{I}}^\star_\mathsf{a}$, and $\mathcal{I}^\star_\mathsf{h} \cup \mathcal{I}^\star_\mathsf{a} = \widehat{\mathcal{I}}^\star_\mathsf{h} \cup \widehat{\mathcal{I}}^\star_\mathsf{a}$. O_KG is an oracle that takes id as input, runs $(\mathsf{pk}_\mathsf{id}, \mathsf{sk}_\mathsf{id}) \leftarrow \mathsf{KG}(\mathsf{par}, \mathsf{id})$, returns pk_id. Without loss of generality, suppose that all $\mathsf{id} \in \mathcal{I}^\star_\mathsf{h} \cup \mathcal{I}^\star_\mathsf{a} (= \widehat{\mathcal{I}}^\star_\mathsf{h} \cup \widehat{\mathcal{I}}^\star_\mathsf{a})$ are issued to O_KG. O_Crpt is an oracle that takes id, which was previously issued to O_KG, as input, adds id to $\mathcal{I}_\mathsf{Crpt}$, and returns a stored secret key sk_id.

identifier id and a randomly-chosen challenge ch and outputs a transcript $(\mathsf{com}_\mathsf{id}, \mathsf{ch}, \mathsf{res}_\mathsf{id})$. Note that Trans reuses ch if it is previously generated.

The correctness property is defined with an experiment $\mathsf{Exp}^{\mathsf{cor}}_{\Sigma,\mathsf{A}}$ in Fig. 6.

Definition 7 (Correctness)**.** Let Σ be an mID protocol. Σ is said to be correct if for sufficiently large $\kappa \in \mathbb{N}$, all $N = \mathsf{poly}(\kappa) \in \mathbb{N}$, all PPT algorithm A, $\Pr[\mathsf{Exp}^{\mathsf{cor}}_{\Sigma,\mathsf{A}}(1^\kappa, 1^N) \rightarrow 1] \geq 1 - \mathsf{negl}(\kappa)$ holds.

3.2 Security Notions

Security against impersonation under passive attacks. As in the standard canonical identification protocols, we consider security against impersonation under passive attacks as a basic security notion for mID. This guarantees that the adversary provided valid transcripts between the honest aggregator and honest entities cannot impersonate the honest entities. This notion is defined with an experiment $\mathsf{Exp}^{\mathsf{imp\text{-}pa}}_{\Sigma,\mathsf{A}}$ in Fig. 7 as follows.

Definition 8 (Security against Impersonation under Passive Attacks)**.** Let Σ be mID. mID is said to be secure against impersonation under passive attacks if for sufficiently large $\kappa \in \mathbb{N}$, all $N = \mathsf{poly}(\kappa) \in \mathbb{N}$, all PPT algorithms A, $\Pr[\mathsf{Exp}^{\mathsf{imp\text{-}pa}}_{\Sigma,\mathsf{A}}(1^\kappa, 1^N) \rightarrow 1] \leq \mathsf{negl}(\kappa)$ holds.

Experiment: $\mathsf{Exp}^{\mathsf{z}}_{\Sigma,\mathsf{A}}(1^\kappa, 1^N)$

```
1:  $\mathsf{par} \leftarrow \mathsf{PG}(1^\kappa, 1^N)$
2:  $(\mathcal{I}^\star_{\mathsf{h}}, \mathcal{I}^\star_{\mathsf{a}}, \{\mathsf{com}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}^\star_{\mathsf{a}}}, \mathsf{state}) \leftarrow \mathsf{A}_1^{\mathsf{Trans}_{\mathsf{par}}, \mathsf{O}_{\mathsf{KG}}, \mathsf{O}_{\mathsf{Crpt}}}(\mathsf{par})$
3:  for $\forall \mathsf{id} \in \mathcal{I}^\star_{\mathsf{h}}$ do
4:      $(\mathsf{com}_{\mathsf{id}}, \mathsf{st}_{\mathsf{id}}) \leftarrow \mathsf{P}_1(\mathsf{sk}_{\mathsf{id}})$
5:  $\mathsf{ch}^\star \xleftarrow{\$} \mathcal{C}$
6:  $(\widehat{\mathcal{I}}^\star_{\mathsf{h}}, \widehat{\mathcal{I}}^\star_{\mathsf{a}}, \{\mathsf{res}_{\mathsf{id}}\}_{\mathsf{id}\in\widehat{\mathcal{I}}^\star_{\mathsf{a}}}) \leftarrow \mathsf{A}_2^{\mathsf{Trans}_{\mathsf{par}}, \mathsf{O}_{\mathsf{KG}}, \mathsf{O}_{\mathsf{Crpt}}}(\mathsf{state}, \mathsf{ch}^\star, \{\mathsf{com}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}^\star_{\mathsf{h}}})$
7:  for $\forall \mathsf{id} \in \widehat{\mathcal{I}}^\star_{\mathsf{h}}$ do
8:      $\mathsf{res}_{\mathsf{id}} \leftarrow \mathsf{P}_2(\mathsf{sk}_{\mathsf{id}}, \mathsf{com}_{\mathsf{id}}, \mathsf{ch}^\star, \mathsf{st}_{\mathsf{id}})$
9:  $\mathsf{V}(\{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\widehat{\mathcal{I}}^\star_{\mathsf{h}}\cup\widehat{\mathcal{I}}^\star_{\mathsf{a}}}, \mathsf{ch}^\star; \{\mathsf{pk}_{\mathsf{id}}, \mathsf{res}_{\mathsf{id}}\}_{\mathsf{id}\in\widehat{\mathcal{I}}^\star_{\mathsf{h}}\cup\widehat{\mathcal{I}}^\star_{\mathsf{a}}}) \rightarrow (\widehat{\mathcal{I}}^\star; \mathsf{ack})$
10: $\mathcal{I}^\star_{\mathsf{comp}} := \emptyset$ and $\mathcal{I}^\star_{\mathsf{snd}} := \emptyset$
11: for $\forall \mathsf{id} \in \widehat{\mathcal{I}}_{\mathsf{h}} \cup \widehat{\mathcal{I}}_{\mathsf{a}}$ do
12:     if $\mathsf{V}(\mathsf{pk}_{\mathsf{id}}, \mathsf{ch}^\star; \mathsf{pk}_{\mathsf{id}}, \mathsf{res}_{\mathsf{id}}) \rightarrow (\perp; \mathsf{ack})$ then
13:         $\mathcal{I}^\star_{\mathsf{comp}} \leftarrow \mathsf{id}$ // set id as an accepted entity
14:     else if $\mathsf{V}(\mathsf{pk}_{\mathsf{id}}, \mathsf{ch}^\star; \mathsf{pk}_{\mathsf{id}}, \mathsf{res}_{\mathsf{id}}) \rightarrow (\mathsf{id}; \mathsf{ack})$ then
15:         $\mathcal{I}^\star_{\mathsf{snd}} \leftarrow \mathsf{id}$ // set id as a malicious entity
16: if $(\mathsf{z} = \mathsf{comp}) \wedge (\mathcal{I}^\star_{\mathsf{comp}} \cap \widehat{\mathcal{I}}^\star \neq \emptyset)$ then // at least one accepted entity is rejected
17:     return 1
18: else if $(\mathsf{z} = \mathsf{snd}) \wedge (\mathcal{I}^\star_{\mathsf{snd}} \setminus \widehat{\mathcal{I}}^\star \neq \emptyset)$ then // at least one malicious entity is accepted
19:     return 1
20: else
21:     return 0
```

Figure 8: A game for completeness and soundness. It hold $\widehat{\mathcal{I}}^\star_{\mathsf{h}} \subset \mathcal{I}^\star_{\mathsf{h}}$, $\mathcal{I}^\star_{\mathsf{a}} \subset \widehat{\mathcal{I}}^\star_{\mathsf{a}}$, and $\mathcal{I}^\star_{\mathsf{h}} \cup \mathcal{I}^\star_{\mathsf{a}} = \widehat{\mathcal{I}}^\star_{\mathsf{h}} \cup \widehat{\mathcal{I}}^\star_{\mathsf{a}}$. O_{KG} and $\mathsf{O}_{\mathsf{Crpt}}$ are the same oracles as in Fig. 7.

Note that if A outputs $(\mathcal{I}^\star_{\mathsf{h}}, \mathcal{I}^\star_{\mathsf{a}})$ such that $|\mathcal{I}^\star_{\mathsf{h}}| = 0$ and $|\mathcal{I}^\star_{\mathsf{a}}| = 1$ hold, the security game in Fig. 7 is equivalent to the security game for the standard canonical identification protocol (except for thy existence of the aggregator).

Completeness and soundness. The completeness property guarantees that no false positive occurs, i.e., the server does not identify valid entities as invalid ones, while the soundness property guarantees that no false negative occurs, i.e., the server does not identify invalid entities as valid ones. Completeness and soundness are defined with an experiment $\mathsf{Exp}^{\mathsf{z}}_{\Sigma,\mathsf{A}}$ in Fig. 8 as follows.

Definition 9 (Completeness)**.** Let Σ be mID. mID is said to be complete if for sufficiently large $\kappa \in \mathbb{N}$, all $N = \mathsf{poly}(\kappa) \in \mathbb{N}$, all PPT algorithms A, $\Pr[\mathsf{Exp}^{\mathsf{comp}}_{\Sigma,\mathsf{A}}(1^\kappa, 1^N) \rightarrow 1] \leq \mathsf{negl}(\kappa)$ holds.

Definition 10 (Soundness)**.** Let Σ be mID. mID is said to be sound if for sufficiently large $\kappa \in \mathbb{N}$, all $N = \mathsf{poly}(\kappa) \in \mathbb{N}$, all PPT algorithms A, $\Pr[\mathsf{Exp}^{\mathsf{snd}}_{\Sigma,\mathsf{A}}(1^\kappa, 1^N) \rightarrow 1] \leq \mathsf{negl}(\kappa)$ holds.

4 Proposed Constructions

We propose two mID protocols based on the aggregate entity authentication protocol [2, 3] and Waters and Wu's aggregate signature scheme [17]. Both constructions can be instantiated as

```
mID Σ: $\mathsf{PG}(1^\kappa, 1^N)$
1: $\mathsf{pp} \leftarrow \mathsf{SSetup}(1^\kappa)$
2: return $\mathsf{par} := \mathsf{pp}$
```

```
mID Σ: $\mathsf{KG}(\mathsf{par}, \mathsf{id})$
1: $(\mathsf{sigk}_{\mathsf{id}}, \mathsf{verk}_{\mathsf{id}}) \leftarrow \mathsf{SKGen}(\mathsf{pp})$
2: return $(\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}) := ((\mathsf{sigk}_{\mathsf{id}}, \mathsf{id}), (\mathsf{verk}_{\mathsf{id}}, \mathsf{id}))$
```

```
mID Σ: $\mathsf{P}_1(\mathsf{sk}_{\mathsf{id}})$
1: return $(\mathsf{com}_{\mathsf{id}}, \mathsf{st}_{\mathsf{id}}) := (\perp, \perp)$
```

```
mID Σ: $\mathsf{P}_2(\mathsf{sk}_{\mathsf{id}}, \mathsf{com}_{\mathsf{id}}, \mathsf{ch}, \mathsf{st}_{\mathsf{id}})$
1: $\sigma_{\mathsf{id}} \leftarrow \mathsf{Sign}(\mathsf{sigk}_{\mathsf{id}}, \mathsf{ch})$
2: return $\mathsf{res}_{\mathsf{id}} := (\mathsf{id}, \sigma_{\mathsf{id}})$
```

```
mID Σ: $\mathsf{V}(\{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}}, \mathsf{ch} \in \{0,1\}^{\mathsf{poly}(\kappa)}; \{\mathsf{pk}_{\mathsf{id}}, \mathsf{res}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}})$
Server & Aggregator:
1: Aggregator sends $\mathcal{I}$ to Server // suppose $\mathcal{I} = (\mathsf{id}_1, \ldots, \mathsf{id}_n)$ and $n \le N$
2: if $n = 1$, i.e., $\mathcal{I} = \{\mathsf{id}\}$ then
3:     if Server runs $\mathsf{SigVer}(\mathsf{verk}_{\mathsf{id}}, \mathsf{ch}, \sigma_{\mathsf{id}}) = 1$ then
4:         return $\widehat{\mathcal{I}} := \emptyset$
5:     else
6:         return $\widehat{\mathcal{I}} := \mathsf{id}$
7: Server sends $\mathsf{ch}$ to Aggregator
8: Set $\mathcal{J}_0 := \mathcal{I}$ and $\mathsf{ch}_{\mathsf{id}} := \mathsf{ch}$ // for notational convenience
9: for $i = 1, 2, \ldots, \ell$ do (to line 20)
10:     Determine $\mathcal{G}_i$ based on $\mathsf{ans}_1, \ldots, \mathsf{ans}_{i-1}$
Aggregator:
11:     for $j = 1, 2, \ldots, |\mathcal{G}_i|$ do
12:         $\mathsf{Agg}(\mathsf{pp}, (\mathsf{verk}_{\mathsf{id}}, \mathsf{ch}_{\mathsf{id}}, \sigma_{\mathsf{id}})_{\mathsf{id}\in\mathcal{I}(\boldsymbol{g}_i^{(j)})}) \rightarrow \sigma_i^{(j)}$
13:     Send $(\sigma_i^{(1)}, \ldots, \sigma_i^{(|\mathcal{G}_i|)})$ to Server
Server:
14:     $\mathcal{J}_i := \mathcal{J}_{i-1}$
15:     for $j = 1, 2, \ldots, |\mathcal{G}_i|$ do
16:         $\mathsf{AggVer}(\mathsf{pp}, (\mathsf{verk}_{\mathsf{id}})_{\mathsf{id}\in\mathcal{I}(\boldsymbol{g}_i^{(j)})}, (\mathsf{ch}_{\mathsf{id}})_{\mathsf{id}\in\mathcal{I}(\boldsymbol{g}_i^{(j)})}, \sigma_i^{(j)}) \rightarrow \mathsf{ans}_i^{(j)}$
17:         if $\mathsf{ans}_i^{(j)} = 1$ then
18:             $\mathcal{J}_i \leftarrow \mathcal{J}_i \setminus \mathcal{I}(\boldsymbol{g}_i^{(j)})$
19:     Send $\mathsf{ans}_i := (\mathsf{ans}_i^{(1)}, \ldots, \mathsf{ans}_i^{(|\mathcal{G}_i|)})$ to Aggregator
Server & Aggregator:
20: end for (from line 9)
Server:
21: return $\widehat{\mathcal{I}} := \mathcal{J}_\ell$
Aggregator:
22: return ack
```

Figure 9: An mID Σ from an aggregate signature scheme Π_{AGGS} and a group-testing algorithm GT.

post-quantum protocols.

4.1 Construction from Aggregate Signatures and Group Testing

We propose an mID protocol from an aggregate signature $\Pi_{\text{AGGS}} = (\mathsf{SSetup}, \mathsf{SKGen}, \mathsf{Sign}, \mathsf{SigVer}, \mathsf{Agg}, \mathsf{AggVer})$ and a group-testing algorithm GT. We formally describe our construction in Fig. 9.

Theorem 1. If Π_{AGGS} meets correctness, our mID protocol meets correctness.

Proof. It is straightforward, so we only describe an overview. A key pair $(\mathsf{sigk}_{\mathsf{id}}, \mathsf{verk}_{\mathsf{id}})$ of the underlying aggregate signature is generated independent of the underlying identifier id. Therefore, regardless of A's choice of $\mathsf{id}^\star$, a signature $\sigma_{\mathsf{id}^\star}$ is generated from $\mathsf{sigk}_{\mathsf{id}^\star}$ and a challenge ch. Therefore, breaking the correctness property defined in Definition 7 means that breaking the correctness property of Π_{AGGS} in Definition 3. $\square$

Theorem 2. If Π_{AGGS} meets $\mathsf{UF\text{-}CAMA}$ security, our mID protocol is secure against impersonation under passive attacks.

Proof. We construct an adversary $\mathcal{F}$ that breaks $\mathsf{UF\text{-}CAMA}$ security of Π_{AGGS} by using an adversary A that breaks the security against impersonation under passive attacks. Suppose that $\mathcal{F}$ receives $\mathsf{verk}^\star$ from the challenger of the $\mathsf{UF\text{-}CAMA}$ game. First, $\mathcal{F}$ randomly chooses an index $i^\star$. Let $\mathsf{id}^\star$ be an identifier issued at $i^\star$-th query to O_{KG}. Receiving a query id, $\mathcal{F}$ simulates O_{KG} as follows: if it is $i^\star$-th query, then $\mathcal{F}$ returns $\mathsf{verk}^\star$; otherwise, $\mathcal{F}$ computes $(\mathsf{sigk}_{\mathsf{id}}, \mathsf{verk}_{\mathsf{id}}) \leftarrow \mathsf{SKGen}(\mathsf{pp})$, stores $\mathsf{sigk}_{\mathsf{id}}$, and returns verk. Receiving a query id, $\mathcal{F}$ simulates $\mathsf{O}_{\mathsf{Crpt}}$ by returning the stored signing key $\mathsf{sigk}_{\mathsf{id}}$. If $\mathsf{id}^\star$ is issued, $\mathcal{F}$ aborts the game and output a random bit. Receiving a query $(\mathsf{id}, \mathsf{ch})$, $\mathcal{F}$ simulates $\mathsf{Trans}_{\mathsf{par}}$ as follows: if $\mathsf{id} = \mathsf{id}^*$, $\mathcal{F}$ issues a query ch to $\mathsf{O}_{\mathsf{Sign}}$ to get σ, and simulates the rest of $\mathsf{Trans}_{\mathsf{par}}$ with σ; otherwise, $\mathcal{F}$ has the corresponding signing key $\mathsf{sigk}_{\mathsf{id}}$ and can simulate $\mathsf{Trans}_{\mathsf{par}}$. After A submits $(\widehat{\mathcal{I}}_{\mathsf{h}}^\star, \widehat{\mathcal{I}}_{\mathsf{a}}^\star, \{\mathsf{res}_{\mathsf{id}}\}_{\mathsf{id} \in \widehat{\mathcal{I}}_{\mathsf{a}}^\star})$, $\mathcal{F}$ computes $\mathsf{res}_{\mathsf{id}}$ for $\mathsf{id} \in \widehat{\mathcal{I}}_{\mathsf{h}}^\star$, and runs $\mathsf{V}(\{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id} \in \widehat{\mathcal{I}}_{\mathsf{h}}^\star \cup \widehat{\mathcal{I}}_{\mathsf{a}}^\star}, \mathsf{ch}^\star; \{\mathsf{pk}_{\mathsf{id}}, \mathsf{res}_{\mathsf{id}}\}_{\mathsf{id} \in \widehat{\mathcal{I}}_{\mathsf{h}}^\star \cup \widehat{\mathcal{I}}_{\mathsf{a}}^\star})$ to obtain $\widehat{\mathcal{I}}^\star$. If $\mathsf{id}^\star \notin \widetilde{\mathcal{I}}^\star \setminus \widehat{\mathcal{I}}^\star$, $\mathcal{F}$ aborts the game and outputs a random bit. Otherwise, there must be an aggregated signature σ_{agg} that breaks $\mathsf{UF\text{-}CAMA}$ security, and therefore $\mathcal{F}$ submits it to the challenger of $\mathsf{UF\text{-}CAMA}$ game. Note that the probability that $\mathcal{F}$ correctly guesses $i^\star$ is at least $1/N$. $\square$

Theorem 3. If Π_{DS} meets correctness and GT is complete, then our mID protocol satisfies completeness.

Proof. We omit the proof since it is straightforward. $\square$

Theorem 4. If Π_{AGGS} meets $\mathsf{UF\text{-}CAMA}$ security and GT is sound, then our mID protocol satisfies soundness.

Proof. In the experiment $\mathsf{Exp}^{\mathsf{snd}}_{\Sigma,\mathsf{A}}(1^\kappa, 1^N)$, at least one malicious entity is accepted if and only if (1) GT correctly works but some (aggregated) signatures are successfully forged; or (2) all (aggregated) signatures are valid but a soundness error occurs in GT. The situation (1) never occurs since Π_{AGGS} meets $\mathsf{UF\text{-}CAMA}$ security, and as in the proof of Theorem 2, we can construct an adversary $\mathcal{F}$ that breaks $\mathsf{UF\text{-}CAMA}$ security of Π_{AGGS} by using an adversary A that breaks soundness under the situation (1). The situation (2) also never occurs since GT is sound. $\square$

On post-quantum instantiations and their efficiency. We can obtain lattice-based mID protocol by instantiating the proposed mID protocol with post-quantum aggregate signatures, e.g., [19]. The concrete mID protocol can be instantiated from Tomita and Shikata's aggregate signature [19] that achieves the logarithmic aggregate signature size, i.e., $\mathcal{O}(\log n)$, and the complete adaptive group-testing algorithm with $\mathcal{O}(d \log{(n/d)})$ tests [16], and it requires $\mathcal{O}(d \log{(n/d)} \log n)$ communication costs. Therefore, our instantiation is more efficient than a naive solution, i.e., parallel executions of canonical identification protocols for each entity, which requires $\mathcal{O}(n)$ communication costs.

```
mID Σ:  V({pk_id}_{id∈I}, ch ∈ {0,1}^{poly(κ)}; {pk_id, res_id}_{id∈I})
Server & Aggregator:
 1: Aggregator sends $\mathcal{I}$ to Server // suppose $\mathcal{I} = (\mathsf{id}_1, \ldots, \mathsf{id}_n)$ and $n \leq N$
 2: Server sends $\mathsf{ch}$ to Aggregator
 3: Set $\mathcal{J}_0 := \mathcal{I}$ and $\mathsf{ch}_{\mathsf{id}} := \mathsf{ch}$ // for notational convenience
 4: for $i = 1, 2, \ldots, \ell$ do (to line 18)
 5:   Determine $\mathcal{G}_i$ based on $\mathsf{ans}_1, \ldots, \mathsf{ans}_{i-1}$
Aggregator:
 6:   for $j = 1, 2, \ldots, |\mathcal{G}_i|$ do
 7:     Convert $(\mathsf{verk}_{\mathsf{id}}, \mathsf{ch}_{\mathsf{id}})_{\mathsf{id} \in \mathcal{I}(\boldsymbol{g}_i^{(j)})}$ to NP statements $\vec{\mathsf{x}}_i^{(j)} := (\mathsf{x}_{\mathsf{id}})_{\mathsf{id} \in \mathcal{I}(\boldsymbol{g}_i^{(j)})}$
 8:     Convert $(\sigma_{\mathsf{id}})_{\mathsf{id} \in \mathcal{I}(\boldsymbol{g}_i^{(j)})}$ to witnesses $\vec{\mathsf{w}}_i^{(j)} := (\mathsf{w}_{\mathsf{id}})_{\mathsf{id} \in \mathcal{I}(\boldsymbol{g}_i^{(j)})}$
 9:     $\mathsf{Prove}(\mathsf{crs}, \vec{\mathsf{x}}_i^{(j)}, \vec{\mathsf{w}}_i^{(j)}) \to \pi_i^{(j)}$
10:   Send $(\pi_i^{(1)}, \ldots, \pi_i^{(|\mathcal{G}_i|)})$ to Server
Server:
11:   $\mathcal{J}_i := \mathcal{J}_{i-1}$
12:   for $j = 1, 2, \ldots, |\mathcal{G}_i|$ do
13:     Convert $(\mathsf{verk}_{\mathsf{id}}, \mathsf{ch}_{\mathsf{id}})_{\mathsf{id} \in \mathcal{I}(\boldsymbol{g}_i^{(j)})}$ to NP statements $\vec{\mathsf{x}}_i^{(j)} := (\mathsf{x}_{\mathsf{id}})_{\mathsf{id} \in \mathcal{I}(\boldsymbol{g}_i^{(j)})}$
14:     $\mathsf{Verify}(\mathsf{crs}, \vec{\mathsf{x}}_i^{(j)}, \pi_i^{(j)}) \to \mathsf{ans}_i^{(j)}$
15:     if $\mathsf{ans}_i^{(j)} = 1$ then
16:       $\mathcal{J}_i \leftarrow \mathcal{J}_i \setminus \mathcal{I}(\boldsymbol{g}_i^{(j)})$
17:   Send $\mathsf{ans}_i := (\mathsf{ans}_i^{(1)}, \ldots, \mathsf{ans}_i^{(|\mathcal{G}_i|)})$ to Aggregator
Server & Aggregator:
18: end for (from line 4)
Server:
19: return $\widehat{\mathcal{I}} := \mathcal{J}_\ell$
Aggregator:
20: return ack
```

Figure 10: An mID Σ from a digital signature scheme Π_{DS}, a BARG Π_{BARG}, and a group-testing algorithm GT. $\mathsf{PG}(1^\kappa, 1^N)$ runs $\mathsf{pp} \leftarrow \mathsf{SSetup}(1^\kappa, 1^N)$ and $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa, 1^N)$ and returns $\mathsf{par} := (\mathsf{pp}, \mathsf{crs})$. KG, P_1, and P_2 are the same as those in Fig. 9.

4.2 Construction from Digital Signatures, BARGs, and Group Testing

We can break down the primitive used in the first construction based on Waters and Wu's aggregate signature scheme constructed from digital signatures and BARGs [17]. We show a variant of the construction in the previous section from a digital signature $\Pi_{\mathrm{DS}} = (\mathsf{SSetup}, \mathsf{SKGen}, \mathsf{Sign}, \mathsf{SigVer})$a BARG $\Pi_{\mathrm{BARG}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ for an NP language $\mathcal{L} = \{\mathcal{L}_\kappa := \{(\mathsf{verk}, \mathsf{m}) \mid \exists \pi \in \{0,1\}^{\mathsf{poly}(\kappa)} \text{ s.t. } \mathsf{SigVer}(\mathsf{verk}, \mathsf{m}, \pi) = 1\}\}_{\kappa \in \mathbb{N}}$, and a group-testing algorithm GT. We formally describe our construction in Fig. 10, and omit the proofs since they follow from the security proofs of our mID protocol in the previous section and Waters and Wu's aggregate signature scheme.

Theorem 5. If Π_{DS} meets correctness, our mID protocol meets correctness.

Theorem 6. If Π_{DS} meets $\mathsf{UF\text{-}CMA}$ security and Π_{BARG} is somewhere argument of knowledge, our mID protocol is secure against impersonation under passive attacks.

Theorem 7. If Π_{DS} meets correctness, Π_{BARG} meets completeness, and GT is complete, then our mID protocol satisfies completeness.

Theorem 8. If Π_{DS} meets $\mathsf{UF\text{-}CMA}$ security, Π_{BARG} is somewhere arugment of knowledge, and GT is sound, then our mID protocol satisfies soundness.

On post-quantum instantiations and their efficiency. We can obtain lattice-based mID protocol by instantiating the proposed mID protocol with post-quantum constructions of digital signatures, such as Dilithium [20, 21] and Falcon [22], and BARGs [18, 23, 24, 25]. Dilithium and Falcon are quite efficient, and the recent BARG constructions [18, 23, 24, 25] achieve logarithmic proof sizes in the number of statements, i.e., $\mathcal{O}(\log n)$. The concrete mID protocol can be instantiated from those and the complete adaptive group-testing algorithm with $\mathcal{O}(d \log(n/d))$ tests [16], and requires $\mathcal{O}(d \log(n/d) \log n)$ communication costs. Therefore, as in the previous section, our instantiation is more efficient than a naive solution, i.e., parallel executions of canonical identification protocols for each entity, which requires $\mathcal{O}(n)$ communication costs.

5 Concluding Remarks

In this paper, we considered a public-key variant of aggregate entity authentication protocols [2, 3], and introduced multi-entity identification mID protocols. We formally gave a mathematical model, formalized its security notions, and showed two generic constructions of mID protocols. Since the proposed constructions can be instantiated from lattice-based assumptions, we obtain post-quantum mID protocols. Our constructions are both signature-based ones; they require (aggregate) signature schemes as the building blocks. It would be interesting to show constructions without signature primitives, e.g., an mID protocol constructed from the canonical identification protocol [6, 7, 8].

Acknowledgement

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

References

[1] The internet of things reference model. Technical report, Cisco, 2014.

[2] Shoichi Hirose and Junji Shikata. Group-testing aggregate entity authentication. In *IEEE Information Theory Workshop (ITW) 2023*, pages 227–231. IEEE, 2023.

[3] Shoichi Hirose and Junji Shikata. Aggregate entity authentication identifying invalid entities with group testing. *Electronics*, 12(11), 2023.

[4] Jonathan Katz and AndrewY. Lindell. Aggregate message authentication codes. In *CT-RSA 2008*, volume 4964, pages 155–169. Springer Berlin Heidelberg, 2008.

[5] Robert Dorfman. The detection of defective members of large populations. *The Annals of Mathematical Statistics*, 14(4):436–440, 1943.

[6] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology – CRYPTO' 86*, volume 263, pages 186–194. Springer Berlin Heidelberg, 1987.

[7] Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In *Advances in Cryptology – CRYPTO '88*, volume 403, pages 216–231. Springer, 1988.

[8] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptol.*, 4(3):161–174, 1991.

[9] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology – EUROCRYPT 2003*, volume 2656, pages 416–432. Springer, 2003.

[10] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. How to delegate computations publicly. In *Annual ACM SIGACT Symposium on Theory of Computing (STOC) 2019*, pages 1115–1124. ACM, 2019.

[11] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Unrestricted aggregate signatures. In *ICALP 2007*, volume 4596, pages 411–422. Springer, 2007.

[12] Arkadii G. Dýachkov, Vyacheslav V. Rykov, and Ahmed M. Rashad. Superimposed distance codes. *Problems of Control and Information Theory*, 18:237–250, 1989.

[13] Ely Porat and Amir Rothschild. Explicit non-adaptive combinatorial group testing schemes. In *Automata, Languages and Programming*, pages 748–759. Springer, 2008.

[14] Matthew Aldridge, Oliver Johnson, and Jonathan Scarlett. Group testing: An information theory perspective. *Found. Trends Commun. Inf. Theory*, 15(3-4):196–392, 2019.

[15] Chou Hsiung Li. A sequential method for screening experimental variables. *Journal of the American Statistical Association*, 57(298):455–477, 1962.

[16] David Eppstein, Michael T. Goodrich, and Daniel S. Hirschberg. Improved combinatorial group testing algorithms for real-world problem sizes. *SIAM Journal on Computing*, 36(5):1360–1375, 2007.

[17] Brent Waters and David J. Wu. Batch arguments for NP and more from standard bilinear group assumptions. In *Advances in Cryptology – CRYPTO 2022*, volume 13508, pages 433–463. Springer, 2022.

[18] Arka Rai Choudhuri, Abhihsek Jain, and Zhengzhong Jin. Snargs for $\mathcal{P}$ from lwe. In *IEEE Annual Symposium on Foundations of Computer Science (FOCS) 2021*, pages 68–79, 2022.

[19] Toi Tomita and Junji Shikata. A concretely compact lattice-based aggregate signature scheme. unpublished manuscript, 2023.

[20] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.

[21] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. Dilithium. Technical report, National Institute of Standards and Technology, 2022.

[22] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon. Technical report, National Institute of Standards and Technology, 2022.

[23] Lalita Devadas, Rishab Goyal, Yael Kalai, and Vinod Vaikuntanathan. Rate-1 non-interactive arguments for batch-np and applications. In *IEEE Annual Symposium on Foundations of Computer Science (FOCS) 2022*, pages 1057–1068, 2022.

[24] Omer Paneth and Rafael Pass. Incrementally verifiable computation via rate-1 batch arguments. In *IEEE Annual Symposium on Foundations of Computer Science (FOCS) 2022*, pages 1045–1056, 2022.

[25] Toi Tomita and Junji Shikata. Compact signature aggregation from module-lattices. *IACR Cryptol. ePrint Arch.*, (471), 2023.