1-out-of-n Oblivious Signatures: Security Revisited and a Generic Construction with an Efficient Communication Cost*

Masayuki Tezuka(✉) and Keisuke Tanaka

Tokyo Institute of Technology, Tokyo, Japan
tezuka.m.ac@m.titech.ac.jp

Abstract. 1-out-of-n oblivious signature by Chen (ESORIC 1994) is a protocol between the user and the signer. In this scheme, the user makes a list of n messages and chooses the message that the user wants to obtain a signature from the list. The user interacts with the signer by providing this message list and obtains the signature for only the chosen message without letting the signer identify which messages the user chooses. Tso et al. (ISPEC 2008) presented a formal treatment of 1-out-of-n oblivious signatures. They defined unforgeability and ambiguity for 1-out-of-n oblivious signatures as a security requirement.
In this work, first, we revisit the unforgeability security definition by Tso et al. and point out that their security definition has problems. In particular, we point out that a trivial attack exists in their unforgeability security model and address this problem by modifying their security model and redefining unforgeable security.
Second, we improve the generic construction of a 1-out-of-n oblivious signature scheme by Zhou et al. (IEICE Trans 2022). The bottleneck of their construction is the size of the communication cost. We reduce the communication cost by modifying their scheme with a Merkle tree. Then we prove the security of our modified scheme.

Keywords: 1-out-of-n oblivious signatures · Generic construction · Round-optimal · Merkle tree · Efficient communication cost

1 Introduction

1.1 Background

Oblivious Signatures. The notion of 1-out-of-n oblivious signatures by Chen [6] is an interactive protocol between a signer and a user. In an oblivious signature scheme, first, the user makes a list of n messages $M = (m_i)_{i\in\{1,\ldots,n\}}$ and chooses one of message m_j in M that the user wants to obtain a signature. Then the user interacts with the signer by sending the list M with a first message μ at the beginning of the interaction. The signer can see the candidate messages M that

* This work was supported by JST CREST Grant Number JPMJCR2113 and JSPS KAKENHI Grant Number JP23K16841.

the user wants to get signed, but cannot identify which one of the messages in M is chosen by the user. After completing the interaction with the signer, the user can obtain a signature σ for only the chosen message m_j.

1-out-of-n oblivious signatures should satisfy ambiguity and unforgeability. Ambiguity prevents the signer from identifying which one of the messages the signer wants to obtain the signature in the interaction. Unforgeability requires that for each interaction, the user cannot obtain a signature of a message $m \notin M$ and can obtain a signature for only one message $m \in M$ where M is a list of message that the user sends to the signer at the beginning of the interaction.

Oblivious signatures can be used to protect the privacy of users. Chen [6] explained an application of oblivious signatures as follows. The user will buy software from the seller and the signature from the seller is needed to use the software. However, information about which software the user is interested in may be sensitive at some stage. In this situation, by using oblivious signatures, the user can make a list of n software and obtain a signature only for the one software that the user honestly wants to obtain without revealing it to the seller (signer). The oblivious signature can be used for e-voting systems [7,18].

Oblivious Signatures and Blind Signatures. Signatures with a similar flavor to oblivious signatures are blind signatures proposed by Chaum [5]. In a blind signature scheme, similar to an oblivious signature scheme, a user chooses a message and obtains a corresponding signature by interacting with the signer. Typically, blind signatures satisfy blindness and one-more unforgeability (OMUF). Blindness prevents the signer from linking a message/signature pair to the run of the protocol where it was created. OMUF security prevents the user from forging a new signature.

From the point of view of hiding the contents of the message, it may seem that blind signatures are superior than oblivious signatures. But compared to blind signatures, oblivious signature has merits listed as follows.

- **Avoid Signing Disapprove Messages:** In blind signatures, since the signer has no information about the message that the user wants to obtain the signature, the signer cannot prevent users from obtaining a signature on the message that the signer does not want to approve.
  Partially blind signatures proposed by Abe and Fujisaki [1] mitigate this problem. This scheme allows the user and the signer to agree on a predetermined piece of common information `info` which must be included in the signed message. However, similar to blind signatures, the signer has no information for the blinded part of a message, partially blind signatures do not provide a full solution for the above problem.
  By contrast, oblivious signatures allow the signer to view a list of messages. If the message that the signer does not want to approve is included in the message list, the signer can refuse to sign. Thus, the ambiguity of oblivious signatures provides a better solution for the above problem.
- **Based on Weaker Assumptions:** Recent works on blind signatures are dedicated to constructing efficient round-optimal (i.e., 2-move signing interaction) blind signature schemes [2,4,8,9,10,11,12,13,14,15,16]. However, these

schemes either rely on at least one of strong primitives, models, or assumptions such as pairing groups [4,9,10,11,13,14], non-interactive zero-knowledge (NIZK) [2,8,15,16], the random oracle model (ROM) [8,14], the generic group model (GGM) [9], interactive assumptions [4,10,11,13], q-type assumptions [12], one-more assumptions [2], or knowledge assumptions [12].

By contrast, a generic construction of a round-optimal oblivious signature scheme without the ROM was proposed in the recent work by Zhou, Liu, and Han [21]. This construction uses a digital signature scheme and a commitment scheme. This leads to instantiations in various standard assumptions (e.g., DDH, DCR, Factoring, RSA, LWE) without the ROM. Thus, the round-optimal oblivious signature schemes can be constructed with weaker assumptions than round-optimal blind signature schemes.

Previous Works on Oblivious Signatures. The notion of oblivious signatures was introduced by Chen [6] and proposed 1-out-of-n oblivious signature schemes in the ROM. Following this seminal work, several 1-out-of-n oblivious signature schemes have been proposed.

Tso, Okamoto, and Okamoto [19] formalized the syntax and security definition of the 1-out-of-n oblivious signature scheme. They gave the efficient round-optimal (i.e., 2-move) 1-out-of-n oblivious signature scheme based on the Schnorr signature scheme. The security of this scheme can be proven under the DL assumption in the ROM.

Chiou and Chen [7] proposed a t-out-of-n oblivious signature scheme. This scheme needs 3 rounds for a signing interaction and the security of this scheme can be proven under the RSA assumption in the ROM.

You, Liu, Tso, Tseng, and Mambo [20] proposed the lattice-based 1-out-of-n oblivious signature scheme. This scheme is round-optimal and the security can be proven under the short integer solution (SIS) problem in the ROM.

In recent work by Zhou, Liu, and Han [21], a generic construction of a round-optimal 1-out-of-n oblivious signature scheme was proposed. Their scheme is constructed from a commitment scheme and a digital signature scheme without the ROM. By instantiating a signature scheme and commitment scheme from standard assumptions without the ROM, this generic construction leads 1-out-of-n oblivious signature schemes from standard assumptions without the ROM. As far as we know, their scheme is the first generic construction of a 1-out-of-n oblivious signature scheme without the ROM.

1.2 Motivation

The security model for a 1-out-of-n oblivious signature scheme is formalized by Tso [19]. Their security model is fundamental for subsequent works [21,20]. However, this security model has several problems. Here, we briefly review the unforgeability security model in [19] and explain the problems of their model. The formal description of this security game is given in Section 3.2

Definition of Unforgeability in [19]. Informally, the unforgeability for a 1-out-of-n oblivious signature scheme in [19] is defined by the following game.

Let A be an adversary that executes a user part and tries to forge a new signature. A engages in the signing interaction with the signer. A can make any message list M_i and any one message $m_{i,j_i} \in M_i$. Then, A engages the i-th signing interaction with M_i at the beginning of the interaction. By interacting with the signer, A can obtain a signature σ_i on a message m_{i,j_i}. Let t be the number of signing interaction with the signer and A. Let $\mathbb{L}^{\mathsf{Sign}} = \{m_{i,j_i}\}_{i\in\{1,\ldots,t\}}$ be all messages that A obtained signatures. A wins this game if A outputs a valid signature σ^* on a message $m^* \notin \mathbb{L}^{\mathsf{Sign}}$. A 1-out-of-$n$ oblivious signature scheme satisfies unforgeability if for all PPT adversaries A cannot win the above game in non-negligible probability.

However, the above security game has several problems listed below.

- **Problem 1: How to Store Messages in $\mathbb{L}^{\mathsf{Sign}}$:** In the above security game, we need to store corresponding messages that the signer obtains signatures. However, by ambiguity property, we cannot identify the chosen message m_{i,j_i} that the signer wants to obtain a signature from a transcription of the i-th interaction with M_i. This problem can be addressed by forcing A to output (m_{i,j_i}, σ_i) at the end of each interaction. However, the next problem is serious.
- **Problem 2: Trivial Attack:** One flaw is the existence of a trivial attack on the security game. Let us consider the following adversary A that runs signing protocol execution twice. A chooses $M = (m_0, m_1)$ where m_0 and m_1 are distinct, and sets lists as $M_1 = M_2 = M$. In the 1st interaction, A chooses $m_0 \in M_1$, obtains a signature σ_0 on a message m_0, and outputs (m_0, σ_0) at the end of interaction. In the 2nd interaction, A chooses $m_1 \in M_2$, obtains a signature σ_1 on a message m_1, and outputs (m_0, σ_0) at the end of interaction. Then, A outputs a trivial forgery $(m^*, \sigma^*) = (m_1, \sigma_1)$. This attack is caused by the reuse of a signature (m_0, σ_0) at the end of the signing interaction. The unforgeability security models in previous works [20,21] are based on the model by Tso et al. [19]. This trivial attack also works for these models as well. This fact invalidates unforgeability security proofs in [6,19,20,21] for 1-out-of-n oblivious signature scheme.

  Note that we only claim that the security model in [19] has a flaw. We do not intend to claim that existing schemes in [6,19,20,21] are insecure.
- **Problem 3: Missing Adversary Strategy:** The security game does not capture an adversary with the following strategy. Let us consider an adversary A that executes the signing protocol only once. A interacts with the signer with a message list M and intends to a signature σ^* on a message $m^* \notin M$, but give up outputting (m, σ) where $m \in M$ at the end of signing interaction. Since the security game only considers the adversary that outputting (m, σ) where $m \in M$ at the end of the signing execution, the security game cannot capture the adversary A give up outputting (m, σ) where $m \in M$.

1.3 Our Contribution

The first contribution is providing a new security definition of the unforgeability security for a 1-out-of-n oblivious signature scheme. We address the problems described in the previous section. We refer the reader to Section 3.3 for more detail on our definition of unforgeability security.

The second contribution is an improvement of a generic construction of 1-out-of-n oblivious signature schemes by [20]. This round-optimal construction is obtained by a simple combination of a digital signature scheme and a commitment scheme. However, a bottleneck of this scheme is the communication cost (See Fig. 1).

Scheme	$\lvert\mathsf{vk}^{\mathsf{OS}}\rvert$	$\lvert\mu\rvert$	$\lvert\rho\rvert$	$\lvert\sigma^{\mathsf{OS}}\rvert$
$\mathsf{OS}_{\mathsf{ZLH}}$ [21]	$\lvert\mathsf{vk}^{\mathsf{DS}}\rvert$	$\lvert c^{\mathsf{COM}}\rvert$	$n\lvert\sigma^{\mathsf{DS}}\rvert$	$\lvert\sigma^{\mathsf{DS}}\rvert + \lvert c^{\mathsf{COM}}\rvert + \lvert r^{\mathsf{COM}}\rvert$
$\mathsf{OS}_{\mathsf{Ours}}$ §4.2	$\lvert\mathsf{vk}^{\mathsf{DS}}\rvert$	$\lvert c^{\mathsf{COM}}\rvert$	$\lvert\sigma^{\mathsf{DS}}\rvert$	$\lvert\sigma^{\mathsf{DS}}\rvert + \lvert c^{\mathsf{COM}}\rvert + \lvert r^{\mathsf{COM}}\rvert + (\lceil\log_2 n\rceil + 1)\lambda + \lceil\log_2 n\rceil$

Fig. 1. Comparison with generic construction of 1-out-of-n oblivious signature schemes. $\lvert\mathsf{vk}^{\mathsf{OS}}\rvert$ represents the bit length of the verification key, $\lvert\mu\rvert$ represents the bit length of the first communication, $\lvert\rho\rvert$ represents the bit length of the second communication, and $\lvert\sigma^{\mathsf{OS}}\rvert$ represents the bit length of the 1-out-of-n oblivious signature scheme. In columns, λ denotes a security parameter. $\lvert c^{\mathsf{COM}}\rvert$ (resp. $\lvert r^{\mathsf{COM}}\rvert$) denotes the bit length of a commitment (resp. randomness) and $\lvert\sigma^{\mathsf{DS}}\rvert$ (resp. $\lvert\mathsf{vk}^{\mathsf{DS}}\rvert$) denotes the bit length of a digital signature (resp. verification key) used to instantiate the 1-out-of-n oblivious signature scheme.

Particular, if the user interacts with the signer with a message list $M = (m_i)_{i\in\{1,\ldots,n\}}$ and the first communication message μ, then the signer sends n digital signatures $(\sigma_i^{\mathsf{DS}})_{i\in\{1,\ldots,n\}}$ to the user as the second communication message where σ_i^{DS} is a signature on a message (m_i, μ). This means that the second communication message cost (size) is proportional to n.

We improve the second communication cost by using a Merkle tree. Concretely, instead of signing each (m_i, μ) where $m_i \in M$, we modify it to sign a message (root, μ) where root is a root of the Merkle tree computed from M. By this modification, we reduce the communication cost of the second round from n digital signatures to only one digital signature. As a side effect of our modification, the size of the obtained 1-out-of-n oblivious signature is increasing, but it is proportional to $\log n$. Our modification has the merit that the sum of a second communication message size and a signature size is improved from $O(n)$ to $O(\log n)$.

1.4 Road Map

In Section 2, we introduce notations and review commitments, digital signatures, and Merkle tree. In Section 3, we review 1-out-of-n oblivious signatures, revisit the definition of unforgeability by Tuo et al. [19], and redefine unforgeability. In Section 4, we give a generic construction of 1-out-of-n oblivious signature schemes with efficient communication cost by improving the construction by Zhou et al. [21] and prove security for our scheme. In Section 5, we conclude our result and discuss open problems.

2 Preliminaries

In this section, we introduce notations and review fundamental cryptographic primitives for constructing our 1-out-of-n oblivious signature scheme.

2.1 Notations

Let 1^λ be the security parameter. A function f is negligible in k if $f(k) \leq 2^{-\omega(\log k)}$. For a positive integer n, we define $[n] := \{1, \ldots, n\}$. For a finite set S, $s \xleftarrow{\$} S$ represents that an element s is chosen from S uniformly at random.

For an algorithm A, $y \leftarrow \mathsf{A}(x)$ denotes that the algorithm A outputs y on input x. When we explicitly show that A uses randomness r, we denote $y \leftarrow \mathsf{A}(x; r)$. We abbreviate probabilistic polynomial time as PPT.

We use a code-based security game [3]. The game Game is a probabilistic experiment in which adversary A interacts with an implied challenger C that answers oracle queries issued by A. The Game has an arbitrary amount of additional oracle procedures which describe how these oracle queries are answered. When the game Game between the challenger C and the adversary A outputs b, we write $\mathsf{Game}_\mathsf{A} \Rightarrow b$. We say that A wins the game Game if $\mathsf{Game}_\mathsf{A} \Rightarrow 1$. We implicitly assume that the randomness in the probability term $\Pr[\mathsf{Game}_\mathsf{A} \Rightarrow 1]$ is over all the random coins in the game.

2.2 Commitment Scheme

We review a commitment scheme and its security notion.

Definition 1 (Commitment Scheme). *A commitment scheme* COM *consists of a following tuple of algorithms* $(\mathsf{KeyGen}, \mathsf{Commit})$.

- $\mathsf{KeyGen}(1^\lambda)$: *A key-generation algorithm takes as an input a security parameter* 1^λ*. It returns a commitment key* ck*. In this work, we assume that* ck *defines a message space, randomness space, and commitment space. We represent these space by* $\mathcal{M}_\mathsf{ck}$*,* Ω_ck*, and* $\mathcal{C}_\mathsf{ck}$*, respectively.*
- $\mathsf{Commit}(\mathsf{ck}, m; r)$: *A commit algorithm takes as an input a commitment key* ck*, a message* m*, and a randomness* r*. It returns a commitment* c*. In this work, we use the randomness* r *as the decommitment (i.e., opening) information for* c*.*

Definition 2 (Computational Hiding). *Let* $\mathsf{COM} = (\mathsf{KeyGen}, \mathsf{Commit})$ *a commitment scheme and* A *a PPT algorithm. We say that the* COM *satisfies computational hiding if for all* A*, the following advantage of the hiding game*

$$\mathsf{Adv}^{\mathsf{Hide}}_{\mathsf{COM},\mathsf{A}} := \left| \Pr\left[b = b^* \middle| \begin{array}{l} \mathsf{ck} \leftarrow \mathsf{COM.KeyGen}(1^\lambda), (m_0, m_1, \mathsf{st}) \leftarrow \mathsf{A}(\mathsf{ck}), \\ b \xleftarrow{\$} \{0,1\}, c^* \leftarrow \mathsf{COM.Commit}(\mathsf{ck}, m_b), b^* \leftarrow \mathsf{A}(c^*, \mathsf{st}) \end{array} \right] - \frac{1}{2} \right|$$

is negligible in λ*.*

Definition 3 (Strong Computational Binding). *Let* $\mathsf{COM} = (\mathsf{KeyGen}, \mathsf{Commit})$ *a commitment scheme and* A *a PPT algorithm. We say that the* COM *satisfies strong computational binding if the following advantage*

$$\mathsf{Adv}^{\mathsf{sBind}}_{\mathsf{COM},A} := \Pr\left[\begin{array}{l} \mathsf{Commit}(\mathsf{ck}, m; r) = \mathsf{Commit}(\mathsf{ck}, m'; r') \\ \wedge\, (m, r) \neq (m', r') \end{array} \middle| \begin{array}{l} \mathsf{ck} \leftarrow \mathsf{KeyGen}(1^\lambda), \\ ((m, r), (m', r')) \leftarrow \mathsf{A}(\mathsf{ck}) \end{array} \right]$$

is negligible in λ*.*

A commitment scheme with computational hiding and strong computational binding property can be constructed from a public key encryption (PKE) scheme with indistinguishable under chosen plaintext attack (IND-CPA) security. We refer the reader to [21] for a commitment scheme construction from a PKE scheme.

2.3 Digital Signature Scheme

We review a digital signature scheme and its security notion.

Definition 4 (Digital Signature Scheme). *A digital signature scheme* DS *consists of following four algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*.*

- $\mathsf{Setup}(1^\lambda)$: *A setup algorithm takes as an input a security parameter* 1^λ*. It returns the public parameter* pp*. In this work, we assume that* pp *defines a message space and represents this space by* $\mathcal{M}_{\mathsf{pp}}$*. We omit a public parameter* pp *in the input of all algorithms except for* KeyGen*.*
- $\mathsf{KeyGen}(\mathsf{pp})$: *A key-generation algorithm takes as an input a public parameter* pp*. It returns a verification key* vk *and a signing key* sk*.*
- $\mathsf{Sign}(\mathsf{sk}, m)$: *A signing algorithm takes as an input a signing key* sk *and a message* m*. It returns a signature* σ*.*
- $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$: *A verification algorithm takes as an input a verification key* vk*, a message* m*, and a signature* σ*. It returns a bit* $b \in \{0,1\}$*.*

Correctness. DS *satisfies correctness if for all* $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *for all* $m \in \mathcal{M}_{\mathsf{pp}}$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, *and* $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$, $\mathsf{Verify}(\mathsf{vk}, m, \sigma) = 1$ *holds.*

We review a security notion called the strong existentially unforgeable under chosen message attacks (sEUF-CMA) security for digital signature.

Definition 5 (sEUF-CMA Security). *Let* $\mathsf{DS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *be a signature scheme and* A *a PPT algorithm. The strong existentially unforgeability under chosen message attacks (sEUF-CMA) security for* DS *is defined by the sEUF-CMA security game* $\mathsf{Game}^{\mathsf{sEUFCMA}}_{\mathsf{DS},\mathsf{A}}$ *between the challenger* C *and* A *in Fig. 2.*

GAME $\mathsf{Game}^{\mathsf{sEUFCMA}}_{\mathsf{DS},\mathsf{A}}(1^\lambda)$:
$\mathbb{L}^{\mathsf{Sign}} \leftarrow \{\}$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, $(m^*, \sigma^*) \leftarrow \mathsf{A}^{\mathcal{O}^{\mathsf{Sign}}(\cdot)}(\mathsf{pp}, \mathsf{vk})$
If $\mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*) = 1 \wedge\ (m^*, \sigma^*) \notin \mathbb{L}^{\mathsf{Sign}}$, return 1. Otherwise return 0.

Oracle $\mathcal{O}^{\mathsf{Sign}}(m)$:
$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$, $\mathbb{L}^{\mathsf{Sign}} \leftarrow \mathbb{L}^{\mathsf{Sign}} \cup \{(m, \sigma)\}$, return σ.

Fig. 2. The sEUF-CMA security game $\mathsf{Game}^{\mathsf{sEUFCMA}}_{\mathsf{DS},\mathsf{A}}$.

The advantage of an adversary A *for the* sEUF-CMA *security game is defined by* $\mathsf{Adv}^{\mathsf{sEUFCMA}}_{\mathsf{DS},\mathsf{A}} := \Pr[\mathsf{Game}^{\mathsf{sEUFCMA}}_{\mathsf{DS},\mathsf{A}} \Rightarrow 1]$. DS *satisfies* sEUF-CMA *security if for all PPT adversaries* A, $\mathsf{Adv}^{\mathsf{sEUFCMA}}_{\mathsf{DS},\mathsf{A}}(1^\lambda)$ *is negligible in* λ.

2.4 Merkle Tree Technique

We review the collision resistance hash function family and the Merkle tree technique.

Definition 6 (Collision Resistance Hash Function Family). *Let* $\mathcal{H} = \{H_\lambda\}$ *be a family of hash functions where* $H_\lambda = \{H_{\lambda,i} : \{0,1\}^* \to \{0,1\}^\lambda\}_{i \in \mathcal{I}_\lambda}$. $\mathcal{H}$ *is a family of collision-resistant hash functions if for all PPT adversaries* A, *the following advantage*

$$\mathsf{Adv}^{\mathsf{Coll}}_{\mathcal{H},\mathsf{A}}(1^\lambda) := \Pr[H(x) = H(x') | H \xleftarrow{\$} H_\lambda, (x, x') \leftarrow \mathsf{A}(H)]$$

is negligible in λ.

Definition 7 (Merkle Tree Technique [17]). *The Merkle tree technique* MT *consists of following three algorithms* $(\mathsf{MerkleTree}, \mathsf{MerklePath}, \mathsf{RootReconstruct})$ *with access to a common hash function* $H : \{0,1\}^* \to \{0,1\}^\lambda$.

- $\mathsf{MerkleTree}^H(M = (m_0, \ldots, m_{2^k-1}))$: *A Merkle tree generation algorithm takes as an input a list of* 2^k *elements* $M = (m_0, \ldots, m_{2^k-1})$*. It constructs a complete binary tree whose height is* $k+1$ *(i.e., maximum level is* k*).*
  We represent a root node as w_ϵ *and a node in level* ℓ *as* $w_{b_1,\ldots b_\ell}$ *where* $b_j \in \{0,1\}$ *for* $j \in [\ell]$*. The leaf node with an index* $i \in \{0,\ldots,2^k-1\}$ *represents* $w_{\mathsf{I2B}(i)}$ *where* $\mathsf{I2B}$ *is a conversion function from an integer* i *to the* k*-bit binary representation.*
  Each leaf node with an index $i \in \{0,\ldots,2^k-1\}$ *(i.e.,* $w_{\mathsf{I2B}(i)}$*) is assigned a value* $h_{\mathsf{I2B}(i)} = H(m_i)$*. Each level* j *internal (non-leaf) node* $w_{b_1,\ldots b_j}$ *is assigned a value* $h_{b_1,\ldots b_j} = H(h_{b_1,\ldots b_j,0}||h_{b_1,\ldots b_j,1})$ *where* $h_{b_1,\ldots b_j,0}$ *and* $h_{b_1,\ldots b_j,1}$ *are values assigned to the left-children node* $w_{b_1,\ldots b_j,0}$ *and the right-children node* $w_{b_1,\ldots b_j,1}$*, respectively. The root node* w_ϵ *is assigned a hash value* $h_\epsilon = H(h_0||h_1)$ *and denote this value as* root*. This algorithm outputs a value* root *and the description* tree *which describes the entire tree.*
- $\mathsf{MerklePath}^H(\mathsf{tree}, i)$: *A Merkle path generation algorithm takes as an input a description of a tree* tree *and a leaf node index* $i \in \{0,\ldots,2^k-1\}$*. Then, this algorithm computes* $(b_1, \ldots b_k) = \mathsf{I2B}(i)$ *and outputs a list* $\mathsf{path} = (h_{\overline{b_1}}, h_{b_1,\overline{b_2}}, \ldots, h_{b_1,\ldots,\overline{b_k}})$ *where* $\overline{b_j} = 1 - b_j$ *for* $j \in [k]$*.*
- $\mathsf{RootReconstruct}^H(\mathsf{path}, m_i, i)$: *A root reconstruction algorithm takes as an input a list* $\mathsf{path} = (h_{\overline{b_1}}, h_{b_1,\overline{b_2}}, \ldots, h_{b_1,\ldots,\overline{b_k}})$*, an element* m_i*, and a leaf node index* $i \in \{0,\ldots,2^k-1\}$*. This algorithm computes* $(b_1, \ldots b_k) = \mathsf{I2B}(i)$ *and assigns* $h_{b_1,\ldots b_k}$*. For* $i = k-1$ *to* 1*, computes* $h_{b_1,\ldots b_j} = H(h_{b_1,\ldots b_j,0}||h_{b_1,\ldots b_j,1})$ *and outputs* $\mathsf{root} = H(h_0||h_1)$*.*

Lemma 1 (Collision Extractor for Merkle Tree). *There exists the following efficient collision extractor algorithms* Ext_1 *and* Ext_2*.*

- Ext_1 *takes as an input a description of Merkle tree* tree *whose root node is assigned value* root *and* (m'_i, path, i)*. If* tree *is constructed from a list* $M = (m_0, \ldots, m_{2^k-1})$*,* $m_i \neq m'_i$*, and* $\mathsf{root} = \mathsf{RootReconstruct}^H(\mathsf{path}, m_i, i)$ *holds, it outputs a collision of the hash function* H*.*
- Ext_2 *takes as an input a tuple* $(m, j, \mathsf{path}, \mathsf{path}')$*. If* $\mathsf{RootReconstruct}^H(\mathsf{path}, m, j) = \mathsf{RootReconstruct}^H(\mathsf{path}', m, j)$ *and* $\mathsf{path} \neq \mathsf{path}'$ *hold, it outputs a collision of the hash function* H*.*

3 Security of Oblivious Signatures Revisited

In this section, first, we review a definition of a 1-out-of-n signature scheme and security notion called ambiguity. Next, we review the security definition of the unforgeability in [19] and discuss the flaws of their security model. Then, we redefine the unforgeability security for a 1-out-of-n signature scheme.

3.1 (1, n)-Oblivious Signature Scheme

We review a syntax of a 1-out-of-n oblivious signature scheme and the security definition of ambiguity.

Definition 8 (Oblivious Signature Scheme). *a 1-out-of-n oblivious signature scheme* $(1,n)$-OS *consists of following algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{U}_1, \mathsf{S}_2, \mathsf{U}_{\mathsf{Der}}, \mathsf{Verify})$.

- $\mathsf{Setup}(1^\lambda)$: *A setup algorithm takes as an input a security parameter* 1^λ. *It returns the public parameter* pp. *In this work, we assume that* pp *defines a message space and represents this space by* $\mathcal{M}_{\mathsf{pp}}$. *We omit a public parameter* pp *in the input of all algorithms except for* KeyGen.
- $\mathsf{KeyGen}(\mathsf{pp})$: *A key-generation algorithm takes as an input a public parameter* pp. *It returns a verification key* vk *and a signing key* sk.
- $\mathsf{U}_1(\mathsf{vk}, M = (m_0, \ldots, m_{n-1}), j)$: *This is a first message generation algorithm that is run by a user. It takes as input a verification key* vk, *a list of message* $M = (m_0, \ldots, m_{n-1})$, *and a message index* $j \in \{0, \ldots, n-1\}$. *It returns a pair of a first message and a state* (μ, st) *or* $\perp$.
- $\mathsf{S}_2(\mathsf{vk}, \mathsf{sk}, M = (m_0, \ldots, m_{n-1}), \mu)$: *This is a second message generation algorithm that is run by a signer. It takes as input a verification key* vk, *a signing key* sk, *a list of message* $M = (m_0, \ldots, m_{n-1})$, *and a first message* μ. *It returns a second message* ρ *or* $\perp$.
- $\mathsf{U}_{\mathsf{Der}}(\mathsf{vk}, \mathsf{st}, \rho)$: *This is a signature derivation algorithm that is run by a user. It takes as an input a verification key* vk, *a state* st, *and a second message* ρ. *It returns a pair of a message and its signature* (m, σ) *or* $\perp$.
- $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$: *A verification algorithm takes as an input a verification key* vk, *a message* m, *and a signature* σ. *It returns a bit* $b \in \{0, 1\}$.

Correctness. $(1,n)$-OS *satisfies correctness if for all* $\lambda \in \mathbb{N}$, $n \leftarrow n(\lambda)$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, *for all message set* $\mathcal{M} = (m_0, \ldots, m_{n-1})$ *such that* $m_i \in \mathcal{M}_{\mathsf{pp}}$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, *for all* $j \in \{0, \ldots n-1\}$, $(\mu, \mathsf{st}) \leftarrow \mathsf{U}_1(\mathsf{vk}, M, j)$, $\rho \leftarrow \mathsf{S}_2(\mathsf{vk}, \mathsf{sk}, M, \mu)$, *and* $(m_j, \sigma) \leftarrow \mathsf{U}_{\mathsf{Der}}(\mathsf{vk}, \mathsf{st}, \rho)$, $\mathsf{Verify}(\mathsf{vk}, m_j, \sigma) = 1$ *holds.*

Definition 9 (Ambiguity). *Let* $(1,n)$-$\mathsf{OS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{U}_1, \mathsf{S}_2, \mathsf{U}_{\mathsf{Der}}, \mathsf{Verify})$ *be an oblivious signature scheme and* A *a PPT algorithm. The ambiguity for* $(1,n)$-OS *is defined by the ambiguity security game* $\mathsf{Game}^{\mathsf{Amb}}_{(1,n)\text{-}\mathsf{OS},\mathsf{A}}$ *between the challenger* C *and* A *in Fig. 3.*

GAME $\mathsf{Game}^{\mathsf{Amb}}_{(1,n)\text{-}\mathsf{OS},\mathsf{A}}(1^\lambda)$:
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$,
$(M = (m_0, \ldots, m_{n-1}), i_0, i_1, \mathsf{st}_{\mathsf{A}}) \leftarrow \mathsf{A}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
$b \xleftarrow{\$} \{0,1\}$, $(\mu, \mathsf{st}_{\mathsf{S}}) \leftarrow \mathsf{U}_1(\mathsf{vk}, M, i_b)$, $b^* \leftarrow \mathsf{A}(\mu, \mathsf{st}_{\mathsf{A}})$.
If $b^* = b$ return 1. Otherwise return 0.

Fig. 3. The ambiguity security game $\mathsf{Game}^{\mathsf{Amb}}_{(1,n)\text{-}\mathsf{OS},\mathsf{A}}$.

The advantage of an adversary A *for the ambiguity security game is defined by* $\mathsf{Adv}^{\mathsf{Amb}}_{(1,n)\text{-}\mathsf{OS},\mathsf{A}} := |\Pr[\mathsf{Game}^{\mathsf{Amb}}_{(1,n)\text{-}\mathsf{OS},\mathsf{A}} \Rightarrow 1] - \frac{1}{2}|$. $(1,n)$-OS *satisfies ambiguity if for all PPT adversaries* A, $\mathsf{Adv}^{\mathsf{Amb}}_{(1,n)\text{-}\mathsf{OS},\mathsf{A}}(1^\lambda)$ *is negligible in* λ.

3.2 Definition of Unforgeability Revisited

We review the security definition of unforgeability for $(1,n)$-OS in previous works in [19]. The unforgeability for a 1-out-of-n oblivious signature scheme in [19] is formalized by the following game between a challenger C and a PPT adversary A.

- C runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, and gives $(\mathsf{pp}, \mathsf{vk})$ to A.
- A is allowed to engage polynomially many signing protocol executions. In an i-th protocol execution,
  - A makes a list $M_i = (m_{i,0}, \ldots, m_{i,n-1})$) and chooses m_{i,j_i}.
  - A sends $(\mu_i, M_i = (m_{i,0}, \ldots, m_{i,n-1}))$ to C.
  - C runs $\rho_i \leftarrow \mathsf{S}_2(\mathsf{vk}, \mathsf{sk}, M_i, \mu_i)$ and gives ρ_i to A.
- Let $\mathbb{L}^{\mathsf{Sign}}$ be a list of messages that A obtained signatures. A outputs a forgery (m^*, σ^*) which satisfies $m^* \notin \mathbb{L}^{\mathsf{Sign}}$. A must complete all singing executions before it outputs a forgery.

If no PPT adversary A outputs a valid forgery in negligible probability in λ, $(1,n)$-OS satisfies the unforgeability security.

We point out three problems for the above security definition.

- **Problem 1: How to Store Messages in $\mathbb{L}^{\mathsf{Sign}}$:** In the above security game, we need to store corresponding messages that the signer obtains signatures. However, by ambiguity property, we cannot identify the message m_{i,j_i} that is chosen by the signer from a transcription of the i-th interaction with M_i. This security model does not explain how to record an entry of $\mathbb{L}^{\mathsf{Sign}}$.
- **Problem 2: Trivial Attack:** Let us consider the following adversary A that runs signing protocol execution twice. A chooses $M = (m_0, m_1)$ where m_0 and m_1 are distinct, and sets lists as $M_1 = M_2 = M$. In the 1st interaction, A chooses $m_0 \in M_1$, obtains a signature σ_0 on a message m_0, and outputs (m_0, σ_0) at the end of interaction. In the 2nd interaction, A chooses $m_1 \in M_2$, obtains a signature σ_1 on a message m_1, and outputs (m_0, σ_0) at the end of interaction. Then, A outputs a trivial forgery $(m^*, \sigma^*) = (m_1, \sigma_1)$. This attack is caused by the reuse of a signature (m_0, σ_0) at the end of the signing interaction.
- **Problem 3: Missing Adversary Strategy:** The security game does not capture an adversary with the following strategy. Let us consider an adversary A that executes the signing protocol only once. A interacts with the signer with a message list M and intends to forge a signature σ^* on a message $m^* \notin M$, but give up outputting (m, σ) where $m \in M$ at the end of signing interaction. Since the security game only considers the adversary that outputting (m, σ) where $m \in M$ at the end of the signing execution, the security game cannot capture the adversary A give up outputting (m, σ) where $m \in M$ and forge (m^*, σ^*) where $m^* \notin M$.

3.3 New Unforgeability Definition

To address the problems of the unforgeability security model by Tso et al. [19], we modify their security model and redefine the unforgeability security. Here, we briefly explain how to address these problems.

- **Countermeasure for Problem 1:** This problem is easy to fix by forcing A to output (m_{i,j_i}, σ_i) at the end of each signing interaction.
- **Countermeasure for Problem 2:** This attack is caused by the reuse of a signature at the end of signing interactions. That is A submits (m, σ) twice or more at the end of signing interactions.
  To address this problem, we introduce the signature reuse check. This prevents resubmission of (m, σ) at the end of signing interactions. However, this is not enough to prevent the reuse of a signature. If a signature has a re-randomizable property (i.e., The property that a signature is refreshed without the signing key), A can easily avoid resubmission and succeed in the trivial attack.
  For this reason, normal unforgeability security is not enough. We address this issue by letting strong unforgeability security be a default for the security requirement.
- **Countermeasure for Problem 3:** This problem is addressed by adding another winning condition for A. When A submits (m^*, σ^*) at the end of i-th signing interaction, if (m^*, σ^*) is valid and $m^* \notin M_i$, A wins the game where M_i is a list of messages send by A at the beginning of i-th signing interaction.

By reflecting the above countermeasures to the unforgeability security model by Tso et al. [19], we redefine the unforgeability security model as the strong unforgeability under chosen message attacks under the sequential signing interaction (Seq-sEUF-CMA) security.

Definition 10 (Seq-sEUF-CMA Security). *Let* $(1,n)$-$\mathsf{OS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{U}_1, \mathsf{S}_2, \mathsf{U}_{\mathsf{Der}}, \mathsf{Verify})$ *be a* 1*-out-of-*n *oblivious signature scheme and* A *a PPT algorithm. The strong unforgeability under chosen message attacks under the sequential signing interaction* (Seq-sEUF-CMA) *security for* $(1,n)$-OS *is defined by the* Seq-sEUF-CMA *security game* $\mathsf{Game}^{\mathsf{Seq\text{-}sEUFCMA}}_{(1,n)\text{-}\mathsf{OS},\mathsf{A}}$ *between the challenger* C *and* A *in Fig. 4.*

The advantage of an adversary A *for the* Seq-sEUF-CMA *security game is defined by* $\mathsf{Adv}^{\mathsf{Seq\text{-}sEUFCMA}}_{(1,n)\text{-}\mathsf{OS},\mathsf{A}}(1^\lambda) := \Pr[\mathsf{Game}^{\mathsf{Seq\text{-}sEUFCMA}}_{(1,n)\text{-}\mathsf{OS},\mathsf{A}}(1^\lambda) \Rightarrow 1]$. $(1,n)$-OS *satisfies* Seq-sEUF-CMA *security if for all PPT adversaries* A, $\mathsf{Adv}^{\mathsf{Seq\text{-}sEUFCMA}}_{(1,n)\text{-}\mathsf{OS},\mathsf{A}}(1^\lambda)$ *is negligible in* λ.

Our security model is the sequential signing interaction model. One may think that it is natural to consider the concurrent signing interaction model. However, by extending our model to the concurrent signing setting there is a trivial attack. We discuss the security model that allows concurrent signing interaction in Section 5.

4 Our Construction

In this section, first, we review the generic construction by Zhou et al. [21]. Second, we propose our new generic construction based on their construction. Then, we prove the security of our proposed scheme.

GAME $\mathsf{Game}^{\mathsf{Seq\text{-}sEUFCMA}}_{(1,n)\text{-}\mathsf{OS},\mathsf{A}}(1^\lambda)$:
$\mathbb{L}^{\mathsf{Sign}} \leftarrow \{\}$, $\mathbb{L}^{\mathsf{ListM}} \leftarrow \{\}$, $q^{\mathsf{Sign}} \leftarrow 0$, $q^{\mathsf{Fin}} \leftarrow 0$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, $(m^*,\sigma^*) \leftarrow \mathsf{A}^{\mathcal{O}^{\mathsf{Sign}}(\cdot,\cdot),\mathcal{O}^{\mathsf{Fin}}(\cdot,\cdot)}(\mathsf{pp},\mathsf{vk})$
If $q^{\mathsf{Sign}} = q^{\mathsf{Fin}} \wedge \mathsf{Verify}(\mathsf{vk},m^*,\sigma^*) = 1 \wedge (m^*,\sigma^*) \notin \mathbb{L}^{\mathsf{Sign}}$, return 1.
Otherwise return 0.

Oracle $\mathcal{O}^{\mathsf{Sign}}(M_{q^{\mathsf{Sign}}},\mu)$:
If $q^{\mathsf{Sign}} \neq q^{\mathsf{Fin}}$, return $\perp$.
$\rho \leftarrow \mathsf{S}_2(\mathsf{vk},\mathsf{sk},M,\mu)$, if $\rho = \perp$, return $\perp$.
If $\rho \neq \perp$, $q^{\mathsf{Sign}} \leftarrow q^{\mathsf{Sign}} + 1$, $\mathbb{L}^{\mathsf{ListM}} \leftarrow \mathbb{L}^{\mathsf{ListM}} \cup \{(q^{\mathsf{Sign}}, M_{q^{\mathsf{Sign}}})\}$, return ρ.
Oracle $\mathcal{O}^{\mathsf{Fin}}(m^*,\sigma^*)$:
If $q^{\mathsf{Sign}} \neq q^{\mathsf{Fin}} + 1$, return $\perp$.
If $\mathsf{Verify}(\mathsf{vk},m^*,\sigma^*) = 0$, return the game output 0 and abort.
(//Oblivious signature reuse check)
If $(m^*,\sigma^*) \in \mathbb{L}^{\mathsf{Sign}}$, return the game output 0 and abort.
Retrieve an entry $(q^{\mathsf{Sign}}, M_{q^{\mathsf{Sign}}}) \in \mathbb{L}^{\mathsf{ListM}}$.
If $m^* \in M_{q^{\mathsf{Sign}}}$, $\mathbb{L}^{\mathsf{Sign}} \leftarrow \mathbb{L}^{\mathsf{Sign}} \cup \{(m^*,\sigma^*)\}$, $q^{\mathsf{Fin}} \leftarrow q^{\mathsf{Fin}} + 1$, return "`accept`".
(//Capture adversaries that give up completing signing executions in the game.)
If $m^* \notin M_{q^{\mathsf{Sign}}}$, return the game output 1 and abort.

Fig. 4. The Seq-sEUF-CMA security game $\mathsf{Game}^{\mathsf{Seq\text{-}sEUFCMA}}_{(1,n)\text{-}\mathsf{OS},\mathsf{A}}$. The main modifications from previous works security game are highlighted in white box.

4.1 Generic Construction by Zhou et al. [21]

The generic construction of a 1-out-of-n signature scheme $(1,n)$-$\mathsf{OS}_{\mathsf{ZLH}}$ by Zhou et al. [21] is a combination of a commitment scheme COM and a digital signature scheme DS. Their construction $(1,n)$-$\mathsf{OS}_{\mathsf{ZLH}}[\mathsf{COM},\mathsf{DS}] = (\mathsf{OS.Setup}, \mathsf{OS.KeyGen}, \mathsf{OS.U}_1, \mathsf{OS.S}_2, \mathsf{OS.U}_{\mathsf{Der}}, \mathsf{OS.Verify})$ is given in Fig. 5.

We briefly provide an overview of a signing interaction and an intuition for the security of their construction. In the signing interaction, the user chooses a message list $M = (m_i)_{i\in\{0,\ldots,n-1\}}$ and a specific message m_{j_i} that the user wants to obtain the corresponding signature. To hide this choice from the signer, the signer computes the commitment c on m_j with the randomness r. The user sends $(M, \mu = c)$ to the signer.

Here, we provide an intuition for the security of their construction. From the view of the signer, by the hiding property of the commitment scheme, the signer does not identify m_j from $(M, \mu = c)$. This guarantees the ambiguity of their construction. The signer computes a signature σ_i^{DS} on a tuple (m_i, c) for $i \in \{0,\ldots,n-1\}$ and sends $\rho = (\sigma_i^{\mathsf{DS}})_{i\in\{0,\ldots,n-1\}}$.

If the signer honestly computes c on $m_j \in M$, we can verify that m_{j_i} is committed into c by decommitting with r. An oblivious signature on m_j is obtained as $\sigma^{\mathsf{OS}} = (c, r, \sigma_j^{\mathsf{DS}})$. If a malicious user wants to obtain two signatures for two distinct messages $m, m' \in M$ or obtain a signature on $m^* \notin M$ from the signing protocol execution output $(M = (m_i)_{i\in\{0,\ldots,n-1\}}, \mu, \rho)$, the malicious

$\mathsf{OS.Setup}(1^\lambda)$:
  $\mathsf{ck} \leftarrow \mathsf{COM.KeyGen}(1^\lambda)$, $\mathsf{pp}^{\mathsf{DS}} \leftarrow \mathsf{DS.Setup}(1^\lambda)$, return $\mathsf{pp}^{\mathsf{OS}} \leftarrow (\mathsf{ck}, \mathsf{pp}^{\mathsf{DS}})$.
$\mathsf{OS.KeyGen}(\mathsf{pp}^{\mathsf{OS}} = (\mathsf{ck}, \mathsf{pp}^{\mathsf{DS}}))$:
  $(\mathsf{vk}^{\mathsf{DS}}, \mathsf{sk}^{\mathsf{DS}}) \leftarrow \mathsf{DS.KeyGen}(\mathsf{pp}^{\mathsf{DS}})$, return $(\mathsf{vk}^{\mathsf{OS}}, \mathsf{sk}^{\mathsf{OS}}) \leftarrow (\mathsf{vk}^{\mathsf{DS}}, \mathsf{sk}^{\mathsf{DS}})$.
$\mathsf{OS.U}_1(\mathsf{vk}^{\mathsf{OS}}, M = (m_0, \ldots, m_{n-1}), j \in \{0, \ldots, n-1\})$:
  $r \xleftarrow{\$} \Omega_{\mathsf{ck}}$, $c \leftarrow \mathsf{COM.Commit}(\mathsf{ck}, m; r)$, $\mu \leftarrow c$, $\mathsf{st} \leftarrow (M, c, r, j)$, return (μ, st).
$\mathsf{OS.S}_2(\mathsf{vk}^{\mathsf{OS}}, \mathsf{sk}^{\mathsf{OS}} = \mathsf{sk}^{\mathsf{DS}}, M = (m_0, \ldots, m_{n-1}), \mu = c)$:
  For $i = 0$ to $n-1$, $\sigma_i^{\mathsf{DS}} \leftarrow \mathsf{DS.Sign}(\mathsf{sk}^{\mathsf{DS}}, (m_i, c))$.
  Return $\rho \leftarrow (\sigma_0^{\mathsf{DS}}, \ldots, \sigma_{n-1}^{\mathsf{DS}})$.
$\mathsf{OS.U}_{\mathsf{Der}}(\mathsf{vk}^{\mathsf{OS}} = \mathsf{vk}^{\mathsf{DS}}, \mathsf{st} = (M = (m_0, \ldots, m_{n-1}), c, r, j), \rho = (\sigma_0^{\mathsf{DS}}, \ldots, \sigma_{n-1}^{\mathsf{DS}}))$:
  For $i = 0$ to $n-1$, if $\mathsf{DS.Verify}(\mathsf{vk}^{\mathsf{DS}}, (m_i, c), \sigma_i^{\mathsf{DS}}) = 0$, return $\perp$.
  $\sigma^{\mathsf{OS}} \leftarrow (c, r, \sigma_j^{\mathsf{DS}})$, return $(m_j, \sigma^{\mathsf{OS}})$.
$\mathsf{OS.Verify}(\mathsf{vk}^{\mathsf{OS}} = \mathsf{vk}^{\mathsf{DS}}, m, \sigma^{\mathsf{OS}} = (c, r, \sigma^{\mathsf{DS}}))$:
  If $c \neq \mathsf{COM.Commit}(\mathsf{ck}, m; r)$, return 0.
  If $\mathsf{DS.Verify}(\mathsf{vk}^{\mathsf{DS}}, (m, c), \sigma^{\mathsf{DS}}) = 0$, return 0.
  Otherwise return 1.

Fig. 5. The generic construction $(1, n)$-$\mathsf{OS}_{\mathsf{ZLH}}[\mathsf{COM}, \mathsf{DS}]$.

user must break either the EUF-CMA security of DS or the binding property of COM. This guarantees the unforgeability security of their construction.

A drawback of their construction is the second communication cost. A second message ρ consists of n digital signatures. If n becomes large, it will cause heavy communication traffic. It is desirable to reduce the number of signatures in ρ.

4.2 Our Generic Construction

As explained in the previous section, the drawback of the construction by Zhou et al. [21] is the size of a second message ρ. To circumvent this bottleneck, we improve their scheme by using a Merkle tree technique. Concretely, instead of signing on (m_i, c) for each $m_i \in M$, we modify it to sign on (root, c) where root is a root of the Merkle tree computed from M. This modification allows us to reduce the number of digital signatures included in ρ from n to 1.

Now, we describe our construction. Let COM be a commitment scheme, DS a digital signature scheme, $\mathcal{H} = \{H_\lambda\}$ a hash function family, and $\mathsf{MT} = (\mathsf{MerkleTree}, \mathsf{MerklePath}, \mathsf{RootReconstruct})$ a Merkle tree technique in Section 2.4. To simplify the discussion, we assume that $n > 1$ is a power of 2. [1]

Our generic construction $(1, n)$-$\mathsf{OS}_{\mathsf{Ours}}[\mathcal{H}, \mathsf{COM}, \mathsf{DS}] = (\mathsf{OS.Setup}, \mathsf{OS.KeyGen}, \mathsf{OS.U}_1, \mathsf{OS.S}_2, \mathsf{OS.U}_{\mathsf{Der}}, \mathsf{OS.Verify})$ is given in Fig. 6.

[1] With the following modification, our scheme also supports the case where $n > 1$ is not a power of 2. Let k be an integer such that $2^{k-1} < n < 2^k$. We change a list of message $M = (m_0, \ldots m_{n-1})$ which is given to $\mathsf{OS.U}_1$ and $\mathsf{OS.S}_2$ as a part of an input to an augmented message list $M' = (m'_0, \ldots m'_{2^k-1})$ where $m'_i = m_i$ for $i \in \{0, \ldots, n-1\}$, $m'_{n-1+i} = \phi || i$ for $i \in \{1, \ldots 2^k - n\}$, and ϕ is a special symbol representing that a message is empty.

$\mathsf{OS.Setup}(1^\lambda)$:
$H \xleftarrow{\$} H_\lambda$, $\mathsf{ck} \leftarrow \mathsf{COM.KeyGen}(1^\lambda)$, $\mathsf{pp}^{\mathsf{DS}} \leftarrow \mathsf{DS.Setup}(1^\lambda)$,
Return $\mathsf{pp}^{\mathsf{OS}} \leftarrow (H, \mathsf{ck}, \mathsf{pp}^{\mathsf{DS}})$.
$\mathsf{OS.KeyGen}(\mathsf{pp}^{\mathsf{OS}} = (\mathsf{ck}, \mathsf{pp}^{\mathsf{DS}}))$:
$(\mathsf{vk}^{\mathsf{DS}}, \mathsf{sk}^{\mathsf{DS}}) \leftarrow \mathsf{DS.KeyGen}(\mathsf{pp}^{\mathsf{DS}})$, return $(\mathsf{vk}^{\mathsf{OS}}, \mathsf{sk}^{\mathsf{OS}}) \leftarrow (\mathsf{vk}^{\mathsf{DS}}, \mathsf{sk}^{\mathsf{DS}})$.
$\mathsf{OS.U}_1(\mathsf{vk}^{\mathsf{OS}}, M = (m_0, \ldots, m_{n-1}), j \in \{0, \ldots, n-1\})$:
If there exists $(t, t') \in \{0, \ldots, n-1\}^2$ s.t. $t \neq t' \wedge m_t = m_{t'}$, return $\perp$.
$r \xleftarrow{\$} \Omega_{\mathsf{ck}}$, $c \leftarrow \mathsf{COM.Commit}(\mathsf{ck}, m; r)$, $\mu \leftarrow c$, $\mathsf{st} \leftarrow (M, c, r, j)$, return (μ, st).
$\mathsf{OS.S}_2(\mathsf{vk}^{\mathsf{OS}}, \mathsf{sk}^{\mathsf{OS}} = \mathsf{sk}^{\mathsf{DS}}, M = (m_0, \ldots, m_{n-1}), \mu = c)$:
If there exists $(t, t') \in \{0, \ldots, n-1\}^2$ s.t. $t \neq t' \wedge m_t = m_{t'}$, return $\perp$.
$(\mathsf{root}, \mathsf{tree}) \leftarrow \mathsf{MerkleTree}^H(M)$, $\sigma^{\mathsf{DS}} \leftarrow \mathsf{DS.Sign}(\mathsf{sk}^{\mathsf{DS}}, (\mathsf{root}, c))$.
Return $\rho \leftarrow \sigma^{\mathsf{DS}}$.
$\mathsf{OS.U}_{\mathsf{Der}}(\mathsf{vk}^{\mathsf{OS}} = \mathsf{vk}^{\mathsf{DS}}, \mathsf{st} = (M = (m_0, \ldots, m_{n-1}), c, r, j), \rho = (\sigma_1^{\mathsf{DS}}, \ldots, \sigma_n^{\mathsf{DS}}))$:
$(\mathsf{root}, \mathsf{tree}) \leftarrow \mathsf{MerkleTree}^H(M)$, $\mathsf{path} \leftarrow \mathsf{MerklePath}^H(\mathsf{tree}, j)$
If $\mathsf{DS.Verify}(\mathsf{vk}^{\mathsf{DS}}, (\mathsf{root}, c), \sigma^{\mathsf{DS}}) = 0$, return $\perp$.
$\sigma^{\mathsf{OS}} \leftarrow (\mathsf{root}, c, \sigma^{\mathsf{DS}}, \mathsf{path}, j, r)$, return $(m_j, \sigma^{\mathsf{OS}})$.
$\mathsf{OS.Verify}(\mathsf{vk}^{\mathsf{OS}} = \mathsf{vk}^{\mathsf{DS}}, m, \sigma^{\mathsf{OS}} = (\mathsf{root}, c, \sigma^{\mathsf{DS}}, \mathsf{path}, j, r)$:
If $\mathsf{root} \neq \mathsf{RootReconstruct}^H(\mathsf{path}, m, j)$, return 0.
If $c \neq \mathsf{COM.Commit}(\mathsf{ck}, m; r)$, return 0.
If $\mathsf{DS.Verify}(\mathsf{vk}^{\mathsf{DS}}, (\mathsf{root}, c), \sigma^{\mathsf{DS}}) = 0$, return 0.
Otherwise return 1.

Fig. 6. Our generic construction $(1,n)$-$\mathsf{OS}_{\mathsf{Ours}}[\mathcal{H}, \mathsf{COM}, \mathsf{DS}]$.

4.3 Analysis

We analyze our scheme $(1,n)$-$\mathsf{OS}_{\mathsf{Ours}}$. It is easy to see that our scheme satisfies the correctness. Now, we prove that our generic construction $(1,n)$-$\mathsf{OS}_{\mathsf{Ours}}$ satisfies the ambiguity and the Seq-sEUF-CMA security.

Theorem 1. *If* COM *is computational hiding commitment,* $(1,n)$-$\mathsf{OS}_{\mathsf{Ours}}[\mathcal{H}, \mathsf{COM}, \mathsf{DS}]$ *satisfies the ambiguity.*

Proof. The ambiguity of our scheme can be proven in a similar way in [21]. Let A be an adversary for the ambiguity game of $(1,n)$-$\mathsf{OS}_{\mathsf{Ours}}$. We give a reduction algorithm B that reduces the ambiguity security of our scheme to the computational hiding property of COM in Fig. 7.

Now, we confirm that B simulates the ambiguity game of $(1,n)$-$\mathsf{OS}_{\mathsf{Ours}}$. In the case that $b = 0$, $c^* \leftarrow \mathsf{COM.Commit}(\mathsf{ck}, m_0^* = m_{i_0})$ holds. B simulates μ on the choice of m_{i_0} in this case. Similary, in the case that $b = 1$, $c^* \leftarrow \mathsf{COM.Commit}(\mathsf{ck}, m_1^* = m_{i_1})$ holds. B simulates μ on the choice of m_{i_1} in this case. Since b is chosen uniformly at random from $\{0,1\}$, B perfectly simulates the ambiguity game of $(1,n)$-$\mathsf{OS}_{\mathsf{Ours}}$. We can see that $\mathsf{Adv}^{\mathsf{Hide}}_{\mathsf{COM},\mathsf{B}}(1^\lambda) = \mathsf{Adv}^{\mathsf{Amb}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}(1^\lambda)$ holds. Thus, we can conclude Theorem 1. □

Theorem 2. *If* $\mathcal{H}$ *is a family of collision-resistant hash functions,* DS *satisfies the* sEUF-CMA *security, and* COM *is computational binding commitment,* $(1,n)$-$\mathsf{OS}_{\mathsf{Ours}}[\mathcal{H}, \mathsf{COM}, \mathsf{DS}]$ *satisfies the* Seq-sEUF-CMA *security.*

$\mathsf{B}(1^\lambda, \mathsf{ck})$:
$H \xleftarrow{\$} H_\lambda$, $\mathsf{pp}^{\mathsf{DS}} \leftarrow \mathsf{DS.Setup}(1^\lambda)$, $\mathsf{pp}^{\mathsf{OS}} \leftarrow (H, \mathsf{ck}, \mathsf{pp}^{\mathsf{DS}})$,
$(\mathsf{vk}^{\mathsf{DS}}, \mathsf{sk}^{\mathsf{DS}}) \leftarrow \mathsf{DS.KeyGen}(\mathsf{pp}^{\mathsf{DS}})$, $(\mathsf{vk}^{\mathsf{OS}}, \mathsf{sk}^{\mathsf{OS}}) \leftarrow (\mathsf{vk}^{\mathsf{DS}}, \mathsf{sk}^{\mathsf{DS}})$,
$(M = (m_0, \ldots, m_{n-1}), i_0, i_1, \mathsf{st}_{\mathsf{A}}) \leftarrow \mathsf{A}(\mathsf{pp}^{\mathsf{OS}}, \mathsf{vk}^{\mathsf{OS}}, \mathsf{sk}^{\mathsf{OS}})$
$m_0^* \leftarrow m_{i_0}$, $m_1^* \leftarrow m_{i_1}$, send (m_0^*, m_1^*) to the challenger C and obtain
a commitment c^* where $c^* \leftarrow \mathsf{COM.Commit}(\mathsf{ck}, m_b^*)$ and $b \xleftarrow{\$} \{0,1\}$ is chosen C.
$b' \leftarrow \mathsf{A}(\mu = c^*, \mathsf{st}_{\mathsf{A}})$, return $b^* \leftarrow b'$.

Fig. 7. The reduction algorithm B.

Proof. Let A be a PPT adversary for the Seq-sEUF-CMA game of $(1,n)$-$\mathsf{OS}_{\mathsf{Ours}}$. We introduce the base game $\mathsf{Game}^{\mathsf{Base}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}$ which simulates $\mathsf{Game}^{\mathsf{Seq\text{-}sEUFCMA}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}$. We provide $\mathsf{Game}^{\mathsf{Base}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}$ in Fig. 8.

$\mathsf{Game}^{\mathsf{Base}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}$ simulates the game $\mathsf{Game}^{\mathsf{Seq\text{-}sEUFCMA}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}$ by introducing flags (e.g., Final, $\mathtt{DS}_{\mathtt{reuse}}$) which are used for classifying forgery type and a table $\mathbb{T}$ which stores the computation of the signing oracle $\mathcal{O}^{\mathsf{Sign}}$. More precisely, the flag Final represents that a forgery $(m^*, \sigma^{*\mathsf{OS}} = (\mathsf{root}^*, c^*, \sigma^{*\mathsf{DS}}, \mathsf{path}^*, j^*, r^*))$ is submitted in the final output ($\mathsf{Final} = \mathtt{true}$) or $\mathcal{O}^{\mathsf{Fin}}$ ($\mathsf{Final} = \mathtt{false}$). The flag $\mathtt{DS}_{\mathtt{reuse}}$ represents that there is a pair $(\widetilde{m}, \widetilde{\sigma}^{\mathsf{OS}}) \neq (\widetilde{m}', \widetilde{\sigma}'^{\mathsf{OS}})$ in $\mathbb{L}^{\mathsf{Sign}}$ such that the first three elements of σ^{OS} are the same. i.e., $(\widetilde{\mathsf{root}}, \widetilde{c}, \widetilde{\sigma}^{\mathsf{DS}}) = (\widetilde{\mathsf{root}}', \widetilde{c}', \widetilde{\sigma}'^{\mathsf{DS}})$ holds. We represent that such a pair exists as $\mathtt{DS}_{\mathtt{reuse}} = \mathtt{true}$. The table $\mathbb{T}$ stores a tuple $(i, M, \mathsf{root}, c, \sigma^{\mathsf{DS}})$ where (M, c) is an input for an i-th $\mathcal{O}^{\mathsf{Sign}}$ query, $(\mathsf{root}, \mathsf{tree}) \leftarrow \mathsf{MerkleTree}^H(M)$, $\sigma^{\mathsf{DS}} \leftarrow \mathsf{DS.Sign}(\mathsf{sk}^{\mathsf{DS}}, (\mathsf{root}, c))$. The counter q^{Sign} represents the number of outputs that A received from the $\mathcal{O}^{\mathsf{Sign}}$ oracle and q^{Fin} represent the number of submitted signatures from A.

Now, we divide an adversary A into three types $\mathsf{A}_1, \mathsf{A}_2, \mathsf{A}_3$ according to states of flags $\mathtt{DS}_{\mathtt{reuse}}$, $\mathtt{DS}_{\mathtt{forge}}$, and $\mathtt{COM}_{\mathtt{coll}}$ when A wins the game $\mathsf{Game}^{\mathsf{Base}}$.

- A_1 wins the game with $\mathtt{DS}_{\mathtt{forge}} = \mathtt{true}$.
- A_2 wins the game with $\mathtt{COM}_{\mathtt{coll}} = \mathtt{true}$.
- A_3 wins the game with $\mathtt{DS}_{\mathtt{forge}} = \mathtt{false} \wedge \mathtt{COM}_{\mathtt{coll}} = \mathtt{false}$.

For adversaries A_1, A_2, and A_3, we can construct a reduction for the security of DS, COM, and H respectively. Now, we give reductions for these adversaries.

Reduction B^{DS}: A reduction B^{DS} to the $\mathsf{sEUFCMA}$ security game of DS is obtained by modifying $\mathsf{Game}^{\mathsf{Base}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}$ as follows. Instead of running $\mathsf{pp}^{\mathsf{DS}} \leftarrow \mathsf{DS.Setup}(1^\lambda)$ and $(\mathsf{vk}^{\mathsf{DS}}, \mathsf{sk}^{\mathsf{DS}}) \leftarrow \mathsf{DS.KeyGen}(\mathsf{pp}^{\mathsf{DS}})$, B^{DS} uses $(\mathsf{pp}^{\mathsf{DS}}, \mathsf{vk}^{\mathsf{DS}})$ given by the $\mathsf{sEUFCMA}$ security game of DS. For a signing query (M, c) from A, B^{DS} query (root, c) to the signing oracle of the $\mathsf{sEUFCMA}$ security game of DS, obtains $\sigma^{\mathsf{DS}} \leftarrow \mathsf{DS.Sign}(\mathsf{sk}^{\mathsf{DS}}, (\mathsf{root}, c))$, and returns σ^{DS}. To simplify the discussion, we assume that A makes distinct (M, c) to B^{DS}. (If A makes the same (M, c) more than once, B^{DS} simply outputs return $\sigma^{\mathsf{DS}} \leftarrow \mathsf{DS.Sign}(\mathsf{sk}^{\mathsf{DS}}, (\mathsf{root}, c))$ which was previously obtained by the signing oracle of the $\mathsf{sEUFCMA}$ security game where root is computed from M.)

$\mathsf{Game}^{\mathsf{Base}}_{\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}(1^\lambda)$:
$\mathbb{L}^{\mathsf{Sign}} \leftarrow \{\}$, $\mathbb{L}^{\mathsf{ListM}} \leftarrow \{\}$, $\mathbb{T} \leftarrow \{\}$, $q^{\mathsf{Sign}} \leftarrow 0$, $q^{\mathsf{Fin}} \leftarrow 0$, $\mathsf{Final} \leftarrow \mathtt{false}$,
$\mathtt{DS}_{\mathtt{reuse}} \leftarrow \mathtt{false}$, $\mathtt{COM}_{\mathtt{coll}} \leftarrow \mathtt{false}$, $\mathtt{DS}_{\mathtt{forge}} \leftarrow \mathtt{false}$, $\mathsf{ck} \leftarrow \mathsf{COM.KeyGen}(1^\lambda)$,
$\mathsf{pp}^{\mathsf{DS}} \leftarrow \mathsf{DS.Setup}(1^\lambda)$, $\mathsf{pp}^{\mathsf{OS}} \leftarrow (H, \mathsf{ck}, \mathsf{pp}^{\mathsf{DS}})$, $(\mathsf{vk}^{\mathsf{DS}}, \mathsf{sk}^{\mathsf{DS}}) \leftarrow \mathsf{DS.KeyGen}(\mathsf{pp}^{\mathsf{DS}})$,
$(\mathsf{vk}^{\mathsf{OS}}, \mathsf{sk}^{\mathsf{OS}}) \leftarrow (\mathsf{vk}^{\mathsf{DS}}, \mathsf{sk}^{\mathsf{DS}})$, $(m^*, \sigma^{*\mathsf{OS}}) \leftarrow \mathsf{A}^{\mathcal{O}^{\mathsf{Sign}}(\cdot,\cdot),\mathcal{O}^{\mathsf{Fin}}(\cdot,\cdot)}(\mathsf{pp}^{\mathsf{OS}}, \mathsf{vk}^{\mathsf{OS}})$
If $q^{\mathsf{Sign}} \neq q^{\mathsf{Fin}} \vee \mathsf{OS.Verify}(\mathsf{vk}^{\mathsf{OS}}, m^*, \sigma^*) \neq 1 \vee (m^*, \sigma^{*\mathsf{OS}}) \in \mathbb{L}^{\mathsf{Sign}}$, return 0.
$\mathsf{Final} \leftarrow \mathtt{true}$, $\mathbb{L}^{\mathsf{Sign}} \leftarrow \mathbb{L}^{\mathsf{Sign}} \cup \{(m^*, \sigma^{*\mathsf{OS}})\}$, $q^{\mathsf{Fin}} \leftarrow q^{\mathsf{Fin}} + 1$
Search a pair $(\widetilde{m}, \widetilde{\sigma}^{\mathsf{OS}}) \neq (\widetilde{m}', \widetilde{\sigma}'^{\mathsf{OS}})$ in $\mathbb{L}^{\mathsf{Sign}}$ such that the first three elements of σ^{OS} are the same. i.e., $(\widetilde{\mathsf{root}}, \widetilde{c}, \widetilde{\sigma}^{\mathsf{DS}}) = (\widetilde{\mathsf{root}}', \widetilde{c}', \widetilde{\sigma}'^{\mathsf{DS}})$
If there is no such a pair, $\mathtt{DS}_{\mathtt{forge}} \leftarrow \mathtt{true}$, return 1.
$(\mathsf{Final} = \mathtt{true} \wedge \mathtt{DS}_{\mathtt{reuse}} = \mathtt{false} \wedge \mathtt{DS}_{\mathtt{forge}} = \mathtt{true} \wedge \mathtt{COM}_{\mathtt{coll}} = \mathtt{false})$
$\mathtt{DS}_{\mathtt{reuse}} \leftarrow \mathtt{true}$.
Parse $\widetilde{\sigma}^{\mathsf{OS}}$ as $(\mathsf{root}^*, c^*, \sigma^{*\mathsf{DS}}, \widetilde{\mathsf{path}}, \widetilde{j}, \widetilde{r})$, $\widetilde{\sigma}'^{\mathsf{OS}}$ as $(\mathsf{root}^*, c^*, \sigma^{*\mathsf{DS}}, \widetilde{\mathsf{path}}', \widetilde{j}', \widetilde{r}')$.
If $(\widetilde{m}, \widetilde{r}) \neq (\widetilde{m}', \widetilde{r}')$, $\mathtt{COM}_{\mathtt{coll}} \leftarrow \mathtt{true}$, return 1.
$(\mathsf{Final} = \mathtt{true} \wedge \mathtt{DS}_{\mathtt{reuse}} = \mathtt{true} \wedge \mathtt{DS}_{\mathtt{forge}} = \mathtt{false} \wedge \mathtt{COM}_{\mathtt{coll}} = \mathtt{true})$
Otherwise, return 1.
$(\mathsf{Final} = \mathtt{true} \wedge \mathtt{DS}_{\mathtt{reuse}} = \mathtt{true} \wedge \mathtt{DS}_{\mathtt{forge}} = \mathtt{false} \wedge \mathtt{COM}_{\mathtt{coll}} = \mathtt{false})$

Oracle $\mathcal{O}^{\mathsf{Sign}}(M = (m_0, \ldots, m_{n-1}), \mu = c)$:
If $q^{\mathsf{Sign}} \neq q^{\mathsf{Fin}}$, return $\perp$.
If there exists a pair $(t \neq t' \in \{0, \ldots, n-1\})$ such that $m_t = m_{t'}$, return $\perp$.
$(\mathsf{root}, \mathsf{tree}) \leftarrow \mathsf{MerkleTree}^H(M)$, $\sigma^{\mathsf{DS}} \leftarrow \mathsf{DS.Sign}(\mathsf{sk}^{\mathsf{DS}}, (\mathsf{root}, c))$,
$q^{\mathsf{Sign}} \leftarrow q^{\mathsf{Sign}} + 1$, $M_{q^{\mathsf{Sign}}} \leftarrow M$, $\mathbb{L}^{\mathsf{ListM}} \leftarrow \mathbb{L}^{\mathsf{ListM}} \cup \{(q^{\mathsf{Sign}}, M_{q^{\mathsf{Sign}}})\}$,
$\mathbb{T} \leftarrow \mathbb{T} \cup \{(q^{\mathsf{Sign}}, M_{q^{\mathsf{Sign}}}, \mathsf{root}, c, \sigma^{\mathsf{DS}})\}$,
return $\rho \leftarrow \sigma^{\mathsf{DS}}$ to A.
Oracle $\mathcal{O}^{\mathsf{Fin}}(m^*, \sigma^{*\mathsf{OS}})$:
If $q^{\mathsf{Sign}} \neq q^{\mathsf{Fin}} + 1$, return $\perp$.
If $\mathsf{OS.Verify}(\mathsf{vk}^{\mathsf{OS}}, m^*, \sigma^{*\mathsf{OS}}) \neq 1$, return the game output 0 and abort.
If $(m^*, \sigma^{*\mathsf{OS}}) \in \mathbb{L}^{\mathsf{Sign}}$, return the game output 0 and abort.
$\mathbb{L}^{\mathsf{Sign}} \leftarrow \mathbb{L}^{\mathsf{Sign}} \cup \{(m^*, \sigma^{*\mathsf{OS}})\}$, $q^{\mathsf{Fin}} \leftarrow q^{\mathsf{Fin}} + 1$, retrieve $(q^{\mathsf{Sign}}, M_{q^{\mathsf{Sign}}}) \in \mathbb{L}^{\mathsf{ListM}}$.
If $m^* \in M_{q^{\mathsf{Sign}}}$, return "$\mathtt{accept}$" to A.
Parse $\sigma^{*\mathsf{OS}}$ as $(\mathsf{root}^*, c^*, \sigma^{*\mathsf{DS}}, \mathsf{path}^*, j^*, r^*)$.
If $(q^{\mathsf{Sign}}, *, \mathsf{root}^*, c^*, \sigma^{*\mathsf{DS}}) \in \mathbb{T}$ return the game output 1.
$(\mathsf{Final} = \mathtt{false} \wedge \mathtt{DS}_{\mathtt{reuse}} = \mathtt{false} \wedge \mathtt{DS}_{\mathtt{forge}} = \mathtt{false} \wedge \mathtt{COM}_{\mathtt{coll}} = \mathtt{false})$
Search a pair $(\widetilde{m}, \widetilde{\sigma}^{\mathsf{OS}}) \neq (\widetilde{m}', \widetilde{\sigma}'^{\mathsf{OS}})$ in $\mathbb{L}^{\mathsf{Sign}}$ such that the first three elements of σ^{OS} are the same. i.e., $(\widetilde{\mathsf{root}}, \widetilde{c}, \widetilde{\sigma}^{\mathsf{DS}}) = (\widetilde{\mathsf{root}}', \widetilde{c}', \widetilde{\sigma}'^{\mathsf{DS}})$
If there is no such a pair, $\mathtt{DS}_{\mathtt{forge}} \leftarrow \mathtt{true}$, return the game output 1.
$(\mathsf{Final} = \mathtt{false} \wedge \mathtt{DS}_{\mathtt{reuse}} = \mathtt{false} \wedge \mathtt{DS}_{\mathtt{forge}} = \mathtt{true} \wedge \mathtt{COM}_{\mathtt{coll}} = \mathtt{false})$
$\mathtt{DS}_{\mathtt{reuse}} \leftarrow \mathtt{true}$.
Parse $\widetilde{\sigma}^{\mathsf{OS}}$ as $(\mathsf{root}^*, c^*, \sigma^{*\mathsf{DS}}, \widetilde{\mathsf{path}}, \widetilde{j}, \widetilde{r})$, $\widetilde{\sigma}'^{\mathsf{OS}}$ as $(\mathsf{root}^*, c^*, \sigma^{*\mathsf{DS}}, \widetilde{\mathsf{path}}', \widetilde{j}', \widetilde{r}')$.
If $(\widetilde{m}, \widetilde{r}) \neq (\widetilde{m}', \widetilde{r}')$, $\mathtt{COM}_{\mathtt{coll}} \leftarrow \mathtt{true}$, return the game output 1.
$(\mathsf{Final} = \mathtt{false} \wedge \mathtt{DS}_{\mathtt{reuse}} = \mathtt{true} \wedge \mathtt{DS}_{\mathtt{forge}} = \mathtt{false} \wedge \mathtt{COM}_{\mathtt{coll}} = \mathtt{true})$
Otherwise, return the game output 1.
$(\mathsf{Final} = \mathtt{false} \wedge \mathtt{DS}_{\mathtt{reuse}} = \mathtt{true} \wedge \mathtt{DS}_{\mathtt{forge}} = \mathtt{false} \wedge \mathtt{COM}_{\mathtt{coll}} = \mathtt{false})$

Fig. 8. The base game $\mathsf{Game}^{\mathsf{Base}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}$.

If B^{DS} outputs 1 with the condition where $\mathtt{DS}_{\mathtt{forge}} = \mathtt{true}$, there is the forgery $(\widetilde{\mathsf{root}}, \widetilde{c}, \widetilde{\sigma}^{\mathsf{DS}})$. Since $\mathtt{DS}_{\mathtt{forge}} = \mathtt{true}$ holds, $\mathtt{DS}_{\mathtt{reuse}} = \mathtt{false}$ holds. This fact implies that for $(m, \sigma^{\mathsf{OS}}) \in \mathbb{L}^{\mathsf{Sign}}$, the first three elements $(\mathsf{root}, c, \sigma^{\mathsf{DS}})$ of σ^{OS} are all distinct in $\mathbb{L}^{\mathsf{Sign}}$ and valid signatures for DS (i.e., $\mathsf{DS.Verify}(\mathsf{vk}^{\mathsf{DS}}, (\mathsf{root}, c), \sigma^{\mathsf{DS}}) = 1$). Moreover, B^{DS} makes q^{Sign} signing queries to signing oracle, $q^{\mathsf{Sign}} < q^{\mathsf{Fin}}$ holds where q^{Fin} is the number of entry in $\mathbb{L}^{\mathsf{Sign}}$. Hence, there is a forgery $((\widetilde{\mathsf{root}}, \widetilde{c}), \widetilde{\sigma}^{\mathsf{DS}})$ of DS. By modifying $\mathsf{Game}^{\mathsf{Base}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}$ to output this forgery $((\widetilde{\mathsf{root}}, \widetilde{c}), \widetilde{\sigma}^{\mathsf{DS}})$, we can obtain B^{DS}.

Reduction $\mathsf{B}^{\mathsf{COM}}$: A reduction $\mathsf{B}^{\mathsf{COM}}$ to the computational binding property of COM is obtained by modifying $\mathsf{Game}^{\mathsf{Base}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}$ as follows. $\mathsf{B}^{\mathsf{COM}}$ uses ck given by the strong computational binding security game of COM.

If $\mathsf{Game}_{\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}$ outputs 1 with the condition where $\mathtt{COM}_{\mathtt{coll}} = \mathtt{true}$, there is a collision $(\widetilde{m}, \widetilde{r}) \neq (\widetilde{m}', \widetilde{r}')$ such that $\mathsf{COM.Commit}(\mathsf{ck}, \widetilde{m}; \widetilde{r}) = \mathsf{COM.Commit}(\mathsf{ck}, \widetilde{m}'; \widetilde{r}')$ holds. Since if $\mathtt{COM}_{\mathtt{coll}} = \mathtt{true}$ holds, $\mathtt{DS}_{\mathtt{reuse}} = \mathtt{true}$ holds in $\mathsf{Game}^{\mathsf{Base}}_{\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}$. This fact implies that there is a pair $(\widetilde{m}, \widetilde{\sigma}^{\mathsf{OS}} = (\mathsf{root}^*, c^*, \sigma^{*\mathsf{DS}}, \widetilde{\mathsf{path}}, \widetilde{j}, \widetilde{r})) \neq (\widetilde{m}', \widetilde{\sigma}'^{\mathsf{OS}} = (\mathsf{root}^*, c^*, \sigma^{*\mathsf{DS}}, \widetilde{\mathsf{path}}', \widetilde{j}', \widetilde{r}'))$. Since $(\widetilde{m}, \widetilde{\sigma}^{\mathsf{OS}})$ and $(\widetilde{m}', \widetilde{\sigma}'^{\mathsf{OS}})$ are valid signatures, $(\widetilde{m}, \widetilde{r}) \neq (\widetilde{m}', \widetilde{r}')$ and $\mathsf{COM.Commit}(\mathsf{ck}, \widetilde{m}; \widetilde{r}) = \mathsf{COM.Commit}(\mathsf{ck}, \widetilde{m}'; \widetilde{r}')$ hold. By modifying $\mathsf{Game}^{\mathsf{Base}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}$ to output this collision $((\widetilde{m}, \widetilde{\sigma}^{\mathsf{OS}}), (\widetilde{m}', \widetilde{\sigma}'^{\mathsf{OS}}))$, we can obtain $\mathsf{B}^{\mathsf{COM}}$.

Reduction $\mathsf{B}^{\mathsf{Hash}}$: We explain how to obtain a reduction $\mathsf{B}^{\mathsf{Hash}}$ to the collision resistance property from $\mathsf{Game}^{\mathsf{Base}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}$. If $\mathsf{Game}^{\mathsf{Base}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}$ outputs 1 with the condition where $\mathsf{Final} = \mathtt{false} \wedge \mathtt{DS}_{\mathtt{reuse}} = \mathtt{false} \wedge \mathtt{DS}_{\mathtt{forge}} = \mathtt{false}$, a collision a hash function can be found. Since $\mathsf{Final} = \mathtt{false} \wedge \mathtt{DS}_{\mathtt{reuse}} = \mathtt{false} \wedge \mathtt{DS}_{\mathtt{forge}} = \mathtt{false}$ holds, then $(q^{\mathsf{Sign}}, *, \mathsf{root}^*, c^*, \sigma^{*\mathsf{DS}}) \in \mathbb{T}$ holds. Let $(M_{q^{\mathsf{Sign}}}, c_{q^{\mathsf{Sign}}})$ be an input for the q^{Sign}-th $\mathcal{O}^{\mathsf{Sign}}$ query. Then, by the computation of $\mathcal{O}^{\mathsf{Sign}}$ and table $\mathbb{T}$, $c^* = c_{q^{\mathsf{Sign}}}$, $(\mathsf{root}^*, \mathsf{tree}^*) = \mathsf{MerkleTree}^H(M_{q^{\mathsf{Sign}}})$, and $\mathsf{DS.Verify}(\mathsf{vk}^{\mathsf{DS}}, (\mathsf{root}^*, c^*), \sigma^{*\mathsf{DS}}) = 1$ holds. Since $m^* \notin M_{q^{\mathsf{Sign}}}$, a collision of a hash function H can be computed by $(x, x') \leftarrow \mathsf{Ext}_1(\mathsf{tree}^*, (m^*, \mathsf{path}^*, i^*))$. We modify $\mathsf{Game}^{\mathsf{Base}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}$ to output this collision (x, x') in this case.

If $\mathsf{B}^{\mathsf{Base}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}$ outputs 1 with the condition where $\mathtt{DS}_{\mathtt{reuse}} = \mathtt{true} \wedge \mathtt{DS}_{\mathtt{forge}} = \mathtt{false} \wedge \mathtt{COM}_{\mathtt{coll}} = \mathtt{false}$ (regardless of the bool value Final), a collision of a hash function can be also found. Since $\mathtt{DS}_{\mathtt{reuse}} = \mathtt{false} \wedge \mathtt{COM}_{\mathtt{coll}} = \mathtt{false}$ holds, then there is a pair $(\widetilde{m}, \widetilde{\sigma}^{\mathsf{OS}} = (\mathsf{root}^*, c^*, \sigma^{*\mathsf{DS}}, \widetilde{\mathsf{path}}, \widetilde{j}, \widetilde{r})) \neq (\widetilde{m}, \widetilde{\sigma}'^{\mathsf{OS}} = (\mathsf{root}^*, c^*, \sigma^{*\mathsf{DS}}, \widetilde{\mathsf{path}}', \widetilde{j}', \widetilde{r}))$ holds. From this fact, we can see that $(\widetilde{\mathsf{path}}, \widetilde{j}) \neq (\widetilde{\mathsf{path}}', \widetilde{j}')$ holds. If $j^* \neq \widetilde{j}$ holds, we can obtain a collision of a hash function H as $(x, x') \leftarrow \mathsf{Ext}_1(\mathsf{tree}^*, (m^*, \mathsf{path}^*, i^*))$. If $j^* = \widetilde{j}$ holds, then $\widetilde{\mathsf{path}} = \widetilde{\mathsf{path}}'$ holds and thus we can compute a collision of a hash function as $(x, x') \leftarrow \mathsf{Ext}_2(m, j^*, \widetilde{\mathsf{path}}, \widetilde{\mathsf{path}}')$. We modify $\mathsf{Game}^{\mathsf{Base}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}$ to output this collision (x, x') in these case.

By reduction algorithms B^{DS}, $\mathsf{B}^{\mathsf{COM}}$, and $\mathsf{B}^{\mathsf{Hash}}$ described above, we can bound the advantage $\mathsf{Adv}^{\mathsf{Seq\text{-}sEUFCMA}}_{(1,n)\text{-}\mathsf{OS},\mathsf{A}}(1^\lambda)$ as

$$
\begin{aligned}
&\mathsf{Adv}^{\mathsf{Seq\text{-}sEUFCMA}}_{(1,n)\text{-}\mathsf{OS},\mathsf{A}}(1^\lambda) \\
&= \Pr[\mathsf{Game}^{\mathsf{Seq\text{-}sEUFCMA}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}(1^\lambda) \Rightarrow 1] \\
&= \Pr[\mathsf{Game}^{\mathsf{Base}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}(1^\lambda) \Rightarrow 1] \\
&= \Pr[\mathsf{Game}^{\mathsf{Base}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}(1^\lambda) \Rightarrow 1 \wedge \mathtt{DS}_{\mathtt{forge}} = \mathtt{true}] \\
&\quad + \Pr[\mathsf{Game}^{\mathsf{Base}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}(1^\lambda) \Rightarrow 1 \wedge \mathtt{COM}_{\mathtt{coll}} = \mathtt{true}] \\
&\quad + \Pr[\mathsf{Game}^{\mathsf{Base}}_{(1,n)\text{-}\mathsf{OS}_{\mathsf{Ours}},\mathsf{A}}(1^\lambda) \Rightarrow 1 \wedge \mathtt{DS}_{\mathtt{forge}} = \mathtt{false} \wedge \mathtt{COM}_{\mathtt{coll}} = \mathtt{false}] \\
&\leq \mathsf{Adv}^{\mathsf{sEUFCMA}}_{\mathsf{DS},\mathsf{A}_1}(1^\lambda) + \mathsf{Adv}^{\mathsf{sBind}}_{\mathsf{COM},\mathsf{A}_2}(1^\lambda) + \mathsf{Adv}^{\mathsf{Coll}}_{\mathcal{H},\mathsf{A}_3}(1^\lambda).
\end{aligned}
$$

By this fact, we can conclude Theorem 2. □

5 Conclusion

Summary of Our Results. In this paper, we revisit the unforgeability security for a 1-out-of-n oblivious signature scheme and point out problems. By reflecting on these problems, we define the Seq-sEUF-CMA security. We propose the improved generic construction of a 1-out-of-n oblivious signature scheme $(1,n)$-$\mathsf{OS}_{\mathsf{Ours}}$. Compared to the construction by Zhou et al. [21], our construction offers a smaller second message size. The sum of a second message size and a signature size is improved from $O(n)$ to $O(\log n)$.

Discussion of Our Security Model and Open Problem. We introduce the Seq-sEUF-CMA security in Definition 10. This security model restricts an adversary A to execute signing interactions only in a sequential manner. It is natural to consider a model that allows concurrent signing interactions. However, if we straightforwardly extend our security model to a concurrent setting, there is a trivial attack.

Let us consider the following adversary A that runs signing protocol executions twice concurrently. A chooses two list $M_1 = (m_{1,0}, \ldots, m_{1,n-1})$ and $M_2 = (m_{2,0}, \ldots, m_{2,n-1})$ such that $M_1 \cap M_2 = \emptyset$ (i.e., there is no element m such that $m \in M_1 \wedge m \in M_2$). In the 1st interaction, A chooses $m_{1,0} \in M_1$, obtains a signature σ_1 on a message $m_{1,0}$. In the 2nd interaction A chooses $m_{2,0} \in M_2$, obtains a signature σ_2 on a message $m_{2,0}$. A finishs the 1st interaction by outputting $(m_{2,0}, \sigma_2)$. Since $m_{2,0} \notin M_1$, A trivially wins the unforgeability game.

Due to this trivial attack, we cannot straightforwardly extend our security model to the concurrent signing interaction setting. We leave an open problem to define the unforgeability security model for a 1-out-of-n oblivious signature scheme that supports concurrent signing interactions.

References

1. M. Abe and E. Fujisaki. How to date blind signatures. In K. Kim and T. Matsumoto, editors, *Advances in Cryptology - ASIACRYPT '96, International Con-*

ference on the Theory and Applications of Cryptology and Information Security, Kyongju, Korea, November 3-7, 1996, Proceedings, volume 1163 of *Lecture Notes in Computer Science*, pages 244–251. Springer, 1996.

2. S. Agrawal, E. Kirshanova, D. Stehlé, and A. Yadav. Practical, round-optimal lattice-based blind signatures. In H. Yin, A. Stavrou, C. Cremers, and E. Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 39–53. ACM, 2022.
3. M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.
4. P. Blaskiewicz, L. Hanzlik, K. Kluczniak, L. Krzywiecki, M. Kutylowski, M. Slowik, and M. Wszola. Pseudonymous signature schemes. In K. Li, X. Chen, and W. Susilo, editors, *Advances in Cyber Security: Principles, Techniques, and Applications*, pages 185–255. Springer, 2019.
5. D. Chaum. Blind signatures for untraceable payments. In D. Chaum, R. L. Rivest, and A. T. Sherman, editors, *Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982*, pages 199–203. Plenum Press, New York, 1982.
6. L. Chen. Oblivious signatures. In D. Gollmann, editor, *Computer Security - ESORICS 94, Third European Symposium on Research in Computer Security, Brighton, UK, November 7-9, 1994, Proceedings*, volume 875 of *Lecture Notes in Computer Science*, pages 161–172. Springer, 1994.
7. S. Chiou and J. Chen. Design and implementation of a multiple-choice e-voting scheme on mobile system using novel t -out-of- n oblivious signature. *J. Inf. Sci. Eng.*, 34(1):135–154, 2018.
8. R. del Pino and S. Katsumata. A new framework for more efficient round-optimal lattice-based (partially) blind signature via trapdoor sampling. In Y. Dodis and T. Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 306–336. Springer, 2022.
9. G. Fuchsbauer, C. Hanser, C. Kamath, and D. Slamanig. Practical round-optimal blind signatures in the standard model from weaker assumptions. In V. Zikas and R. D. Prisco, editors, *Security and Cryptography for Networks - 10th International Conference, SCN 2016, Amalfi, Italy, August 31 - September 2, 2016, Proceedings*, volume 9841 of *Lecture Notes in Computer Science*, pages 391–408. Springer, 2016.
10. G. Fuchsbauer, C. Hanser, and D. Slamanig. Practical round-optimal blind signatures in the standard model. In R. Gennaro and M. Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 233–253. Springer, 2015.
11. E. Ghadafi. Efficient round-optimal blind signatures in the standard model. In A. Kiayias, editor, *Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers*, volume 10322 of *Lecture Notes in Computer Science*, pages 455–473. Springer, 2017.

12. L. Hanzlik and K. Kluczniak. A short paper on blind signatures from knowledge assumptions. In J. Grossklags and B. Preneel, editors, *Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers*, volume 9603 of *Lecture Notes in Computer Science*, pages 535–543. Springer, 2016.
13. L. Hanzlik and K. Kluczniak. Two-move and setup-free blind signatures with perfect blindness. In J. Baek and R. Zhang, editors, *Proceedings of the 4th ACM International Workshop on ASIA Public-Key Cryptography, APKC@AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2, 2017*, pages 1–11. ACM, 2017.
14. L. Hanzlik, J. Loss, and B. Wagner. Rai-choo! evolving blind signatures to the next level. In C. Hazay and M. Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 753–783. Springer, 2023.
15. S. Katsumata, R. Nishimaki, S. Yamada, and T. Yamakawa. Round-optimal blind signatures in the plain model from classical and quantum standard assumptions. In A. Canteaut and F. Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 404–434. Springer, 2021.
16. V. Lyubashevsky, N. K. Nguyen, and M. Plançon. Efficient lattice-based blind signatures via gaussian one-time signatures. In G. Hanaoka, J. Shikata, and Y. Watanabe, editors, *Public-Key Cryptography - PKC 2022 - 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8-11, 2022, Proceedings, Part II*, volume 13178 of *Lecture Notes in Computer Science*, pages 498–527. Springer, 2022.
17. R. C. Merkle. A digital signature based on a conventional encryption function. In C. Pomerance, editor, *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer, 1987.
18. C. Song, X. Yin, and Y. Liu. A practical electronic voting protocol based upon oblivious signature scheme. In *2008 International Conference on Computational Intelligence and Security, CIS 2008, 13-17 December 2008, Suzhou, China, Volume 1 - Conference Papers*, pages 381–384. IEEE Computer Society, 2008.
19. R. Tso, T. Okamoto, and E. Okamoto. 1-out-of-n oblivious signatures. In L. Chen, Y. Mu, and W. Susilo, editors, *Information Security Practice and Experience, 4th International Conference, ISPEC 2008, Sydney, Australia, April 21-23, 2008, Proceedings*, volume 4991 of *Lecture Notes in Computer Science*, pages 45–55. Springer, 2008.
20. J. You, Z. Liu, R. Tso, Y. Tseng, and M. Mambo. Quantum-resistant 1-out-of-n oblivious signatures from lattices. In C. Cheng and M. Akiyama, editors, *Advances in Information and Computer Security - 17th International Workshop on Security, IWSEC 2022, Tokyo, Japan, August 31 - September 2, 2022, Proceedings*, volume 13504 of *Lecture Notes in Computer Science*, pages 166–186. Springer, 2022.
21. Y. Zhou, S. Liu, and S. Han. Generic construction of 1-out-of-n oblivious signatures. *IEICE Trans. Inf. Syst.*, 105-D(11):1836–1844, 2022.