Signer and Message Ambiguity from a Variety of Keys

George Teşeleanu[1,2,a]
[1]Advanced Technologies Institute, 10 Dinu Vintilă, Bucharest, Romania
[2]Simion Stoilow Institute of Mathematics of the Romanian Academy, 21 Calea Grivitei, Bucharest, Romania

Keywords: Signer and Message Ambiguity, 1-out-of-n Signature, Oblivious Signature.

Abstract: A signer and message ambiguous signature enables a recipient to request a signer to sign a sensible message such that the signer cannot guess what message he signed and the receiver cannot deduce the signer's identity. In this work, we formalize this type of signature, introduce the corresponding security requirements and describe two instantions. The first one assumes that the signer hides his identity in n independently generated public keys, while the second one assumes that all n public keys share the same public parameters.

1 INTRODUCTION

In order to address the increasing interest in privacy protection, Chen (Chen, 1994) introduced the concept of oblivious signatures. He considered two such classes. The first one is an oblivious signature scheme with n keys, while the second one is an oblivious signature with n messages.

In the case of oblivious signatures with n keys, we have n signers $\mathcal{S}_0, \ldots, \mathcal{S}_{n-1}$ (or a signer with n different keys) and a recipient $\mathcal{R}$. A high level description of the protocol is the following:

- the recipient chooses a message m and can get it signed with one of the n keys;
- the signers, even the holder of the accepted key, do not have an idea on who really signed m;
- when necessary, $\mathcal{R}$ can show that he received a valid signature from one of the n signers.

On the other hand, in the version of oblivious signatures with n messages we have only one signer $\mathcal{S}$ and the main features are the following:

- the recipient chooses n messages $m_0, \ldots, m_{n-1}$ and can get only one signed;
- the signer cannot deduce which message he actually signed;
- when necessary, $\mathcal{R}$ can show that he received a valid signature on one of the n messages.

Remark that in both cases the signer(s) can read the received message(s) and decide if he(they) agree(s) with the content before signing it. The two concepts can also be mixed and thus obtain an oblivious protocol with n_1 messages and n_2 keys. Some examples of oblivious protocols can be found in (Chen, 1994, Tso et al., 2008, Tso, 2016, Tso, 2019).

Blind signatures (Chaum, 1982, Juels et al., 1997) share the same privacy goal as oblivious signatures with n messages. More precisely, both signatures allow users to request a signature without revealing the exact message to the signer. The main difference is that in the case of blind signature the signer is not aware of the message's content, while in the case of oblivious signatures the signer sees a message pool that contains n messages. Hence, oblivious signatures offer a guarantee to the signer that no message outside the pool will be signed and thus can be considered an improvement of blind signatures.

In contrast to oblivious signatures with n keys, an 1-out-of-n signature convinces a verifier that a message was signed by one of n possible independent signers without allowing the verifier to deduce which signer it was. Hence, the privacy requirement is shifted from the signers to the verifier. Also, in this case, only the actual signer decides if he agrees to the message's content, while the remaining $n-1$ signers have access to the message only after the signing process is over. Some examples can be found in (Abe et al., 2002, Cramer et al., 1994, Rivest et al., 2001).

In some applications we encounter situations where a mixture of oblivious signatures with n_2 messages and 1-out-of-n_1 signatures is required. Hence, a receiver wants to hide his request, while the signer wants to keep its anonymity. We further call this type

[a] https://orcid.org/0000-0003-3953-2744

Teşeleanu, G.
Signer and Message Ambiguity from a Variety of Keys.
DOI: 10.5220/0010502903950402
In *Proceedings of the 18th International Conference on Security and Cryptography (SECRYPT 2021)*, pages 395-402
ISBN: 978-989-758-524-1

of signatures as signer and message ambiguous signatures.

A possible usage for these signatures is the following. Multiple small companies[1] contribute with servers to a storage pool and split the profits according the contributed storage space. A client wants to make a query to this cluster, but wants to be able to prove to a third party that the answer is authentic. Therefore, the cluster has to sign the answer. But the customer must be oblivious of which company is hosting the corresponding data. Hence, the cluster can use an 1-out-of-n_1 signature to hide the exact location of the data. On the other hand, the client wants to hide the exact content of his query. Thus, he can hide his query into $n_2 - 1$ unrelated queries. In this case, we can see that a mixture of an 1-out-of-n_1 signature and an n_2 message oblivious signature can offer a possible solution.

In this paper, we propose the first signer and message ambiguous signatures, one in the key separable model (*i.e.* the users' use independently generated public parameters) and one in the non-separable model (*i.e.* the users' public parameters are identical). In the separable model, we used the zero-knowledge version of Abe *et. al* signature (Abe et al., 2002) in conjunction with a generalized and modified Tso *et. al* signature (Tso et al., 2008). In the non-separable model, we used the same signature based on Tso *et. al*, but we combined it with a generalized version of Abe *et. al* signature (Abe et al., 2002). The formalization method used for generalizing the signatures is similar to the approach described in (Maurer, 2009).

Structure of the Paper. We introduce notations and definitions used throughout the paper in Section 2. In Sections 3 and 4 we present our main results, namely two signer and message ambiguous signatures, one in the separable model and one in the non-separable model. Their performance is analysed in Section 5. We conclude in Section 6.

2 PRELIMINARIES

Notations. Throughout the paper, the notation $|S|$ denotes the cardinality of a set S. The action of selecting a random element x from a sample space X is denoted by $x \stackrel{\$}{\leftarrow} X$, while $x \leftarrow y$ represents the assignment of value y to variable x. The probability of the event E to happen is denoted by $Pr[E]$. The subset $\{0,\ldots,s-1\} \in \mathbb{N}$ is denoted by $[0,s)$. Note we further consider that all of $\mathcal{N}$'s subsets are of the form $[0,s)$ and $n_2 \leq s$. A vector v of length n is denoted either $v = (v_0,\ldots,v_{n-1})$ or $v = \{v_i\}_{i\in[0,n)}$. Also, we use the notations C^n_k to denote binomial coefficients and exp to denote Euler's constant.

[1] Each with its unique public certificate.

2.1 Groups

Let $(\mathbb{G},\star)$ and $(\mathbb{H},\otimes)$ be two groups. We assume that the group operations $\star$ and $\otimes$ are efficiently computable.

Let $f : \mathbb{G} \to \mathbb{H}$ be a function (not necessarily one-to-one). We say that f is a homomorphism if $f(x \star y) = f(x) \otimes f(y)$. Throughout the paper we consider f to be a one-way function, *i.e.* it is infeasible to compute x from $f(x)$. To be consistent with (Maurer, 2009), we denote by $[x]$ the value $f(x)$. Note that given $[x]$ and $[y]$ we can efficiently compute $[x\star y] = [x] \otimes [y]$, due to the fact that f is a homomorphism.

2.2 Signer and Message Ambiguous Signatures

Based on the formal models defined in (Abe et al., 2002,Tso et al., 2008,Tso, 2016), we introduce signer and message ambiguous signatures (SMAS) and their corresponding security models. Hence, a SMAS involves three types of entities:

- A signature requester $\mathcal{R}$. For any list of public keys L and any list of messages M, $\mathcal{R}$ can choose any message from M to get signed by any of the signers from L. Note that $\mathcal{R}$ is not able to learn which signer from L actually signed the message.
- An ambiguous signer $\mathcal{S}$. One of the signers from L proceeds to sign the message chosen by $\mathcal{R}$, but he is not able to learn which message from M has actually been signed.
- A verifier $\mathcal{V}$. $\mathcal{R}$ converts the SMAS into a signer ambiguous signature σ and transmits σ to $\mathcal{V}$. The verifier is able to check the validity of σ without modifying the verification algorithm of the original signer ambiguous signature.

Definition 2.1 (Signer and Message Ambiguous Signature)**.** A signer and message ambiguous signature scheme is a digital signature comprised of the following algorithms:

Setup(λ): On input a security parameter λ, this algorithm outputs the private and public keys (sk_i, pk_i) of all the participants and the public parameters $pp = (\mathcal{M},\mathcal{S})$, where $\mathcal{M}$ is the message space and $\mathcal{S}$ is the signature space.

Signature Generation(): An interactive protocol between $\mathcal{R}$ and $\mathcal{S}$. In the first step, the recipient takes as input a list of public keys L and sends to the signer a list of messages M and some additional information A. Then, $\mathcal{S}$ takes as input a list of messages M, the information A, the private key sk_k and a list of public keys L such that $pk_k \in L$ and sends a list of signatures $\mathcal{W}$ to $\mathcal{R}$. After receiving $\mathcal{W}$, the recipient uses A to convert the SMAS into a signer ambiguous signature σ for a message $m \in M$ and then outputs (m, σ, L).

Verify(m, σ, L): An algorithm that on input a message m, a signature σ and a list of public keys L outputs either `true` or `false`.

The following definitions capture the intuitive notions of signer and message ambiguity. In Definitions 2.2 and 2.3 we assume that the attacker is $\mathcal{R}$ and, respectively, $\mathcal{S}$.

Definition 2.2 (Signer Ambiguity)**.** Let $\mathcal{L} = \{pk_i\}_{i\in[0,n_1)}$, where pk_i are generated by the *Setup* algorithm. Also, for a set $L \subseteq \mathcal{L}$ we define $\tilde{L} = \{sk_k \mid sk_k \text{ is the secret key corresponding to } pk_k \in L\}$. A SMAS is perfectly signer ambiguous if for any list of messages M and their corresponding additional information A, any $L \subseteq \mathcal{L}$, any $sk_k \in \tilde{L}$ and any signature $\mathcal{W}$ generated by $\mathcal{S}(M, A, sk_k, L)$, any unbounded adversary $\mathcal{A}$ outputs an sk such that $sk = sk_k$ with probability exactly $1/|L|$.

Definition 2.3 (Message Ambiguity)**.** A SMAS is perfectly message ambiguous if for any list of messages M and their corresponding additional information A and for any message $m_\ell \in M$ chosen by $\mathcal{R}$ to be signed, any unbounded adversary $\mathcal{A}$ outputs an m such that $m = m_\ell$ with probability exactly $1/|M|$.

The security requirement for $\mathcal{S}$ is unforgeability of signatures, even when the signature receiver is the adversary.

Definition 2.4 (Existential Unforgeability against Adaptive Chosen Message and Chosen Public Key Attacks - EUF-CMCPA)**.** The notion of unforgeability for signatures is defined in terms of the following security game between the adversary $\mathcal{A}$ and a challenger:

1. The *Setup* algorithm is run and all the public parameters are provided to $\mathcal{A}$.
2. For any list of messages M and any subset of $\mathcal{L} = \{pk_i\}_{i\in[0,n_1)}$, $\mathcal{A}$ can fix a message $m_\ell \in M$ and request the signature associated to m_ℓ to the challenger.
3. Finally, $\mathcal{A}$ outputs a signature (m, σ, L), where $L \subseteq \mathcal{L}$.

$\mathcal{A}$ wins the game if *Verify*$(m, \sigma, L) = \texttt{true}$, $L \subseteq \mathcal{L}$ and $\mathcal{A}$ did not query the challenger on any pair (M, L) such that $m_\ell = m$. We say that a signature scheme is unforgeable when the success probability of $\mathcal{A}$ in this game is negligible.

We further introduce the notions of a Boolean matrix and of a heavy row in such a matrix (Ohta and Okamoto, 1998). These definitions are then used in stating the heavy row lemma (Ohta and Okamoto, 1998).

Definition 2.5 (Boolean Matrix of Random Tapes)**.** Let us consider a matrix M whose rows consist of all possible random choices of an adversary and the columns consist of all possible random choices of a challenger. Its entries are 0 if the adversary fails the game and 1 otherwise.

Definition 2.6 (Heavy Row)**.** A row of M is heavy if the fraction of 1's along the row is at least $\varepsilon/2$, where ε is the adversary's success probability.

Lemma 2.1 (Heavy Row Lemma). The 1's in M are located in heavy rows with a probability of at least $1/2$.

3 SMAS WITH KEY SEPARATION

3.1 Description

By modifying the protocol from (Tso et al., 2008) and endowing it with the technique described in (Abe et al., 2002) we developed a SMAS in the separable model. Note that in a separable scheme each key pair can be generated by a different scheme, under a different hardness assumption. In practice, each user can use different trusted third parties (TTP) to generate their public parameters and key pair. To simplify description we present the *Setup* algorithm as a centralized algorithm. We will denote the following signature with SMAS-KSS.

Setup(λ): Let $i \in [0, n_1)$. Choose for each user two groups $\mathbb{G}_i$, $\mathbb{H}_i$, a homomorphism $[\cdot]_i : \mathbb{G}_i \to \mathbb{H}_i$ and a hash function $H_i : \{0,1\}^* \to \mathcal{C}_i \subseteq \mathbb{N}$. Note that we require that $|\mathbb{G}_i| \geq 2^\lambda$. Choose $a_i, x_i \xleftarrow{\$} \mathbb{G}_i$ and compute $y_i \leftarrow [x_i]_i$ and $b_i \leftarrow [a_i]_i$. Output the public key $pk_i = y_i$. The secret key is $sk_i = x_i$. The elements b_i are known to all participants, but the a_i's are used only once and are discarded afterwards.

Listing()*:* Collect the public keys and randomly shuffle them. Store the result into a list $\mathcal{L} = \{y_j\}_{j\in[0,n_1)}$ and output $\mathcal{L}$.[2]

[2] Note that $\mathcal{L}$ can be fixed or periodically updated.

Signature Generation(): Assume that recipient $\mathcal{R}$ would like to get a signature from signer $\mathcal{S}$ on a message $m_\ell \in \{m_t\}_{t\in[0,n_2)}$. To compute the ambiguous signature the following protocol is executed:

Step 1: For $j \in [0,n_1)$, $\mathcal{R}$ selects $\alpha_j \xleftarrow{\$} \mathbb{G}_j$ and computes $c_j \leftarrow [\alpha_j]_j \otimes_j b_j^\ell$. Then, $\mathcal{R}$ sends $C = \{c_j\}_{j\in[0,n_1)}$ and $M = \{m_t\}_{t\in[0,n_2)}$ to $\mathcal{S}$.

Step 2: For $t \in [0,n_2)$, $\mathcal{S}$ (with access to x_k) does the following:

a) Generate an element $\beta_t \xleftarrow{\$} \mathbb{G}_k$ and compute $z_{k,t} \leftarrow c_k \otimes_k b_k^{-t} \otimes_k [\beta_t]_k$ and $d_{k+1,t} \leftarrow H_{k+1}(\mathcal{L}, m_t, z_{k,t})$.

b) For $j \in [k+1,n_1) \cup [0,k)$, randomly select $s_{j,t} \xleftarrow{\$} \mathbb{G}_j$ and then compute $z_{j,t} \leftarrow c_j \otimes_j b_j^{-t} \otimes_j [s_{j,t}]_j \otimes_j y_j^{d_{j,t}}$ and $d_{j+1,t} \leftarrow H_{j+1}(\mathcal{L}, m_t, z_{j,t})$[3].

c) Compute $s_{k,t} \leftarrow \beta_t \star_k x_k^{-d_{k,t}}$.

d) Send to $\mathcal{R}$ the signature $(d_{0,t}, \mathcal{W}_t)$, where $\mathcal{W}_t = \{s_{j,t}\}_{j\in[0,n_1)}$.

Step 3: For $j \in [0,n_1)$ and $t \in [0,n_2)$, $\mathcal{R}$ computes $\delta_{j,t} \leftarrow [\alpha_j]_j \otimes_j b_j^{\ell-t}$, $e_{j,t} \leftarrow \delta_{j,t} \otimes_j [s_{j,t}]_j \otimes_j y_j^{d_{j,t}}$ and then $d_{j+1,t} \leftarrow H_{j+1}(\mathcal{L}, m_t, e_{j,t})$ if $j \neq n_1 - 1$. $\mathcal{R}$ accepts the ambiguous signature if and only if $d_{0,t} = H_0(\mathcal{L}, m_t, e_{n_1-1,t})$, where $t \in [0,n_2)$. Otherwise, output `false`.

Step 4: To convert the signer and message ambiguous signature into a signer ambiguous signature, $\mathcal{R}$ sets $d_0 \leftarrow d_{0,\ell}$ and computes $s_j \leftarrow \alpha_j \star_j s_{j,\ell}$, where $j \in [0,n_1)$. Output the signature $(d_0, \mathcal{W})$, where $\mathcal{W} = \{s_j\}_{j\in[0,n_1)}$.

Verify$(m, d_0, \mathcal{W}, \mathcal{L})$: For $j \in [0,n_1)$, compute $e_j \leftarrow [s_j]_j \otimes_j y_j^{d_j}$ and then $d_{j+1} \leftarrow H_{j+1}(\mathcal{L}, m, e_j)$ if $j \neq n_1 - 1$. Output `true` if and only if $d_0 = H_0(\mathcal{L}, m, e_{n_1-1})$. Otherwise, output `false`.

Remark. In the *Setup* phase, the b_i elements can be generated for each user after they receive their public parameters and key-pairs, and by a TTP different from the one generating the initial system's parameters. Thus, our scheme is compatible with preexisting signature certificates and can be seen as adding an extra functionality to existing systems.

Correctness. First we need to check that $\mathcal{R}$ accepts a genuine signature. Thus, if $(c_{0,t}, \mathcal{W}_t)$ is generated according to the scheme, then for $j \neq k$ we have

$$\begin{aligned} e_{j,t} &= \delta_{j,t} \otimes_j [s_{j,t}]_j \otimes_j y_j^{d_{j,t}} \\ &= c_j \otimes_j b_j^{-t} \otimes_j [s_{j,t}]_j \otimes_j y_j^{d_{j,t}} \\ &= z_{j,t} \end{aligned}$$

and for $j = k$ we have

$$\begin{aligned} e_{k,t} &= \delta_{j,t} \otimes_k [s_{k,t}]_k \otimes_k y_k^{d_{k,t}} \\ &= c_k \otimes_k b_k^{-t} \otimes_k [\beta_t \star_k x_k^{-d_{k,t}}]_k \otimes_k y_k^{d_{k,t}} \\ &= c_k \otimes_k b_k^{-t} \otimes_k [\beta_t]_k \otimes_k [x_k]_k^{-d_{k,t}} \otimes_k y_k^{d_{k,t}} \\ &= c_k \otimes_k b_k^{-t} \otimes_k [\beta_t]_k \\ &= z_{k,t}. \end{aligned}$$

Now we need to check if the verification process returns `true`. Hence, if the pair $(d_0, \mathcal{W})$ is generated according to the scheme, then we have

$$\begin{aligned} e_j &= [s_j]_j \otimes_j y_j^{d_j} \\ &= [\alpha_j \star_j s_{j,\ell}]_j \otimes_j y_j^{d_j} \\ &= \delta_{j,\ell} \otimes_j [s_{j,\ell}]_j \otimes_j y_j^{d_j} \\ &= e_{j,\ell}. \end{aligned}$$

3.2 Security Analysis

Theorem 3.1. The SMAS-KSS scheme is perfectly signer ambiguous.

Proof. Note that all $s_{j,t}$ are taken randomly from $\mathbb{G}_j$, except for $s_{k,t}$. Since β_t is a random element from $\mathbb{G}_k$, then $s_{k,t}$ is also randomly distributed in $\mathbb{G}_k$. Hence, for a fixed $(m_t, \mathcal{L})$ the probability of $\mathcal{W}_t$ is always $1/\prod|G_i|$, regardless of the closing point $s_{k,t}$ and index t. The remaining $c_{0,t}$ are uniquely determined from $(m_t, \mathcal{L})$ and $\mathcal{W}_t$. □

Theorem 3.2. The SMAS-KSS scheme is perfectly message ambiguous.

Proof. All the information regarding ℓ is contained in the c_j elements. Since α_j is random, then c_j is also random. Thus, for a fixed M the probability of $\{c_j\}_{j\in[0,|M|)}$ is always $1/\prod|G_i|$. So, no information about ℓ is leaked to $\mathcal{S}$. □

Theorem 3.3. If the following statements are true

- an EUF-CMCPA attack on the SMAS-KSS has non-negligible probability of success in the ROM,
- for all i values, $f_i \in \mathbb{Z}$ are known such that $\gcd(d_0 - d_1, f_i) = 1$ for all $d_0, d_1 \in \mathcal{C}_i$ with $d_0 \neq d_1$,

[3]When $j = n_1 - 1$ we abuse notation and consider $j + 1 = 0$.

- for all i values, $u_i \in \mathbb{G}_i$ are known such that $[u_i]_i = y_i^{f_i}$,

then at least a homomorphism $[\cdot]_i$ can be inverted in polynomial time.

Proof. Let $\mathcal{A}$ be an efficient EUF-CMCPA attacker for SMAS-KSS that requests at most q_s and q_h signing and, respectively, random oracle queries. Also, let ε be its success probability and τ its running time. By q_m we denote the total number of messages sent to $\mathcal{S}$ for signing.

In order to make $\mathcal{A}$ work properly we simulate the random oracles that correspond to each hash function (see Algorithm 1) and the signing oracle (see Algorithm 2). For simplicity we treat all the random oracles as one big random oracle O_H that takes as input the j-th query (i, L_j, m_j, r_j) and returns a random value corresponding to $H_i(L_j, m_j, r_j)$. To avoid complicated suffixes y_0 and m_0, for example, refer to the first public key and the first message from the current L_j and, respectively, M_j. Hence, $y_0 \in L_j$ and $y_0 \in L_{j'}$ could differ. The same is also true for m_0.

Algorithm 1: Hashing oracle O_H simulation for all H_i.

Input: A hashing query (i, L_j, m_j, r_j) from $\mathcal{A}$
if $\exists h_j$ such that $\{L_j, m_j, r_j, h_j\} \in T_i$ **then**
  $e \leftarrow h_j$
else
  $e \xleftarrow{\$} \mathcal{C}_i$
  Append $\{L_j, m_j, r_j, e\}$ to T_i
end if
return e

The signing oracle O_S fails and returns $\perp$ only if we cannot assign $d_{0,t}$ to $(L_j, m_j, e_{|L_j|-1,t})$ without causing an inconsistency in T_0. This event happens with probability at most q_h/q, where $q = 2^\lambda$. Thus, O_S is successful with probability at least $(1 - q_h/q)^{q_s q_m} \geq 1 - q_h q_s q_m / q$.

Let Θ and Ω be the random tapes given to O_S and $\mathcal{A}$. The adversary's success probability is taken over the space defined by Θ, Ω and O_H. Let Σ be the set of (Θ, Ω, O_H) with which $\mathcal{A}$ successfully creates a forgery, while having access to a real signing oracle. Let $(m, d_0, \{s_i\}_{i\in[0,n')}, L)$ be $\mathcal{A}$'s forgery, where $|L| = n'$. Then, T_{i+1} contains a query for (L, m, e_i) for all $i \in [0, n')$ with probability at least $1 - 1/|\mathcal{C}_{i+1}|$, due to the ideal randomness of O_H. Let $\Sigma' \subseteq \Sigma$ be the set of (Θ, Ω, O_H) with which $\mathcal{A}$ successfully creates a forgery, while having access only to the simulated oracle O_S. Then, $Pr[(\Theta, \Omega, O_H) \in \Sigma'] \geq \varepsilon'$, where

Algorithm 2: Signing oracle O_S simulation.

Input: A signature query (M_j, C_j, L_j) from $\mathcal{A}$
for $t \in [0, |M_j|)$ **do**
  $d_{0,t} \xleftarrow{\$} \mathcal{C}_0$
  for $i \in [0, |L_j|)$ **do**
    $s_{i,t} \xleftarrow{\$} \mathbb{G}_i$
    $e_{i,t} \leftarrow c_i \otimes_i b_i^{-t} \otimes_i [s_{i,t}]_i \otimes_i y_i^{d_{i,t}}$
    if $i \neq |L_j| - 1$ **then**
      $d_{i+1,t} \leftarrow H_{i+1}(L_j, m_t, e_{i,t})$
    end if
  end for
  if $\nexists h_t$ such that $\{L_j, m_t, e_{|L_j|-1,t}, h_t\} \in T_0$ **then**
    Append $\{L_j, m_t, e_{|L_j|-1,t}, d_{0,t}\}$ to T_0
    Sent to $\mathcal{A}$ the signature $(d_{0,t}, \{s_{i,t}\}_{i\in[0,|L_j|)})$
  else
    return $\perp$
  end if
end for

$\varepsilon' = (1 - q_h q_s q_m / q)(1 - 1/w)\varepsilon$ and w is the smallest $|\mathcal{C}_i|$.

Since the queries form a ring, there exists at least an index $k \in [0, n')$ such that the u query $Q_u = (k + 1, L, m, e_k)$ and the v query $Q_v = (k, L, m, e_{k-1})$ satisfy $u \leq v$. Such a pair (u, v) is called a gap index. Remark that $u = v$ only when $n' = 1$. If there are two or more gap indices with regard to a signature, we only consider the smallest one.

We denote by $\Sigma'_{u,v}$ the set of (Θ, Ω, O_H) that yield the gap index (u, v). There are at most $C_2^{q_h} + C_1^{q_h} = q_h(q_h+1)/2$ such sets. If we invoke $\mathcal{A}$ with randomly chosen (Θ, Ω, O_H) at most $1/\varepsilon'$ times, then we will find at least one $(\Theta, \Omega, O_H) \in \Sigma'_{u,v}$ for some gap index (u, v) with probability $1 - (1 - \varepsilon')^{1/\varepsilon'} > 1 - exp(-1) > 3/5$.

We define the sets $GI = \{(u, v) \mid |\Sigma'_{u,v}|/|\Sigma'| \geq 1/(q_h(q_h+1))\}$ and $B = \{(\Theta, \Omega, O_H) \in \Sigma'_{u,v} \mid (u, v) \in GI\}$. Then, we have $Pr[B|\Sigma'] \geq 1/2$. Using the heavy row lemma we obtain that a triplet (Θ, Ω, O_H) that yields a successful run of $\mathcal{A}$ is in B with probability at least $1/2$.

Let $O_{H'}$ be the identical to O_H except for the Q_v query to which $O_{H'}$ responds with a random element $d'_k \neq d_k$. Then according to the heavy row lemma, with probability $1/2$, $(\Theta, \Omega, O_{H'})$ satisfies $Pr[(\Theta, \Omega, O_{H'}) \in \Sigma'_{u,v}] = \varepsilon''/2$, where $\varepsilon'' = \varepsilon'/(2q_h(q_h+1))$. Hence, if we run $\mathcal{A}$ at most $2/\varepsilon''$ times, then with probability $1/2 \cdot [1 - (1 - \varepsilon''/2)^{2/\varepsilon''}] > 1/2 \cdot (1 - exp(-1)) > 3/10$ we will find at least one d'_k such that $(\Theta, \Omega, O_{H'}) \in \Sigma'_{u,v}$. Since Q_u is queried before Q_v, e_k remains unchanged. Therefore we can compute

$$\tilde{x}_k = u_k^a \star_k (s_k'^{-1} \star_k s_k)^b,$$

where a and b are computed using Euclid's algorithm such that $f_k a + (d'_k - d_k)b = 1$. Note that for some β

$$\begin{aligned}[s_k'^{-1} \star_k s_k]_k &= [s_k'^{-1}]_k \otimes_k [s_k]_k \\ &= y_k^{d'_k} \otimes_k ([\beta]_k)^{-1} \otimes c_k^{-1} \otimes c_k \otimes_k [\beta]_k \otimes_k y_k^{-d_k} \\ &= y_k^{d'_k - d_k}\end{aligned}$$

and thus

$$\begin{aligned}[\tilde{x}_k]_k &= [u_k^a \star_k (s_k'^{-1} \star_k s_k)^b]_k \\ &= ([u_k]_k)^a \otimes_k ([s_k'^{-1} \star_k s_k]_k)^b \\ &= (y_k^{f_k})^a \otimes_k (y_k^{d'_k - d_k})^b \\ &= y_k.\end{aligned}$$

The overall success probability is $9/100 = 3/5 \cdot 1/2 \cdot 3/10$ and $\mathcal{A}$ is invoked at most $1/\varepsilon' + 2/\varepsilon''$ times. $\square$

3.3 Concrete Examples

In this subsection we present a few concrete examples of the SMAS in order to help readers who are familiar with Schnorr or Guillou-Quisquater type signatures. The reader can easily infer more examples from the unified zero-knowledge protocol's instantiations described in (Maurer, 2009, Teşeleanu, 2018).

All Discrete Logarithm Case. Let p and q be two prime numbers such that $q|p-1$. Select an element $h \in \mathbb{H}_p$ of order q in some multiplicative group of order $p-1$. The discrete logarithm of an element $z \in \mathbb{H}_p$ is an exponent x such that $z = h^x$. We further describe the parameters of the all discrete logarithm signature.

Define $(\mathbb{G}_i, \star_i) = (\mathbb{Z}_{q_i}, +)$ and $\mathbb{H}_i = \langle h_i \rangle$. The one-way group homomorphism is defined by $[x_i]_i = h_i^{x_i}$ and the challenge space $\mathcal{C}_i$ can be any arbitrary subset of $[0, q_i)$. Let 1_i be the neutral element of $\mathbb{H}_i$. Then the conditions of Theorem 3.3 are satisfied for $f_i = q_i$ and $u_i = 0$. Note that we have $[u]_i = [0]_i = 1_i = y_i^{f_i} = y_i^{q_i}$ since every element of $\mathbb{H}_i$ raised to the group order q_i is the neutral element 1_i.

All e^{th}-root Case. Let p and q be two safe prime numbers such that $(p-1)/2$ and $(q-1)/2$ are also prime. Compute $N = pq$ and choose a prime e such that $\gcd(e, \varphi(N)) = 1$. An e^{th}-root of an element $z \in \mathbb{Z}_N^*$ is a base x such that $z = x^e$. Note that the e^{th}-root is not unique. We further describe the parameters of the all e^{th}-root signature.

Define $(\mathbb{G}_i, \star_i) = (\mathbb{H}_i, \otimes_i) = (\mathbb{Z}_{N_i}^*, \cdot)$. The one-way group homomorphism is defined by $[x_i]_i = x_i^{e_i}$ and the challenge space $\mathcal{C}_i$ can be any arbitrary subset of $[0, e_i)$. The conditions of Theorem 3.3 are satisfied for $f_i = e_i$ and $u_i = y_i$.

Mixture of Discrete Logarithm and e^{th}-root. For simplicity, we consider the case $n = 2$. Let $(\mathbb{G}_0, \star_0) = (\mathbb{Z}_q, +)$, $\mathbb{H}_0 = \langle h \rangle$ and $(\mathbb{G}_1, \star_1) = (\mathbb{H}_1, \otimes_1) = (\mathbb{Z}_N^*, \cdot)$. The one-way group homomorphisms are defined by $[x_0]_0 = h^{x_0}$ and $[x_1]_1 = x_1^e$. The corresponding challenge spaces $\mathcal{C}_0$ and $\mathcal{C}_1$ can be any arbitrary subset of $[0, q)$ and, respectively, $[0, e)$. Finally, the conditions of Theorem 3.3 are satisfied for $f_0 = q$, $f_1 = e$, $u_0 = 0$ and $u_1 = y_1$.

4 SMAS WITHOUT KEY SEPARATION

4.1 Description

In this section we present a more efficient SMAS signature. This signature only works when all the participants use the same underlying commutative group. To achieve our goal, we used a generalized version of the technique developed in (Abe et al., 2002). We further denote the following signature with SMAS-NKSS.

Setup(λ): Choose two commutative groups $\mathbb{G}$, $\mathbb{H}$, a homomorphism $[\cdot] : \mathbb{G} \to \mathbb{H}$ and a hash function $H : \{0,1\}^* \to \mathcal{C} \subseteq \mathbb{N}$. Note that we require that $|\mathbb{G}| \geq 2^\lambda$. Choose $a \xleftarrow{\$} \mathbb{G}$ and compute $b \leftarrow [a]$. For each user, choose $x_i \xleftarrow{\$} \mathbb{G}$ and compute $y_i \leftarrow [x_i]$. Output the public key $pk_i = y_i$. The secret key is $sk_i = x_i$. The element b is known to all participants, but a is used only once and is discarded afterwards.

Listing$()$: Collect the public keys and randomly shuffle them. Store the result into a list $\mathcal{L} = \{y_j\}_{j \in [0, n_1)}$ and output $\mathcal{L}$.

Signature Generation$()$: Assume that recipient $\mathcal{R}$ would like to get a signature from signer $\mathcal{S}$ on a message $m_\ell \in \{m_t\}_{t \in [0, n_2)}$. To compute the ambiguous signature the following protocol is executed:

Step 1: $\mathcal{R}$ selects $\alpha \xleftarrow{\$} \mathbb{G}$ and computes $c \leftarrow [\alpha] \otimes b^\ell$. Then, $\mathcal{R}$ sends c and $M = \{m_t\}_{t \in [0, n_2)}$ to $\mathcal{S}$.

Step 2: For $t \in [0, n_2)$, $\mathcal{S}$ generates a random element $\beta_t \xleftarrow{\$} \mathbb{G}$ and $d_{j,t} \xleftarrow{\$} \mathcal{C}$, where $j \in [0, n_1) \setminus$

$\{k\}$. Then computes the following:

$$z_t \leftarrow c \otimes b^{-t} \otimes [\beta_t] \otimes y_0^{d_{0,t}} \otimes \ldots \otimes y_{k-1}^{d_{k-1,t}}$$
$$z_t \leftarrow z \otimes y_{k+1}^{d_{k+1,t}} \otimes \ldots \otimes y_{n_1-1}^{d_{n_1-1,t}}$$
$$d_t \leftarrow H(\mathcal{L}, m_t, z_t)$$
$$d_{k,t} \leftarrow d_t - d_{0,t} - \ldots - d_{k-1,t} \bmod |\mathcal{C}|$$
$$d_{k,t} \leftarrow d_{k,t} - d_{k+1,t} - \ldots - d_{n_1-1,t} \bmod |\mathcal{C}|$$
$$s_t \leftarrow \beta_t \star x_k^{-d_{k,t}}.$$

Send to $\mathcal{R}$ the signature $(s_t, \mathcal{W}_t)$, where $\mathcal{W}_t = \{d_{j,t}\}_{j\in[0,n_1)}$.

Step 3: For $t \in [0,n_2)$, $\mathcal{R}$ computes $\delta_t \leftarrow [\alpha] \otimes b^{\ell-t}$, $u_t \leftarrow \sum_{j=0}^{n_1-1} d_{j,t} \bmod |\mathcal{C}|$ and $v_t \leftarrow \delta_t \otimes [s_t] \otimes (\otimes_{j=0}^{n_1-1} y_j^{d_{j,t}})$. $\mathcal{R}$ accepts the ambiguous signature if and only if $u_t \equiv H(\mathcal{L}, m_t, v_t) \bmod |\mathcal{C}|$, where $t \in [0,n_2)$. Otherwise, output `false`.

Step 4: To convert the signer and message ambiguous signature into a signer ambiguous signature, $\mathcal{R}$ computes $s \leftarrow \alpha \star s_\ell$ and sets $d_j \leftarrow d_{j,\ell}$, where $j \in [0,n_1)$. Output the signature $(s, \mathcal{W})$, where $\mathcal{W} = \{d_j\}_{j\in[0,n_1)}$.

Verify$(m, s, \mathcal{W}, \mathcal{L})$: Compute the intermediary values $u \leftarrow \sum_{j=0}^{n_1-1} d_j \bmod c$ and $v \leftarrow [s] \otimes (\otimes_{j=0}^{n_1-1} y_j^{d_j})$. Output `true` if and only if $u = H(\mathcal{L}, m, v)$. Otherwise, output `false`.

Correctness. First we need to check that $\mathcal{R}$ accepts a genuine signature. Thus, if $(s_t, \mathcal{W}_t)$ is generated according to the scheme, then we have

$$\begin{aligned} v_t &= \delta_t \otimes [s_t] \otimes (\otimes_{j=0}^{n-1} y_j^{d_{j,t}}) \\ &= c \otimes b^{-t} \otimes [\beta_t] \otimes [x_k]^{-d_{k,t}} \otimes (\otimes_{j=0}^{n-1} y_j^{d_{j,t}}) \\ &= z_t. \end{aligned}$$

Now we need to check if the verification process returns `true`. Hence, if the pair $(s, \mathcal{W})$ is generated according to the scheme, then we have

$$\begin{aligned} v &= [s] \otimes (\otimes_{j=0}^{n-1} y_j^{d_j}) \\ &= [\alpha \star s_\ell] \otimes (\otimes_{j=0}^{n-1} y_j^{d_{j,\ell}}) \\ &= \delta_\ell \otimes [s_\ell] \otimes (\otimes_{j=0}^{n-1} y_j^{d_{j,\ell}}) \\ &= v_\ell. \end{aligned}$$

4.2 Security Analysis

Theorems 4.1 and 4.2's proofs are similar to Theorems 3.1 and 3.2's proofs and thus are omitted. Theorem 4.3's proof is provided in the full version of the paper and is omitted due to space limitations.

Theorem 4.1. The SMAS-NKSS scheme is perfectly signer ambiguous.

Theorem 4.2. The SMAS-NKSS scheme is perfectly message ambiguous.

Theorem 4.3. If the following statements are true

- an EUF-CMCPA attack on the SMAS-NKSS has non-negligible probability of success in the ROM,
- an $f \in \mathbb{Z}$ is known such that $\gcd(d_0 - d_1, f) = 1$ for all $d_0, d_1 \in \mathcal{C}$ with $d_0 \neq d_1$,
- for all i values, $u_i \in \mathbb{G}$ are known such that $[u_i] = y_i^f$,

then the homomorphism $[\cdot]$ can be inverted in polynomial time.

5 PERFORMANCE ANALYSIS

When $n_1 = 1$ and $n_2 = n$ both SMAS schemes become an oblivious signature with n messages (denoted simply as SMAS). Two such signatures are described in (Tso et al., 2008, Chen, 1994) for $\mathbb{G} = \mathbb{Z}_p^*$. In **??** and table 1 we provide the reader with the performance analysis of our scheme. In **??** the communication overhead is measured in bits, while in Table 1 the computation cost is measured in exponentiations. We consider $|p| = 3072$ and $|q| = 256$, which according to (Elaine, 2020) offers a security strength of 128 bits.

Table 1: Computation cost comparison.

Scheme	$\mathcal{S}$	$\mathcal{R}$	$\mathcal{V}$
SMAS	$2n$	$3n+2$	2
Tso *et. al* (Tso et al., 2008)	$2n$	$3n+2$	2
Chen (Chen, 1994)	$3n$	$2n+10$	8

In order to measure the efficiency of our SMAS schemes, we compare them to the protocol described in (Tso, 2016). Although the philosophy of this scheme is a little bit different than ours, it is the closest one. Again, let $\mathbb{G} = \mathbb{Z}_p^*$. The results are presented in Tables 2 and 3. Note that in Tso's protocol, the receiver transforms the signature into a Schnorr signature[4] (Schnorr, 1989), while we transform it into an Abe *et.al* signature (Abe et al., 2002). Hence, the larger communication and computational overhead on $\mathcal{V}$'s side.

[4] *i.e.* $n_1 = 1$

Table 2: Communication cost comparison.

Scheme	Steps	$\mathcal{R} \to \mathcal{S}$	$\mathcal{S} \to \mathcal{R}$	$\mathcal{R} \to \mathcal{V}$
SMAS-KSS	2	$n_1\|q\|$	$(n_1+1)n_2\|q\|$	$(n_1+1)\|q\|$
SMAS-NKSS	2	$\|q\|$	$(n_1+1)n_2\|q\|$	$(n_1+1)\|q\|$
Tso (Tso, 2016)	2	$\|q\|$	$2n_1n_2\|q\|$	$2\|q\|$

Table 3: Computation cost comparison.

Scheme	$\mathcal{S}$	$\mathcal{R}$	$\mathcal{V}$
SMAS-KSS	$3n_1n_2 - n_2$	$3n_1n_2 + 2n_1$	$2n_1$
SMAS-NKSS	$(n_1+1)n_2$	$(n_1+2)n_2+2$	n_1+1
Tso (Tso, 2016)	$2n_1n_2$	$3n_1n_2+2$	2

6 CONCLUSION

Our SMAS protocols are the abstraction of a large class of protocols that allow users to sign sensible information, while maintaining the signers anonymity. We introduced two versions, one with independently selected public parameters and one with common public parameters. We managed to relate the presented protocols' security to the hardness of inverting one-way homomorphisms.

REFERENCES

Abe, M., Ohkubo, M., and Suzuki, K. (2002). 1-out-of-n Signatures from a Variety of Keys. In *ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 415–432. Springer.

Chaum, D. (1982). Blind Signatures for Untraceable Payments. In *CRYPTO 1982*, pages 199–203. Plenum Press, New York.

Chen, L. (1994). Oblivious Signatures. In *ESORICS 1994*, volume 875 of *Lecture Notes in Computer Science*, pages 161–172. Springer.

Cramer, R., Damgård, I., and Schoenmakers, B. (1994). Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocolss. In *CRYPTO 1994*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer.

Elaine, B. (2020). NIST Special Publication 800-57 Part 1 Revision 5 - Recommendation for Key Management: Part 1 – General. Technical report, NIST.

Juels, A., Luby, M., and Ostrovsky, R. (1997). Security of Blind Digital Signatures. In *CRYPTO 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 150–164. Springer.

Maurer, U. (2009). Unifying Zero-Knowledge Proofs of Knowledge. In *AFRICACRYPT 2009*, volume 5580 of *Lecture Notes in Computer Science*, pages 272–286. Springer.

Ohta, K. and Okamoto, T. (1998). On Concrete Security Treatment of Signatures Derived from Identification. In *CRYPTO 1998*, volume 1462 of *Lecture Notes in Computer Science*, pages 354–369. Springer.

Rivest, R. L., Shamir, A., and Tauman, Y. (2001). How to Leak a Secret. In *ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer.

Schnorr, C.-P. (1989). Efficient Identification and Signatures For Smart Cards. In *CRYPTO 1989*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer.

Teşeleanu, G. (2018). Unifying Kleptographic Attacks. In *NordSec 2018*, volume 11252 of *Lecture Notes in Computer Science*, pages 73–87. Springer.

Tso, R. (2016). Two-in-One Oblivious Signatures Secure in the Random Oracle Model. In *NSS 2016*, volume 9955 of *Lecture Notes in Computer Science*, pages 143–155. Springer.

Tso, R. (2019). Two-in-One Oblivious Signatures. *Future Generation Computer Systems*, 101:467–475.

Tso, R., Okamoto, T., and Okamoto, E. (2008). 1-out-of-n Oblivious Signatures. In *ISPEC 2008*, volume 4991 of *Lecture Notes in Computer Science*, pages 45–55. Springer.