On Concrete Security Treatment of Signatures Based on Multiple Discrete Logarithms

George Teşeleanu[1,2]

[1] Advanced Technologies Institute
10 Dinu Vintilă, Bucharest, Romania
tgeorge@dcti.ro

[2] Simion Stoilow Institute of Mathematics of the Romanian Academy
21 Calea Grivitei, Bucharest, Romania

Abstract. In this paper, we present a generalization of Schnorr's digital signature that allows a user to simultaneously sign multiple messages. Compared to Schnorr's scheme that concatenates messages and then signs them, the new protocol takes advantage of multiple threads to process messages in parallel. We prove the security of our novel protocol and discuss different variants of it. Last but not least, we extend Ferradi *et al.*'s co-signature protocol by exploiting the inherent parallelism of our proposed signature scheme.

Keywords: digital signatures, Schnorr signature, zero knowledge protocols, multi-message signature

1 Introduction

Schnorr's signature scheme was introduced in [28] and was proven secure in [23, 26, 27] using the random oracle paradigm. Pointcheval and Stern [26, 27] analyze Schnorr's signature directly and use the forking lemma to prove it secure. A different approach is used in [23]. Instead of analyzing the signature, Ohta and Okamoto [23] study the security of the underlying primitive and then reduce the security of the signature to that of the primitive (*e.g.* the identification protocol used to derive Schnorr's signature). Therefore, they obtain a more natural way to prove the signature's security, since analyzing the corresponding identification protocol is easier.

In this paper, we present a generalization of Schnorr's signature, which enables users to simultaneously sign multiple messages. Based on Ohta and Okamoto's work [23], we adapt their reduction technique to a multiple-messages signature and then use it to prove the security of our proposal. Our solution enhances the efficiency of the signing process by using multiple threads. In this scenario, each message is processed by a separate thread, allowing us to accelerate the signing and verification process. By employing this approach, we reduce the running time compared to concatenating all the messages and signing them using Schnorr's signature. This multithreading technique optimizes the performance of our proposal, making it a possible solution for fast and efficient signature processing. To the best of our knowledge, this is the first signature proposal that uses parallelism to accelerate the signing of multiple messages.

We also propose a different use case in the field of contract signing. When designing contract signing protocols, we have several design categories to choose from: gradual release [13, 15, 17, 25], optimistic [5, 7, 22] and concurrent [9, 29] or legally fair [12, 19] models. Compared to older paradigms such as gradual release or optimistic models, concurrent signatures or legally fair protocols do not rely on trusted third parties and require less interaction between co-signers. Therefore, this design category is more attractive to users. In consequence, in this paper we only consider legally fair co-signing protocols, excluding the older solutions.

In general, a co-signing protocol involves two mutually distrustful signing partners *Alice* and *Bob*, who wish to sign a common contract. Using our multi-message signature, we managed to adapt the solution presented in [12] to the multi-message scenario. In addition to the speed advantage inherited from our signature proposal, the new co-signature scheme offers an additional advantage. Typically, when two companies want to enter into an agreement, the process involves multiple contracts that need signing. Moreover, the agreement phase for each contact can involve multiple negotiations, making the signing of all contracts a sequential process. Using our proposal, *Alice* and *Bob* can start the signing process and handle contracts as they become ready. Once all the negotiations are concluded, they can complete the signing protocol and output a single signature for all the documents.

Structure of the paper. We establish the theoretical framework needed for our proposals in Section 2. In Section 3 we introduce our security reduction technique from a multi-message signature's security to a multi-challenge protocol's security. Our proposed multi-challenge identification protocol is presented in Section 4. In Section 5 we present our main result, a multi-message signature protocol. Following Ferradi *et al.*'s co-signature protocol [12], in Section 6 we describe a multi-message co-signature protocol. We conclude in Section 7.

Notations. In this paper, λ represents a security parameter. The notation $|S|$ denotes the cardinality of a set S. The subset $\{1, \ldots, s\} \in \mathbb{N}$ is denoted by $[1, s]$. The action of selecting a random element x from a sample space X is represented by $x \xleftarrow{\$} X$, while $x \leftarrow y$ indicates the assignment of value y to variable x. The concatenation of two strings a and b is denoted by $a \| b$. Multidimensional vectors $v = (v_1, \ldots, v_s)$ are represented as $v = \{v_i\}_{i \in [1,s]}$. For a probabilistic polynomial time machine we use the abbreviation PPT. The base of the natural logarithm is denoted by e.

2 Framework

In this section we define a special class of signature schemes that support the signing of multiple messages at once. In order to prove their security, we reduce the breaking of the signature primitive to a type of identification scheme that allows the prover to vary the number of challenges sent to the verifier. Finally, we introduce the framework needed for co-signature schemes based on multi-messages signatures. Note that when we restrict ourselves to only one message, the notions introduced in this section coincide with the classical notions for identification [23], signature [23] and co-signature [12] schemes and their corresponding security models.

2.1 Multi-Challenge Identification Scheme

Definition 1 (Multi-Challenge Identification Scheme). *A multi-challenge identification scheme between a prover P and a verifier V is composed of the following*

Key generation: *Let $n > 0$ be an integer. The prover P interrogates a key generation algorithm $\mathcal{G}$ which on input λ outputs n pairs $\{(K_p^i, K_s^i)\}_{i\in[1,n]}$. The prover's public key is $\{K_p^i\}_{i\in[1,n]}$, while the corresponding secret key is $\{K_s^i\}_{i\in[1,n]}$.*

Identification protocol: *P proves his identity to V as follows*

- **Step 1.** *P selects an integer $\ell \in [1, n]$, generates X from a random string R and sends them to V.*
- **Step 2.** *Let $\mathcal{E}$ be the challenge space. V randomly generates $E_i \xleftarrow{\$} \mathcal{E}$ for $i \in [1, \ell]$ and sends the challenge $\{E_i\}_{i\in[1,\ell]}$ to P.*
- **Step 3.** *P generates an answer Y from $(\{K_s^i\}_{i\in[1,\ell]}, R, \{E_i\}_{i\in[1,\ell]})$ and sends it to V.*
- **Step 4.** *V checks the validity of the relations of $(\{K_p^i\}_{i\in[1,\ell]}, X, \{E_i\}_{i\in[1,\ell]}, Y)$.*

We further assume that multi-challenge identification schemes are perfect zero knowledge against an honest verifier. We recall the definition [16, 21] of a zero knowledge protocol next.

Definition 2 (Zero Knowledge Protocol). *A protocol (P, V) is zero-knowledge if for every efficient program $\bar{V}$ there exists an efficient program S, the simulator, such that the output of S is indistinguishable from a transcript of the protocol execution between P and $\bar{V}$. If the indistinguishability is perfect[3], then the protocol is called perfect zero-knowledge.*

Definition 3 (Soundness). *A PPT adversary A breaks a multi-challenge identification scheme with (t, ε) if and only if A as a prover can cheat an honest verifier V with a success probability greater than ε within processing time t. Note that the probability is taken over the coin flips of A and V, and A does not conduct any active attack.*

A multi-challenge identification scheme is (t, ε)-secure if and only if there is no adversary that can break it with (t, ε).

[3] i.e. the probability distribution of the simulated and the actual transcript are identical

An essential tool for constructing a zero knowledge simulator, is a special property called c-simulatability. More precisely, it is sufficient to check that a protocol is c-simulatable for one round of the protocol in order to prove that it is zero knowledge. We recall these results next [21].

Definition 4. *A three-move protocol round*[4] *with challenge space $\mathcal{E}$ is c-simulatable if for any value $E_i \in \mathcal{E}$ one can efficiently generate a triple $(X, \{E_i\}_{i\in[1,\ell]}, Y)$ with the same distribution as occurring in the protocol.*

Theorem 1. *A protocol consisting of c-simulatable three-move rounds, with uniformly chosen challenge from a polynomially bounded*[5] *challenge space $\mathcal{E}^\ell$, is perfect zero-knowledge.*

To show that the multi-challenge identification protocols proposed by us are secure we will reduce their security to a key searching problem. Therefore, we further introduce a definition from [23].

Definition 5 (Key Searching Problem). *A PPT adversary A breaks a key searching problem if and only if A can find the secret key from a public key with a success probability greater than ε within processing time t. Note that the probability is taken over the coin flips of A.*

A key searching problem is (t, ε)-secure if and only if there is no adversary that can break it with (t, ε).

We further introduce the notions of a Boolean matrix [23] and of an τ-heavy row in such a matrix. These definitions are then used in stating the τ-heavy row lemma. Note that when $\tau = 2$ we obtain the concept of a heavy row and the heavy row lemma presented in [10, 23].

Definition 6 (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 7 (τ-Heavy Row). *Let $\tau > 1$ be an integer. A row of M is τ-heavy if the fraction of 1's along the row is at least ε/τ, where ε is the adversary's success probability.*

Lemma 1 (τ-Heavy Row Lemma). *The 1's in M are located in τ-heavy rows with a probability of at least $(\tau - 1)/\tau$.*

Proof. Let M' be the sub-matrix of M consisting of all the rows that are not τ-heavy. Let μ and μ' be the number of entries in M and M', respectively. Using our assumption we have that the number of 1's in M and M' is $\mu\varepsilon$ and smaller than $\mu'\varepsilon/\tau$, respectively. Therefore, the number of 1's located in the τ-heavy rows h satisfies

$$h > \mu\varepsilon - \frac{\mu'\varepsilon}{\tau} \geq \mu\varepsilon - \frac{\mu\varepsilon}{\tau} = \mu\varepsilon \cdot \frac{\tau - 1}{\tau},$$

as desired. ∎

[4] P sends X, V sends $\{E_i\}_{i\in[1,\ell]}$, P sends Y

[5] per round

2.2 Multi-Message Signatures Scheme

Definition 8 (Multi-Message Signature Scheme). *A multi-message signature scheme consists of the following three PPT algorithms*

Key generation: *Let $n > 0$ be an integer. The P interrogates a key generation algorithm $\mathcal{G}$ which on input λ outputs n pairs $\{(K_p^i, K_s^i)\}_{i\in[1,n]}$. The signer's public key is $\{K_p^i\}_{i\in[1,n]}$, while the corresponding secret key is $\{K_s^i\}_{i\in[1,n]}$.*

Signature generation: *Let $\ell \in [1, n]$. P generates the signature of his messages $M = \{m_i\}_{i\in[1,\ell]}$ using ℓ distinct public random oracle functions $\{F_i\}_{i\in[1,\ell]}$ as follows: P generates X from a random string R, accesses the random oracle functions to get $E_i \leftarrow F_i(X, m_i) \in \mathcal{E}$ for $i \in [1, \ell]$, computes Y using $\{K_s^i\}_{i\in[1,\ell]}$ and R, $\{E_i\}_{i\in[1,\ell]}$, and publishes the signature (X, M, Y)*[6]*.*

Signature verification: *A verifier V checks the validity of the signature by the relations of $(\{K_p^i\}_{i\in[1,\ell]}, X, \{E_i\}_{i\in[1,\ell]}, Y)$ and $E_i \leftarrow F_i(X, m_i)$ for $i \in [1, \ell]$.*

Definition 9 (Multi-Message Signature Unforgeability). *In a chosen multi-message attack, a PPT adversary A can ask the legitimate user P to sign up to q_{sig} chosen message vectors and output their signatures. We also allow the adversary to invoke the F_i random oracle up to q_{F_i} times.*

We say that A breaks a multi-message signature with $(t, q_{sig}, \{q_{F_i}\}_{i\in[1,n]}, \varepsilon)$ if and only if A can forge a signature of a message vector M with success probability greater than ε within processing time t. Note that the probability is taken over the coin flips of A, F and the signing oracle P. We also impose the restriction that M was never queried to P.

A multi-message signature scheme is $(t, q_{sig}, \{q_{F_i}\}_{i\in[1,n]}, \varepsilon)$-secure if and only if there is no adversary that can break it with $(t, q_{sig}, \{q_{F_i}\}_{i\in[1,n]}, \varepsilon)$.

2.3 Multi-Message Co-Signatures Scheme

Multi-message co-signatures have the same structure as the multi-message signatures. The main differences are that the signature generation is computed jointly by two users and that the signature verification is checked using the joint public key of two users.

In the case of multi-message co-signatures, adversary A can perform the following queries

Random oracle queries: A can request the value of $F_i(x)$ for an x of his choosing.

Sign queries: A can request user C a valid signature (X, Y) for a message vector $\{m_i\}_{i\in[1,\ell]}$ and a public key $\{K_{C,p}^i\}_{i\in[1,n]}$ of his choosing.

CoSign queries: A can request a valid co-signature (X, Y) from users C and D for a message vector $\{m_i\}_{i\in[1,\ell]}$ and a common public key $\{K_{C\|D,p}^i\}_{i\in[1,n]}$ of his choosing.

---

[6] We will simply denote it by (X, Y) when it is clear from the context for which M it was generated.

Transcript queries: A can request a valid transcript of the co-signing protocol for a message vector $\{m_i\}_{i\in[1,\ell]}$ of his choosing, between users C and D of his choosing.

SKExtract queries: A can request the private key corresponding to a public key.

Directory queries: A can request the public key of any user.

Definition 10 (Multi-Message Co-Signature Unforgeability). *The notion of unforgeability for co-signatures is defined in terms of the following security game between the adversary A and a challenger:*

1. *The key generation algorithm is run and all the public parameters are provided to A.*
2. *A can perform any number of queries to the challenger, as described above.*
3. *Finally, A outputs a tuple $(X, \{m_i\}_{i\in[1,\ell]}, Y)$.*

A wins the game if the verification algorithm outputs `true` *and there exist public keys $K_C = \{K^i_{C,p}\}_{i\in[1,n]}$ and $K_D = \{K^i_{D,p}\}_{i\in[1,n]}$ such that $K^i_{C\|D,p} = K^i_{C,p}K^i_{D,p}$ for all i and either of the following holds*

- *A did not query SKExtract on the public keys K_C nor on K_D, and did not query CoSign on $(\{m_i\}_{i\in[1,\ell]}, \{K^i_{C\|D,p}\}_{i\in[1,n]})$, and did not query Transcript on $(\{m_i\}_{i\in[1,\ell]}, K_C, K_D)$ nor $(\{m_i\}_{i\in[1,\ell]}, K_D, K_C)$.*
- *A did not query Transcript on $(\{m_i\}_{i\in[1,\ell]}, K_C, K_i)$ for any $K_i \neq K_C$ and did not query SKExtract on K_C, and did not query CoSign on $(\{m_i\}_{i\in[1,\ell]}, K_C, K_i)$ for any $K_i \neq K_C$.*

We say that a co-signature scheme is unforgeable when the success probability of A in this game is negligible.

The second constrain imposed in Definition 10 corresponds to the situation where the adversary is one of the signers (*i.e.* $A = C$ or D), and thus A knows one of the secret keys.

3 Reduction Lemma

In this section we introduce a technique that reduces the security of multi-message signatures to multi-challenge identification schemes. When $n = 1$ we obtain the result proven in [23]. Since we generalize the result of Ohta and Okamoto [23], our proof is based on their original proof, with necessary modifications to accommodate the new functionality. Note that we assume uniform coin flips over $\mathcal{E}$. We further assume, without loss of generality, that on query $i \in [1, q_{sig}]$ adversary A wants to sign ℓ_i messages and when it outputs the forgery it uses ℓ messages.

Theorem 2. *Let*

$$\varepsilon \geq (\max_i q_{F_i}) \cdot \left(\frac{\tau(\ell+1)}{|\mathcal{E}|^\ell} + \frac{q_{sig}\cdot \max_i \ell_i}{|\mathcal{E}|}\right) + \frac{1-|\mathcal{E}|^\ell}{1-|\mathcal{E}|}\cdot\frac{1}{|\mathcal{E}|^\ell}.$$

1. *If A_1 breaks a multi-message signature with $(t, q_{sig}, \{q_{F_i}\}_{i\in[1,n]}, \varepsilon)$ there exists A_2 which breaks the multi-message signature with $(t, q_{sig}, \{1\}_{i\in[1,n]}, \varepsilon')$, where*

$$\varepsilon' = \frac{1}{(\max_i q_{F_i})^\ell}\left(\varepsilon - \frac{1-|\mathcal{E}|^\ell}{1-|\mathcal{E}|}\cdot\frac{1}{|\mathcal{E}|^\ell}\right).$$

2. *If A_2 breaks a multi-message signature with $(t, q_{sig}, \{1\}_{i\in[1,n]}, \varepsilon')$ there exists A_3 which breaks the multi-message signature with $(t', 0, \{1\}_{i\in[1,n]}, \varepsilon'')$, where*

$$\varepsilon'' = \varepsilon' - \frac{q_{sig}\cdot \max_i \ell_i}{|\mathcal{E}|}$$

and $t' = t +$ the simulation time of q_{sig} signatures.

3. *If A_3 breaks a multi-message signature with $(t', 0, \{1\}_{i\in[1,n]}, \varepsilon'')$ there exists A_4 which breaks the corresponding multi-challenge identification protocol with (t', ε'').*

Proof. Item 1. Let $Q_{i,j}$ be the j-th query from A_1 to the i-th random oracle F_i and $\rho_{i,j}$ be the j-th answer from F_i to A_1. We construct an adversary B using A_1 as follows

Step 1. Randomly select n integers j_i such that $1 \le j_i \le q_{F_i}$.
Step 2. Run A_1 with the random oracles $\{F_i\}_{i\in[1,n]}$ and obtain the values $(X, \{m_i\}_{i\in[1,\ell]}, \{E_i\}_{i\in[1,\ell]}, Y)$.
Step 3. If for any i we have that $(X, m_i) = Q_{i,j_i}$ and $E_i = \rho_{i,j_i}$ then output the forged signature $(X, \{m_i\}_{i\in[1,\ell]}, Y)$. Otherwise, output $\perp$.

If adversary A_1 succeeds in forging a signature $(X, \{m_i\}_{i\in[1,\ell]}, Y)$, then there are $\ell + 1$ cases

- the values (X, m_i) were not asked to the random oracle F_i for any $i \in [1, \ell]$;
- (X, m_1) was asked as the j-th query to oracle F_1, where $j \in [1, q_{F_1}]$ and the remaining values (X, m_i) were not asked to the random oracle F_i for any $i \in [2, \ell]$;
- (X, m_i) was asked as the j-th query to oracle F_i, where $j \in [1, q_{F_i}]$ and $i \in [1, 2]$ and the remaining values (X, m_i) were not asked to the random oracle F_i for any $i \in [3, \ell]$;
- ...
- (X, m_i) was asked as the j-th query to oracle F_i, where $j \in [1, q_{F_i}]$ and $i \in [1, \ell]$.

We denote the last ℓ cases by F. Then, the success probability of A_1 when F is at most

$$Pr[F] \le \frac{1-|\mathcal{E}|^\ell}{1-|\mathcal{E}|}\cdot\frac{1}{|\mathcal{E}|^\ell}$$

due to the randomness of F_i.

Let $Pr[A_1]$ be the probability that A_1 succeeds. In the first case, B's success probability $Pr[B]$ is

$$\begin{aligned}
Pr[B] &\geq \sum_{j_1=0}^{q_{F_1}} \sum_{j_2=0}^{q_{F_2}} \cdots \sum_{j_\ell=0}^{q_{F_\ell}} Pr[(j_1, j_2, \ldots, j_\ell) \text{ is selected}] Pr[A_1 \& ((X, m_i) = Q_{i,j_i}, \forall i)] \\
&= \sum_{j_1=0}^{q_{F_1}} \sum_{j_2=0}^{q_{F_2}} \cdots \sum_{j_\ell=0}^{q_{F_\ell}} \frac{1}{q_{F_1} q_{F_2} \cdots q_{F_\ell}} Pr[A_1 \& ((X, m_i) = Q_{i,j_i}, \forall i)] \\
&= \frac{1}{q_{F_1} q_{F_2} \cdots q_{F_\ell}} \sum_{j_1=0}^{q_{F_1}} \sum_{j_2=0}^{q_{F_2}} \cdots \sum_{j_\ell=0}^{q_{F_\ell}} Pr[A_1 \& ((X, m_i) = Q_{i,j_i}, \forall i)] \\
&\geq \frac{1}{(\max_i q_{F_i})^\ell} \sum_{j_1=0}^{q_{F_1}} \sum_{j_2=0}^{q_{F_2}} \cdots \sum_{j_\ell=0}^{q_{F_\ell}} Pr[A_1 \& ((X, m_i) = Q_{i,j_i}, \forall i)] \\
&= \frac{1}{(\max_i q_{F_i})^\ell} (Pr[A_1] - Pr[A_1 \& F]) \\
&\geq \frac{1}{(\max_i q_{F_i})^\ell} \left(\varepsilon - \frac{1 - |\mathcal{E}|^\ell}{1 - |\mathcal{E}|} \cdot \frac{1}{|\mathcal{E}|^\ell} \right),
\end{aligned}$$

where for the last inequality we used the fact that $Pr[A_1 \& F] < Pr[F]$.

Using B we construct adversary A_2 as follows

Step 1. Randomly select n integers j_i such that $1 \leq j_i \leq q_{F_i}$.
Step 2. Run A_1 with the random oracles $\{F_i\}_{i \in [1,n]}$ and the random tapes $\{\Theta_i\}_{i \in [1,n]}$, and obtain the values $(X, \{m_i\}_{i \in [1,\ell]}, \{E_i\}_{i \in [1,\ell]}, Y)$, where only the j_i query is asked to F_i and the remaining $q_{F_i} - 1$ queries to Θ_i. Here Θ_i contains $q_{F_i} - 1$ random blocks used as answers to Θ_i.
Step 3. If for any i we have that $(X, m_i) = Q_{i,j_i}$ and $E_i = \rho_{i,j_i}$ then output the forged signature $(X, \{m_i\}_{i \in [1,\ell]}, Y)$. Otherwise, output $\perp$.

Remark that A_1 cannot distinguish $q_{F_i} - 1$ random blocks of Θ_i from $q_{F_i} - 1$ answers from F_i due to the randomness of F_i. Therefore, A_2 has the same probability of success as B.

Item 2. We construct adversary A_3 using A_2 as follows

Step 1. For $k = 1$ to q_{sig} do
  Step a. Run A_2 with simulated $(X_j, \{m_{i,j}\}_{i \in [1,\ell_j]}, \{E_{i,j}\}_{i \in [1,\ell_j]}, Y_j)$ for $j \in [1, k-1]$ and get a message $\{m_{i,k}\}_{i \in [1,\ell_k]}$ chosen by A_2 whose signature is requested to the signer.
  Step b. Simulate $(X_k, \{m_{i,k}\}_{i \in [1,\ell_k]}, \{E_{i,k}\}_{i \in [1,\ell_k]}, Y_k)$ by the standard perfect zero knowledge protocol simulation technique of the corresponding identification scheme with an honest verifier. If there exist an integer $j < k$ such that $X_j = X_k$, discard X_k and repeat this step.
Step 2. Run algorithm A_2 with random oracles $\{F_i\}_{i \in [1,n]}$ and simulated values $(X_j, \{m_{i,j}\}_{i \in [1,\ell_j]}, \{E_{i,j}\}_{i \in [1,\ell_j]}, Y_j)$ for $j \in [1, q_{sig}]$ and get the signature $(X, \{m_i\}_{i \in [1,\ell]}, Y)$.

Step 3. Output the forged signature $(X, \{m_i\}_{i\in[1,\ell]}, Y)$.

If A_2 does not ask $(X_j, m_{i,j})$, where $j \in [1, q_{sig}]$, to F_i, then the adversary cannot distinguish the simulated environment from a legitimate one due to the perfect zero knowledge protocol simulation and the randomness of oracle F_i. Therefore, A_3's success probability $Pr[A_3]$ is

$$\begin{aligned}
Pr[A_3] &= Pr[A_2\&((X_j, m_{i,j}) \neq (A_2\text{'s query to } F_i)\forall i \in [1, \ell_j] \text{ and } \forall j \in [1, q_{sig}])] \\
&= Pr[A_2] - Pr[\exists i \in [1, \ell_j]\&\exists j \in [1, q_{sig}]\&((X_j, m_{i,j}) = (A_2\text{'s query to } F_i)] \\
&\geq \varepsilon' - \frac{\ell_1 + \ell_2 + \ldots + \ell_{q_{sig}}}{|\mathcal{E}|} \\
&\geq \varepsilon' - \frac{q_{sig} \cdot \max_i \ell_i}{|\mathcal{E}|}
\end{aligned}$$

and its running time is $t' = t +$ the simulation time of q_{sig} signatures.

Item 3. Let Q_i be a query from A_3 to the i-th random oracle F_i and ρ_i be the answer from F_i to A_3. Then A_4 interacts with an honest verifier V as follows

Step 1. Run A_3 and for all i get the query $Q_i = (X, m_i)$ to F_i.
Step 2. Send (ℓ, X) to V and get a challenge $\{E_i\}_{i\in[1,\ell]}$ from V.
Step 3. Run machine A_3 with input $\{\rho_i\}_{i\in[1,\ell]} = \{E_i\}_{i\in[1,\ell]}$ and get the forged signature $(X, \{m_i\}_{i\in[1,\ell]}, Y)$.
Step 4. Output Y to V.

Note that A_3 outputs a valid signature $(X, \{m_i\}_{i\in[1,\ell]}, Y)$ which satisfies a relation of $(\{K_p^i\}_{i\in[1,\ell]}, X, \{E_i\}_{i\in[1,\ell]}, Y)$ and $E_i \leftarrow F_i(X, m_i)$. Therefore, when V checks the validity of this relation, V accepts A_4's proof with (t', ε''). □

Remark 1. In order to be able to use the τ-heavy row lemma in following section (see Theorem 3), we need $\varepsilon'' \geq \tau(\ell+1)/|\mathcal{E}|^\ell$. This leads to

$$\varepsilon' - \frac{q_{sig} \cdot \max_i \ell_i}{|\mathcal{E}|} \geq \frac{\tau(\ell+1)}{|\mathcal{E}|^\ell} \Leftrightarrow \varepsilon' \geq \frac{\tau(\ell+1)}{|\mathcal{E}|^\ell} + \frac{q_{sig} \cdot \max_i \ell_i}{|\mathcal{E}|}$$

and

$$\begin{aligned}
&\frac{1}{(\max_i q_{F_i})^\ell}\left(\varepsilon - \frac{1 - |\mathcal{E}|^\ell}{1 - |\mathcal{E}|} \cdot \frac{1}{|\mathcal{E}|^\ell}\right) \geq \frac{\tau(\ell+1)}{|\mathcal{E}|^\ell} + \frac{q_{sig} \cdot \max_i \ell_i}{|\mathcal{E}|} \Leftrightarrow \\
&\quad \varepsilon \geq (\max_i q_{F_i}) \cdot \left(\frac{\tau(\ell+1)}{|\mathcal{E}|^\ell} + \frac{q_{sig} \cdot \max_i \ell_i}{|\mathcal{E}|}\right) + \frac{1 - |\mathcal{E}|^\ell}{1 - |\mathcal{E}|} \cdot \frac{1}{|\mathcal{E}|^\ell}.
\end{aligned}$$

This is exactly the condition imposed in Theorem 2.

4 Multi-Challenge Identification Schemes

4.1 Description

Inspired by the Schnorr [28] and the Chaum *et al.* [8] protocols, we introduce a multi-challenge identification protocol. Therefore, let $p = 2q + 1$ be a prime number such that q is also prime. Select an element $g \in \mathbb{G}$ of order q in some multiplicative group of order $p - 1$. Also, let $\mathcal{E}$ be a challenge space. The detailed multi-challenge protocol is presented in Figure 1. Note that $\{x_i\}_{i\in[1,n]}$ and $\{z_i\}_{i\in[1,n]}$ play the role of the secret key and the public key, respectively. Remark that when ℓ is known beforehand, it can be omitted in the first message.

Note that when $n = 1$ the Schnorr scheme [28] is a special case of our protocol. Also, when $\ell = n$ and $\mathcal{E} = \{0, 1\}$ we obtain the Chaum *et al.* protocol [8].

$Peggy$		$Victor$
Knows $\{x_i\}_{i\in[1,n]} \xleftarrow{\$} \mathbb{Z}_q^n$. Computes $\{z_i\}_{i\in[1,n]} = \{g^{x_i}\}_{i\in[1,n]}$.		Knows $\{z_i\}_{i\in[1,n]}$.
Select $\ell \in [1, n]$. Choose $k \xleftarrow{\$} \mathbb{Z}_q$. Compute $t \leftarrow g^k$.		
	$\xrightarrow{\ell,t}$	
		Choose $c = \{c_i\}_{i\in[1,\ell]} \xleftarrow{\$} \mathcal{E}^\ell \subseteq \mathbb{Z}_q^\ell$.
	$\xleftarrow{c}$	
Compute $r \leftarrow k + \sum_{i=1}^{\ell} x_i c_i \bmod q$.		
	$\xrightarrow{r}$	
		If $g^r = t \cdot \left(\prod_{i=1}^{\ell} z_i^{c_i}\right)$ return `true`. Else return `false`.

Fig. 1. A multi-challenge protocol (SBP[7]).

Correctness. To prove that $Peggy$'s proof always convinces $Victor$, we evaluate the verification condition

$$g^r = g^{k+x_1c_1+\ldots+x_\ell c_\ell} = g^k g^{x_1c_1} \ldots g^{x_\ell c_\ell} = tz_1^{c_1} \ldots z_\ell^{c_\ell}.$$

Let $s = |\mathcal{E}|$. Note that a corrupt prover $\bar{P}$ can cheat $Victor$ with a probability of $s^{-\ell}$ per iteration by guessing the $\{c_i\}_{i\in[1,\ell]}$ vector, preparing $t = g^k z_1^{-c_1} \ldots z_\ell^{-c_\ell}$ in the first step, and providing $r = k$ in the last step. Therefore, we must choose s and the number of iteration v such that $s^{-v\ell}$ is negligible.

4.2 Security Analysis

We further assume, without loss of generality, that when an adversary A succeeds in cheating the verifier V, V sends a challenge vector with ℓ entries. Also, to simplify our analysis we assume that $\mathcal{E} = \mathbb{Z}_q$.

In the case of our protocol, the key searching problem that we use is the following: Given $(p, q, g, \mathbb{G}, n, \{z_i\}_{i\in[1,n]})$ compute $x_i \in \mathbb{Z}_q$ such that $g^{x_i} = z_i$ for $i \in [1, \ell]$. We further refer to it as the ℓ out of n discrete logarithms problem. Note that when $n = 1$ we obtain the classical discrete logarithm problem.

Theorem 3. *Let $\varepsilon \geq \tau(\ell+1)/q^\ell$ and $(5\tau/3(\tau-1))^\ell$ be polynomially bounded. Suppose that the ℓ out of n discrete logarithms problem is $(\bar{t}, \bar{\varepsilon})$-secure. Then the SBP protocol is (t, ε)-secure, where*

$$\bar{t} = \frac{(1+\tau\ell)(t+\Phi_1)}{\varepsilon} + \Phi_3 \text{ and } \bar{\varepsilon} = \left(\frac{\tau-1}{\tau}\right)^\ell \left(1-\frac{1}{e}\right)^{\ell+1} > \frac{3}{5}\left(\frac{3(\tau-1)}{5\tau}\right)^\ell.$$

By Φ_1 we denoted the verification time of the identification protocol and Φ_3 is time needed to compute the $\{x_i\}_{i\in[1,\ell]}$ vector in the final stage.

Proof. Before starting our proof we remark that the Boolean matrix M has q^ℓ columns. Therefore, the condition $\varepsilon/\tau \geq (\ell+1)/q^\ell$ assures us that a τ-heavy row contains at least $\ell+1$ ones.

Let A be a PPT adversary who can break the identification scheme with (t, ε). We further construct an adversary B which computes ℓ discrete logarithms out of n with $(\bar{t}, \bar{\varepsilon})$ using A.

B will first repeatedly probe the Boolean matrix M at random, until it finds an entry $a(1)$ with an 1. This happens after an expected number of $1/\varepsilon$ repetitions. In this case, B's success probability is $1-(1-1/\varepsilon)^{1/\varepsilon} > 1-1/e$.

According to the τ-heavy row lemma with probability $(\tau-1)/\tau$, the first 1 that B found lies in a τ-heavy row. Therefore, if B continues probing at random along this row, with probability $(\frac{\varepsilon}{\tau}q-1)/q$ B will find another 1 in one attempt. Therefore, after $q/(\frac{\varepsilon}{\tau}q-1) \simeq \tau/\varepsilon$ repetitions B will find a second entry $a(2)$ with an 1. In this case, B's success probability is

$$\frac{\tau-1}{\tau}\left(1-\left(1-\frac{\tau}{\varepsilon}\right)^{\frac{\tau}{\varepsilon}}\right) > \frac{\tau-1}{\tau}\left(1-\frac{1}{e}\right).$$

Using the same procedure as above, B continues probing until it obtains other $\ell-1$ entries $\{a(i)\}_{i\in[3,\ell+1]}$ with an 1. Note that for entry $i \in [3, \ell+1]$ the expected number of tries to find an 1 is $q/(\frac{\varepsilon}{\tau}q-i+1) \simeq \tau/\varepsilon$.

Therefore, B makes a total of

$$\frac{1}{\varepsilon} + \sum_{i=2}^{\ell+1} \frac{q}{\frac{\varepsilon}{\tau}q-i+1} \simeq \frac{1+\tau\ell}{\varepsilon}$$

[7]Schnorr Based Protocol

expected repetitions and has a success probability of

$$\left(1-\frac{1}{e}\right)\cdot\prod_{i=2}^{\ell+1}\frac{\tau-1}{\tau}\left(1-\frac{1}{e}\right)=\left(\frac{\tau-1}{\tau}\right)^{\ell}\left(1-\frac{1}{e}\right)^{\ell+1}>\left(\frac{\tau-1}{\tau}\right)^{\ell}\left(\frac{3}{5}\right)^{\ell+1}.$$

Now we are in possession of $\ell+1$ entries from the same row. Therefore, we have $\ell+1$ protocol transcripts (t_j, c_j, r_j), where $j \in [1, \ell+1]$, such that $t_1 = \ldots = t_{\ell+1}$. This translates in the following system

$$\begin{cases} r_1 & = k + x_1 c_{1,1} + \ldots + x_\ell c_{1,\ell} \\ r_2 & = k + x_1 c_{2,1} + \ldots + x_\ell c_{2,\ell} \\ & \ldots \\ r_{\ell+1} & = k + x_1 c_{\ell+1,1} + \ldots + x_\ell c_{\ell+1,\ell} \end{cases}$$

that has $\ell+1$ unknowns. According to [6], the probability that the system's determinant is not zero is $(1-1/q)(1-1/q^2)\ldots(1-1/q^{\ell+1}) \simeq 1$. Therefore, we can determine the $\{x_i\}_{i\in[1,\ell]}$ vector with non-negligible probability. □

Remark 2. When $\ell = 1$ and $\tau = 2$ we obtain the result from [23], *i.e.* $\bar{\varepsilon} > 0.18$. For $\ell = 8$ we can select $\tau = 40$ to get $\bar{\varepsilon} > 0.008$.

Theorem 4. *The SBP protocol is a perfect zero knowledge protocol if* $|\mathcal{E}^\ell|$ *is polynomially bounded.*

Proof. For every g and $\{z_i\}_{i\in[1,n]}$ the output of the simulator has to be indistinguishable from the distribution of a real transcript. Such a simulator is presented in Algorithm 1. Therefore, we obtain that SBP is c-simulatable. Using Theorem 1 we obtain the desired result.

Algorithm 1: The simulator S.

Input: The public key $\{z_i\}_{i\in[1,n]}$
Output: A transcript $\mathcal{L}$
foreach $j \in [1, v]$ **do**
  Select an integer ℓ
  Select $c = \{c_i\}_{i\in[1,\ell]}$ at random from C^ℓ
  Select a random number $r \xleftarrow{\$} \mathbb{Z}_q$
  Compute $t \leftarrow g^r z_1^{-c_1} \ldots z_\ell^{-c_\ell}$
  $L \leftarrow L \cup \{(t, c, r)\}$
end
return $\mathcal{L}$

□

Peggy		*Victor*
Knows X.		Knows Z.
Select $\ell \in [1, n]$. Choose $\{k_j\}_{j\in[1,\alpha]} \xleftarrow{\$} \mathbb{Z}_q^\alpha$. Compute $t \leftarrow g_1^{k_1} \dots g_\alpha^{k_\alpha}$.		
	$\xrightarrow{\ell, t}$	
		Choose $c = \{c_i\}_{i\in[1,\ell]} \xleftarrow{\$} \mathcal{E}^\ell \subseteq \mathbb{Z}_q^\ell$.
	$\xleftarrow{c}$	
Compute $\{r_j\}_{j\in[1,\alpha]}$ such that $r_j \leftarrow k_j + \sum_{i=1}^{\ell} x_{i,j} c_i \bmod q$.		
	$\xrightarrow{\{r_j\}_{j\in[1,\alpha]}}$	
		If $g_1^{r_1} \dots g_\alpha^{r_\alpha} = t \cdot \left(\prod_{i=1}^{\ell} z_i^{c_i}\right)$ return `true`. Else return `false`.

Fig. 2. Another multi-challenge protocol.

4.3 Variations

4.3.1 Girault based protocol. We further discuss a variation of the SBP protocol, further denoted GBP. Let $p = 2fp' + 1$ and $q = 2fq' + 1$ be prime numbers such that f, p' and q' are distinct primes. Select an element $g \in \mathbb{Z}_N^*$ of order f, where $N = pq$. Note that p and q are secret.

We further present the differences between the GBP version and the SBP protocol. The secret key is selected from $\mathbb{Z}_f^n$ instead of $\mathbb{Z}_q^n$. In the first step of the protocol *Peggy* randomly selects k from $\mathbb{Z}_f$ instead of $\mathbb{Z}_q$. In the second step the challenge space is chosen such that $\mathcal{E}^\ell \subseteq \mathbb{Z}_f^\ell$ and in the third step we compute r modulo f instead of modulo q.

Note that when $n = 1$ we obtain the Girault protocol [14] and when $\ell = n$ we obtain a protocol introduced in [20].

4.3.2 Multiple discrete logarithm representation based protocol. Let $\alpha > 0$ be an integer. In this variant of the protocol we select α elements $\{g_j\}_{j\in[1,\alpha]}$ of order q from $\mathbb{G}$. *Peggy*'s secret key is $X = (\{x_{1,j}\}_{j\in[1,\alpha]}, \dots, \{x_{n,j}\}_{j\in[1,\alpha]})$ and the public key is computed such that $z_i = g_1^{x_{i,1}} \dots g_\alpha^{x_{i,\alpha}}$ for $i \in [1, n]$. Let $Z = \{z_i\}_{i\in[1,n]}$. We present the protocol based on representations in Figure 2.

Note that when $n = 1$ we obtain a protocol proposed by Maurer in [21] which is a generalization of the protocols presented by Okamoto in [24] and Chaum *et.al.* in [8]. Also, when $\ell = n$ we obtain a protocol introduced in [20].

Chaum *et al.* [8] also provide a protocol variant for a composite n. Thus, by adapting the GBP protocol and tweaking the previously described one, we can obtain a similar version for composite numbers.

Correctness. To prove that $Peggy$'s proof always convinces $Victor$, we evaluate the verification condition

$$\begin{aligned} g_1^{r_1} \dots g_\alpha^{r_\alpha} &= \prod_{j=1}^{\alpha} g_j^{k_j + x_{1,j}c_1 + \ldots + x_{\ell,j}c_\ell} \\ &= \prod_{j=1}^{\alpha} g_j^{k_j} \left(\prod_{j=1}^{\alpha} g_j^{x_{1,j}} \right)^{c_1} \dots \left(\prod_{j=1}^{\alpha} g_j^{x_{\ell,j}} \right)^{c_\ell} \\ &= t z_1^{c_1} \dots z_\ell^{c_\ell}. \end{aligned}$$

Similarly to the case of SBP, a corrupt prover $\bar{P}$ can cheat $Victor$ with a probability of $s^{-\ell}$ per iteration by guessing the challenge. Therefore, we must choose s and v such that $s^{-v\ell}$ is negligible.

5 Multi-Message Signature Schemes

5.1 Description

In this section we transform the SBP protocol into a multi-message signature scheme. Note that when $n = 1$ we obtain the classical Schnorr signature.

Key generation: Let $n > 0$ be an integer. Generate two large prime numbers p, q, such that $q \geq 2^\lambda$ and $q | p - 1$. Select a cyclic group $\mathbb{G}$ of order $p - 1$ and let $g \in \mathbb{G}$ be an element of order q. Let $h : \{0,1\}^* \to \mathbb{Z}_q^*$ be a hash function. Choose $x_i \xleftarrow{\$} \mathbb{Z}_q^*$ and compute $y_i \leftarrow g^{x_i}$ for $i \in [1, n]$. Output the public key $pk = (p, q, g, \mathbb{G}, n, h, \{y_i\}_{i \in [1,n]})$. The secret key is $sk = \{x_i\}_{i \in [1,n]}$.

Signature generation: Let $\ell \in [1, n]$ and $i \in [1, \ell]$. To sign ℓ messages $m_i \in \{0,1\}^*$, first generate a random number $k \xleftarrow{\$} \mathbb{Z}_q$. Then compute the values $t \leftarrow g^k$, $e_i \leftarrow h(i \| t \| m_i)$ and $r \leftarrow k + x_1 e_1 + \ldots + x_\ell e_\ell \bmod q$. Output the signature (t, r).

Signature verification: To verify the signature (t, r) of messages $\{m_i\}_{i \in [1,\ell]}$, compute $e_i \leftarrow h(i \| t \| m_i)$ and check if

$$g^r = t \cdot \left(\prod_{i=1}^{\ell} z_i^{e_i} \right) \tag{1}$$

holds. Output `true` if and only if Equation (1) holds. Otherwise, output `false`.

5.2 Security Analysis

In order to prove the security of our proposal we model $F_i(\cdot) = h(i\|\cdot)$ as random oracles. Therefore, we are able to use Theorems 2 and 3 to obtain the following result. We further denote by SBS[8] our proposed signature.

Theorem 5. *Let* $\varepsilon' \geq (\tau(\ell+1) + q^{\ell-1}q_{sig} \cdot \max_i \ell_i)/q^\ell$*, where*

$$\varepsilon' = \frac{1}{(\max_i q_{F_i})^\ell}\left(\varepsilon - \frac{1-q^\ell}{1-q} \cdot \frac{1}{q^\ell}\right).$$

Also, let $(5\tau/3(\tau-1))^\ell$ *be polynomially bounded. Suppose that the* ℓ *out of* n *discrete logarithms problem is* $(\bar{t}, \bar{\varepsilon})$*-secure. Then the SBS signature scheme is* $(t, q_{sig}, \{q_{F_i}\}_{i\in[1,n]}, \varepsilon)$*-secure, where*

$$\bar{t} = \frac{(1+\tau\ell)t'}{\varepsilon} + \Phi_3 \text{ and } \bar{\varepsilon} = \left(\frac{\tau-1}{\tau}\right)^\ell \left(1 - \frac{1}{e}\right)^{\ell+1} > \frac{3}{5}\left(\frac{3(\tau-1)}{5\tau}\right)^\ell,$$

and

$$t' = t + \Phi_1 + \Phi_2 \text{ and } \varepsilon'' = \varepsilon' - \frac{q_{sig} \cdot \max_i \ell_i}{q}.$$

By Φ_1 *we denoted the verification time of the identification protocol,* Φ_2 *is the simulation time of* q_{sig} *signatures and* Φ_3 *is time needed to compute the* $\{x_i\}_{i\in[1,\ell]}$ *vector in the final stage.*

Proof. The only thing we need to apply Theorem 2 is to provide a simulator for the SBP protocol. The simulator S described in Algorithm 1 can mimic the communication in SBP with an indistinguishable probability distribution. Note that in Algorithm 1 v denotes the number of protocol iterations.

□

5.3 Implementation

We implemented in C using the GMP library [4] our multi-message signature with SHA256 (SBS-256) and with SHA512 (SBS-512). For comparison we also implemented the Schnorr signature with SHA256 (Sch-256) and with SHA512 (Sch-512). The hash function used internally by the algorithms is either SHA256 or SHA512 [1].

The programs were run on a CPU Intel i7-4790 4.00 GHz and compiled with GCC with the O3 flag activated. Note that our processor has at most 8 threads. In our experiments for each prime size of 2048, 3072 and 4096 bits, we ran the algorithms with 100 safe prime numbers from [3]. Let *th* be the number of used threads. For each prime we measured the average running time for 100 random *th* megabytes messages using the function *omp_get_wtime*() [2]. Note that for the Schnorr signature we sign the entire message of size *th* megabytes in one go.

The results of our experiments are presented in Figures 3 to 8. We can see from the plots that our signature has a better execution times than the Schnorr signature no matter if we use SHA256 or SHA512.

[8] Schnorr Based Signature

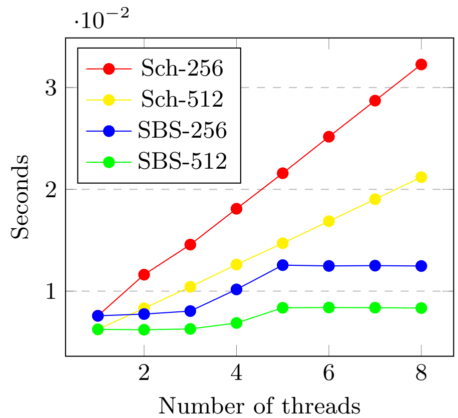

Fig. 3. Signing time for $\lambda = 2048$

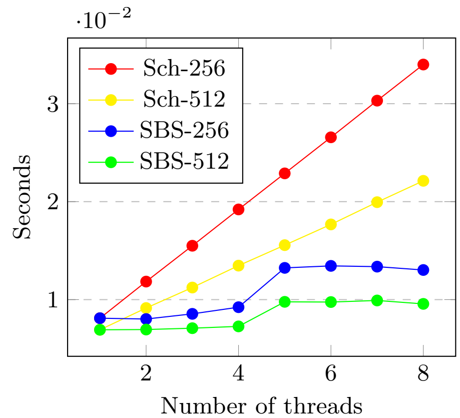

Fig. 4. Verification time for $\lambda = 2048$

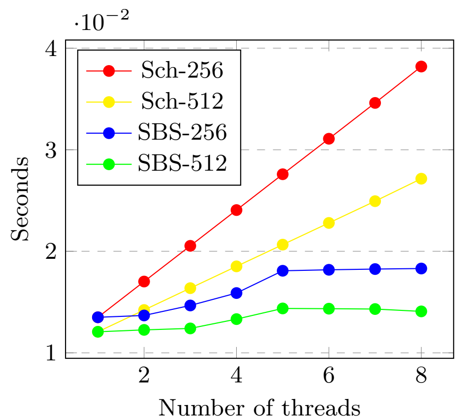

Fig. 5. Signing time for $\lambda = 3072$

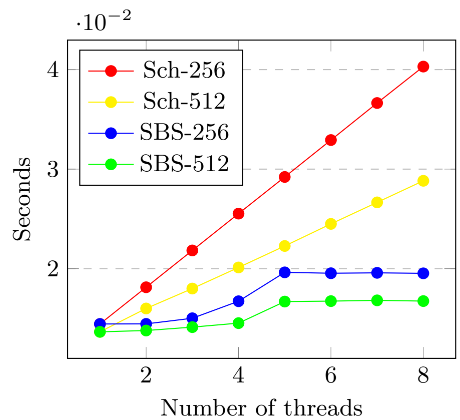

Fig. 6. Verification time for $\lambda = 3072$

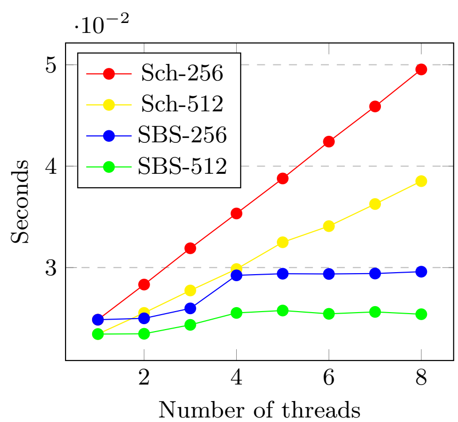

Fig. 7. Signing time for $\lambda = 4096$

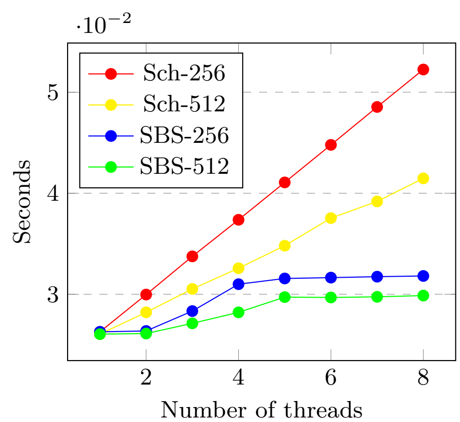

Fig. 8. Verification time for $\lambda = 4096$

6 Multi-Message Co-Signature Scheme

In [12] the authors present a contract signing paradigm to achieve legal fairness. Their provably secure co-signature construction is based on the Schnorr digital signature [28]. Using our multi-message signature, we extend Ferradi *et al.*'s co-signature to support multiple messages. Our novel co-signature is presented in Figure 9. Note that, as in the original scheme [12], the property of ambiguity[9] does not apply, since our scheme produces only a single output. Also, the notion of fairness[10] is inherent, as a co-signature becomes binding for both parties simultaneously.

In Figure 9, $\mathcal{L}$ represents a local non-volatile memory used by *Bob*, and $\{(x_{A,i}, z_{A,i})\}_{i\in[1,n]}$ and $\{(x_{B,i}, z_{B,i})\}_{i\in[1,n]}$ are the keys used by *Alice* and *Bob*, respectively. Also, note that $\{z_i\}_{i\in[1,n]}$ is the joint public key of *Alice* and *Bob*. During the protocol, *Alice* makes use of a publicly known auxiliary signature scheme σ using her secret key $x_{A,1}$, and *Bob* uses σ's verification algorithm to check if ω is correct.

Correctness. The correctness of the co-signing scheme described in Figure 9 follows from

$$\begin{aligned}
g^r &= g^{r_A + r_B} \\
&= g^{r_A} \cdot g^{r_B} \\
&= t_A \cdot \left(\prod_{i=1}^{\ell} (z_{A,i})^{e_i}\right) \cdot t_B \cdot \left(\prod_{i=1}^{\ell} (z_{B,i})^{e_i}\right) \\
&= t_A \cdot t_B \cdot \left(\prod_{i=1}^{\ell} (z_{A,i} \cdot z_{B,i})^{e_i}\right) \\
&= t \cdot \left(\prod_{i=1}^{\ell} z_i^{e_i}\right).
\end{aligned}$$

6.1 Security Analysis

We prove that our proposed co-signature is secure in the random oracle model using the following strategy: assuming that adversary A is an efficient forger for the co-signature protocol, we turn A into an efficient forger for the SBS signature. There are two possible scenarios that we address, either A plays the role of *Alice* or the role of *Bob*. We further denote by Co-SBS our proposed multi-message co-signature.

---

[9] It is impossible to determine which of the two parties produced the signature.

[10] *Bob* cannot be placed in a situation where their signature is bound while *Alice*'s initial signature remains unbound.

Alice		*Bob*
$z_i \leftarrow z_{A,i} \cdot z_{B,i}\ \forall i \in [1,n]$		$z_i \leftarrow z_{A,i} \cdot z_{B,i}\ \forall i \in [1,n]$
$k_A \xleftarrow{\$} \mathbb{Z}_q$		$k_B \xleftarrow{\$} \mathbb{Z}_q$
$t_A \leftarrow g^{k_A}$		$t_B \leftarrow g^{k_B}$
		$\rho \leftarrow h(0 \Vert t_B)$
	$\xleftarrow{\rho}$	
$w \leftarrow \sigma(t_A \Vert \text{Alice} \Vert \text{Bob})$	$\xrightarrow{t_A, w}$	
		if w is incorrect **then** abort
		store w in $\mathcal{L}$
	$\xleftarrow{t_B}$	
if $h(0 \Vert t_B) \neq \rho$ **then** abort		
$t \leftarrow t_A \cdot t_B$		$t \leftarrow t_A \cdot t_B$
$e_i \leftarrow h(i \Vert m_i \Vert t \Vert \text{Alice} \Vert \text{Bob})\ \forall i \in [1,n]$		$e_i \leftarrow h(i \Vert m_i \Vert t \Vert \text{Alice} \Vert \text{Bob})\ \forall i \in [1,n]$
$r_A \leftarrow k_A + \sum_{i=1}^{\ell} x_{A,i} e_i \bmod q$		$r_B \leftarrow k_B + \sum_{i=1}^{\ell} x_{B,i} e_i \bmod q$
		store r_B in $\mathcal{L}$
	$\xleftarrow{r_B}$	
if r_B is incorrect **then** abort		
	$\xrightarrow{r_A}$	
		if r_A is incorrect **then** abort
$r \leftarrow r_A + r_B \bmod q$		$r \leftarrow r_A + r_B \bmod q$
		if $\{m, t, r\}$ is valid **then**
		erase w, r_B from $\mathcal{L}$

Fig. 9. A legally fair multi-message signature.

6.1.1 Adversary Attacks Bob

Theorem 6. *If A_{Alice} plays the role of Alice and is able to forge a Co-SBS co-signature with non-negligible probability, then we can construct an adversary that breaks SBS with a non-negligible probability of success.*

Proof. We further show how to construct a simulator S_{Bob} that interacts with A_{Alice} and forces the adversary to produce an SBS forgery. Note that S_{Bob} has to emulate not only *Bob*, but also all oracles and the directory $\mathcal{D}$. Here is how this simulator behaves at each step of the protocol.

Key Establishment Phase. S_{Bob} is given as input the target's public key $\{z_i\}_{i\in[1,n]}$. To inject it into A_{Alice}, the simulator S_{Bob} reads $\{z_{A,i}\}_{i\in[1,n]}$ from $\mathcal{D}$ and impersonates *Bob* whose public key is $\{z_{B,i}\}_{i\in[1,n]}$, where $z_{B,i} \leftarrow z_i z_{A,i}^{-1}$ for all i. Therefore, the common key between A_{Alice} and S_{Bob} is $\{z_{A\Vert B,i}\}_{i\in[1,n]}$, where $z_{A\Vert B,i} \leftarrow z_{A,i} z_{B,i}$ for all i, which is by construction $\{z_i\}_{i\in[1,n]}$.

Now S_{Bob} starts the protocol with A_{Alice}, who queries the directory and gets $\{z_{B,i}\}_{i\in[1,n]}$. From the point of view of A_{Alice}, she has successfully established a co-signature protocol with the “co-signer” S_{Bob}.

Query Phase. A_{Alice} will start to present queries to S_{Bob}. Therefore, S_{Bob} must respond to three types of queries: random oracles queries, co-signature queries and transcript queries. We present in Algorithm 2 the simulation of the random oracle F_i. Note that at the beginning of the simulation $T_i \leftarrow \emptyset$. The simulation for the co-signature protocol is described in Algorithm 3. Note that when A_{Alice} requests a conversation transcript, S_{Bob} replies by sending the transcript from a previously successful interaction.

Algorithm 2: Random oracle $\mathcal{O}_{F_i}$ simulation for F_i.

Input: A random oracle query q_j from A_{Alice}

if $\exists h_j, \{q_j, h_j\} \in T_i$ **then**
  $e \leftarrow h_j$
else
  $e \xleftarrow{\$} \mathbb{Z}_q$
  Append $\{q_j, e\}$ to T_i
end
return e

Output Phase. After performing its queries, A_{Alice} eventually outputs a valid co-signature (t, r) for $\{z_{A\|B,i}\}_{i\in[1,n]}$ where $t = t_A t_B$ and $r = r_A + r_B$. By design, these parameters are those of an SBS signature, and thus A_{Alice} has produced an SBS forgery.

Algorithm 3: Co-signing oracle simulation for S_{Bob}.

Input: A co-signature query $\{m_i\}_{i\in[1,\ell]}$ from A_{Alice}

$r_B \xleftarrow{\$} \mathbb{Z}_q$
$\{e_i\}_{i\in[1,\ell]} \xleftarrow{\$} \mathbb{Z}_q^\ell$
$t_B \leftarrow g^{r_B} \cdot \prod_{i=1}^{\ell} z_i^{-e_i}$
Send $h(0\|t_B)$ to A_{Alice}
Receive t_A, w from A_{Alice}
Send t_B to A_{Alice}
$t \leftarrow t_A t_B$
$u_i \leftarrow i\|m\|t\|\mathrm{Alice}\|\mathrm{Bob}$
if $\forall i\ \exists e'_i \neq e_i, \{u_i, e'_i\} \in T_i$ **then**
  abort
else
  Append $\{u_i, e_i\}$ to T_i
end
return s_B

There is a case in which S_{Bob} aborts the protocol before completion. This happens when it turns out that $i\|m\|t\|\text{Alice}\|\text{Bob}$ has been previously queried by A_{Alice}. In this case, S_{Bob} cannot reprogram oracle $\mathcal{O}_{F_i}$, and thus it has to abort. Since A_{Alice} does not know the random value t_B, such an event would happen with a negligible probability q_{F_i}/q, where q_{F_i} is the number of queries to $\mathcal{O}_{F_i}$.

Therefore, A_{Alice} breaks the SBS signature with probability $1-(\ell\cdot\max_i q_{F_i})/q$. If A_{Alice} has a success probability ε, the success probability of A_{Alice} in the simulated environment is $\varepsilon' = (1-(\ell\cdot\max_i q_{F_i})/q)\varepsilon$. □

6.1.2 Adversary Attacks Alice

Theorem 7. *If A_{Bob} plays the role of Bob and is able to forge a Co-SBS co-signature with non-negligible probability, then we can construct an adversary that breaks SBS with a non-negligible probability of success if signature σ can be simulated without knowing the secret key $x_{A,1}$.*

Proof. This proof is similar to Theorem 6, and thus we omit some details. In this case, we construct a simulator S_{Alice} that interacts with A_{Bob} and forces it to produce an SDS forgery. The simulator's behavior at different stages of the security game is as follows.

Key Establishment Phase. S_{Alice} is given the target's public key $\{z_i\}_{i\in[1,n]}$. S_{Alice} injects $\{z_{A,i}\}_{i\in[1,n]}$ into A_{Bob} as described in Theorem 6. Now S_{Alice} activates A_{Bob}, who queries $\mathcal{D}$ and gets $\{z_{A,i}\}_{i\in[1,n]}$. From the point of view of A_{Bob}, he has successfully established a co-signature protocol with the "co-signer" S_{Alice}.

Query Phase. A_{Bob} will start to present queries to S_{Alice}. Therefore, S_{Alice} must respond to four types of queries: random oracles queries, signature queries, co-signature queries and transcript queries. We consider oracles $\mathcal{O}_{F_i}$ as in Theorem 6. We denote by $\mathcal{O}_\sigma$ the simulation of σ. The simulation for the co-signature protocol is described in Algorithm 4. Note that when A_{Bob} requests a conversation transcript, S_{Alice} replies by sending the transcript from a previously successful interaction.

Output Phase. After performing its queries, A_{Bob} eventually outputs a valid co-signature (t,r) for $\{z_{A\|B,i}\}_{i\in[1,n]}$ where $t=t_At_B$ and $r=r_A+r_B$. By design, these parameters are those of an SBS signature, and thus A_{Bob} has produced an SBS forgery.

As in Theorem 6, Algorithm 4 may fail with probability q_{F_i}/q. Thus, the success probability of A_{Bob} in the simulated environment is $\varepsilon' = (1-(\ell\cdot\max_i q_{F_i})/q)\varepsilon$. □

Algorithm 4: Co-signing oracle simulation for S_{Alice}.

Input: A co-signature query $\{m_i\}_{i\in[1,\ell]}$ from A_{Bob}

Receive ρ from A_{Bob}
Query T_0 to retrieve t_B such that $F_0(t_B) = \rho$
$r_A \xleftarrow{\$} \mathbb{Z}_q$
$\{e_i\}_{i\in[1,\ell]} \xleftarrow{\$} \mathbb{Z}_q^\ell$
$t \leftarrow t_B \cdot g^{r_A} \cdot \prod_{i=1}^{\ell} z_i^{-e_i}$
$u_i \leftarrow i\|m\|t\|\text{Alice}\|\text{Bob}$
if $\forall i\ \exists e_i' \neq e_i, \{u_i, e_i'\} \in T_i$ **then**
  abort
else
  Append $\{u_i, e_i\}$ to T_i
end
$t_A \leftarrow t t_B^{-1}$
$u_0 \leftarrow t_A\|\text{Alice}\|\text{Bob}$
$w \leftarrow \mathcal{O}_\sigma(u_0)$
Send t_A, w to A_{Bob}
Receive t_B from A_{Bob}
Receive r_B from A_{Bob}
return r_A

7 Conclusions

In this paper we presented a novel signature scheme and we introduced the theoretical framework needed to prove its security. We also conducted a series of experiments to show that our proposal is more efficient that the Schnorr signature in the multiple-message scenario. Based on our signature scheme we also introduced a co-signature scheme that inherits the properties of its underlying primitive.

Future work. Starting from the framework introduced in [21], the authors of [20] further generalize it to include more zero-knowledge protocols. Some particular cases of this generic framework are the Feige-Fiat-Shamir protocol [11], the Guillou-Quisquater [18] protocol and the Schnorr protocol [28]. An interesting research direction is to see how we can apply signatures based on the framework from [20] in the multiple-message scenario. As seen in this paper, we only managed to adapt it to discrete logarithm based signatures.

References

1. mbed TLS. https://tls.mbed.org
2. OpenMP. https://www.openmp.org/
3. Safe Prime Database. https://2ton.com.au/safeprimes/
4. The GNU Multiple Precision Arithmetic Library. https://gmplib.org/

5. Asokan, N., Schunter, M., Waidner, M.: Optimistic Protocols for Fair Exchange. In: CCS 1997. pp. 7–17. ACM (1997)
6. Brent, R.P., McKay, B.D.: Determinants and Ranks of Random Matrices over Z_m. Discrete Mathematics **66**(1-2), 35–49 (1987)
7. Cachin, C., Camenisch, J.: Optimistic Fair Secure Computation. In: CRYPTO 2000. Lecture Notes in Computer Science, vol. 1880, pp. 93–111. Springer (2000)
8. Chaum, D., Evertse, J.H., Van De Graaf, J.: An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations. In: EUROCRYPT 1987. Lecture Notes in Computer Science, vol. 304, pp. 127–141. Springer (1987)
9. Chen, L., Kudla, C., Paterson, K.G.: Concurrent Signatures. In: EUROCRYPT 2004. Lecture Notes in Computer Science, vol. 3027, pp. 287–305. Springer (2004)
10. Damgård, I.: On Σ-protocols. https://www.cs.au.dk/~ivan/Sigma.pdf (2010)
11. Feige, U., Fiat, A., Shamir, A.: Zero-Knowledge Proofs of Identity. Journal of cryptology **1**(2), 77–94 (1988)
12. Ferradi, H., Géraud, R., Maimuţ, D., Naccache, D., Pointcheval, D.: Legally Fair Contract Signing Without Keystones. In: International Conference on Applied Cryptography and Network Security - ACNS'16. Lecture Notes in Computer Science, vol. 9696, pp. 175–190. Springer (2016)
13. Garay, J., MacKenzie, P., Prabhakaran, M., Yang, K.: Resource Fairness and Composability of Cryptographic Protocols. In: TCC 2006. Lecture Notes in Computer Science, vol. 3876, pp. 404–428. Springer (2006)
14. Girault, M.: An Identity-based Identification Scheme Based on Discrete Logarithms Modulo a Composite Number. In: EUROCRYPT 1990. Lecture Notes in Computer Science, vol. 473, pp. 481–486. Springer (1990)
15. Goldwasser, S., Levin, L., Vanstone, S.A.: Fair Computation of General Functions in Presence of Immoral Majority. In: CRYPT0 1990. Lecture Notes in Computer Science, vol. 537, pp. 77–93. Springer (1991)
16. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof Systems. SIAM J. Comput. **18**(1), 186–208 (1989)
17. Gordon, S.D., Hazay, C., Katz, J., Lindell, Y.: Complete Fairness in Secure Two-Party Computation. Jornal of the ACM **58**(6), 1–37 (December 2011)
18. Guillou, L.C., Quisquater, J.J.: A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory. In: EUROCRYP 1988. Lecture Notes in Computer Science, vol. 330, pp. 123–128. Springer (1988)
19. Maimuţ, D., Teşeleanu, G.: A Unified Security Perspective on Legally Fair Contract Signing Protocols. In: SECITC 2018. Lecture Notes in Computer Science, vol. 11359, pp. 477–491. Springer (2018)
20. Maimuţ, D., Teşeleanu, G.: A Generic View on the Unified Zero-Knowledge Protocol and Its Applications. In: WISTP 2019. Lecture Notes in Computer Science, vol. 12024, pp. 32–46. Springer (2019)
21. Maurer, U.: Zero-Knowledge Proofs of Knowledge for Group Homomorphisms. Designs, Codes and Cryptography **77**(2-3), 663–676 (2015)
22. Micali, S.: Simple and Fast Optimistic Protocols for Fair Electronic Exchange. In: PODC 2003. pp. 12–19. ACM (2003)
23. Ohta, K., Okamoto, T.: On Concrete Security Treatment of Signatures Derived from Identification. In: CRYPTO 1998. Lecture Notes in Computer Science, vol. 1462, pp. 354–369. Springer (1998)

24. Okamoto, T.: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In: CRYPTO 1992. Lecture Notes in Computer Science, vol. 740, pp. 31–53. Springer (1992)
25. Pinkas, B.: Fair Secure Two-Party Computation. In: EUROCRYPT 2003. Lecture Notes in Computer Science, vol. 2656, pp. 87–105. Springer (2003)
26. Pointcheval, D., Stern, J.: Security Proofs for Signature Schemes. In: EUROCRYPT 1996. Lecture Notes in Computer Science, vol. 1070, pp. 387–398. Springer (1996)
27. Pointcheval, D., Stern, J.: Security Arguments for Digital Signatures and Blind Signatures. Journal of Cryptology **13**(3), 361–396 (2000)
28. Schnorr, C.P.: Efficient Identification and Signatures For Smart Cards. In: CRYPTO 1989. Lecture Notes in Computer Science, vol. 435, pp. 239–252. Springer (1989)
29. Teşeleanu, G.: Concurrent Signatures from a Variety of Keys. In: Inscrypt 2021. Lecture Notes in Computer Science, vol. 13007, pp. 3–22. Springer (2021)