Publ. Math. Debrecen
Supplementum **100** (2022), 583–595
DOI: 10.5486/PMD.2022.Suppl.3

A provable M-of-N signature scheme based on the BDHI-type assumption in the random oracle model

By Mariusz Jurkiewicz

Abstract. We describe a new group-based M-of-N multisignature scheme (i.e., a protocol which allows a group of signers to produce joint signature on a common message), based on an asymmetric pairing of Type 3. The idea of the scheme is such that there are arbitrary number of signing parties with independent keys that sign the same message. Unlike the regular digital signature schemes, the signing algorithm is split into two separated stages, namely making pre-signatures and generating final aggregate signature. The security analysis is conducted in the euf-cma model, where the security of the scheme is reduced to the computational hardness of solving the bilinear Diffie–Hellman inversion problem. The reduction is made in the random oracle model.

1. Introduction

In this paper we propose and consider a new multisignature scheme. Multisignatures [7] compress signatures made by a group of different signers (each possessing its own private/public key pair) on a common message into a single compact, joint signature. Verification of the validity of a purported signature on a given message is able to be conducted via the set of public keys of all signers. A standard signature scheme can be easily transformed into a multisignature scheme by signing separately a given message and then concatenating all individual signatures. This approach has two fundamental weaknesses, namely both the size of the multisignature and a number of launches of a verification algorithm in these cases grow linearly with the number of signers.

Mathematics Subject Classification: 94A60, 94A62, 68Q25.
Key words and phrases: M-of-N multisig. and random-oracle model and bilinear Diffie–Hellman inversion problem.

So far, the most practical provably secure multisignature scheme that does not impose any key setup or PKI requirements has been proposed by Bellare and Neven [2] and is based on the Schnorr signature scheme [12]. The improvement of this scheme in terms of allowing key aggregation under the Discrete Logarithm assumption in the plain public-key model has been given by Maxwell, Polesta, Seurin and Wuille in [8]. In fact, there is a number of proposals [13], [8], [4] for multisignature schemes that are based on Schnorr signatures. The Schnorr signature scheme uses a cyclic group $\mathbb{G}$ of prime order p, a generator g of $\mathbb{G}$ and a hash function H. A secret/public key pair is a pair $(x, X) \in \mathbb{F}_p \times \mathbb{G}$, where $X = g^x$. To sign a message m, the signer picks $r \xleftarrow{\$} \mathbb{F}_p^*$, computes a nonce $R = g^r$, $c = H(X, R, m)$, and $s = r + cx$. The signature is the pair (R, s), and its validity can be checked by verifying whether $g^s = RX^c$. The naive way to design a multi-signature scheme fully compatible with Schnorr signatures, is a good example of constructions that cause vulnerability to a rogue-key attack (see for example [10], [9]) where a corrupted signer sets public keys of the other signers and sets its public key, that allows him to produce signatures by himself.

As opposed to the Schnorr-based approach, we propose a construction based on Type-3 pairings, which is dedicated to multisignatures with arbitrary number of signing parties. Although, for the reason of this paper, pairings are treated in a completely abstract fashion, they ought to be viewed as being actually defined over $E(\mathbb{F}_{q^n})[p] \times E(\mathbb{F}_{q^{nk}})[p] \to \mathbb{F}_{q^{nk}}[p]$. The main weakness of our proposal is that the length of the signature grows linearly with the number of signers. On the other hand, we were able do obtain full protection against rogue-key attacks, and that signers are only required to have a public key, but do not have to prove knowledge of the private key corresponding to their public key to some certification authority or to other signers before engaging the protocol.

Multi-signatures have many potential applications, but recently they gained popularity for their use in cryptocurrencies [8], [11] in order to save precious block space for multi-input transactions, or as an additional layer of security to protect user wallets. It is well known that traditional bitcoin transactions are conducted between a pair sender/beneficiary and their addresses are derived from the underlying public keys. In January 2012 with Bitcoin Improvement Proposal 16 (BIP-16) it was introduced the new feature, where the beneficiary of a bitcoin transaction is designated as the hash of a script, instead of the owner of a public key. This form of presenting the addresses is called pay-to-script hash (P2SH) addresses and is created from a transaction script, which defines who can spend a transaction output. Currently, the most common implementation of the P2SH function is the multisignature address script. In this case, the underlying script

requires more than one signature to prove ownership and thus spend funds (for more details, see [1]). In this paper we propose a bitcoin multisignature, where there are required M signatures from N available keys. This is the most general variant, known as an M-of-N schemes. It can be used in many common cases, for example, if the funds are joint marital property, and both spouses must accept all the transactions.

The most important and delicate matter for the new cryptographical schemes is to justify that all of the needed security requirements hold. In the modern cryptography this analysis is carried out by doing research within a proper security model, which is euf-cma for the presented scheme. The security proof is conducted in the random oracle model, where the hash function H_1 is modeled as a random oracle.

2. Preliminaries

2.1. Bilinear Diffie–Hellman Inversion Problems. Assume that $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are three multiplicative cyclic groups of prime order p. Let us remind that if $\mathbb{G}_1 \neq \mathbb{G}_2$ and no efficiently computable isomorphism is known between $\mathbb{G}_1$ and $\mathbb{G}_2$, in either direction, then a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is called a pairing of Type 3 if it satisfies the following properties:

bilinearity,: i.e., for all $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ and $\alpha, \beta \in \mathbb{F}_p$ we have

$$\hat{e}\left(g_1^{\alpha}, g_2^{\beta}\right) = \hat{e}\left(g_1, g_2\right)^{\alpha\beta};$$

non-degeneracy,: i.e.,

(i) if for all $g_1 \in \mathbb{G}_1$ we have $\hat{e}(g_1, g_2) = 1_{\mathbb{G}_T}$ then it is equivalent to $g_2 = 1_{\mathbb{G}_2}$;

(ii) if for all $g_2 \in \mathbb{G}_2$ we have $\hat{e}(g_1, g_2) = 1_{\mathbb{G}_T}$ then it is equivalent to $g_1 = 1_{\mathbb{G}_1}$.

Note that for $g_1 \neq 1_{\mathbb{G}_1}$ and $g_2 \neq 1_{\mathbb{G}_2}$ we have $\hat{e}(g_1, g_2) \neq 1_{\mathbb{G}_T}$.

In [3], Boneh et al. defines ℓ-bilinear Diffie–Hellman inversion problem (ℓ-BDHI) for Type-1 pairings. We present below its counterpart for Type-3 pairings, which is denoted by ℓ-BDHI$_3$. To this end, suppose that g_1, g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively and let $\alpha \xleftarrow{\$} \mathbb{F}_p^*$,

$$\ell\text{-BDHI}: \quad \text{given} \quad \{g_i, g_i^{\alpha}, \ldots, g_i^{(\alpha^{\ell})}\},\ i = 1, 2, \ \text{compute} \ \ \hat{e}(g_1, g_2)^{\frac{1}{\alpha}}.$$

In [3] and [6] authors justify that ℓ-BDHI is a computational hard problem. By analogy, the same reasoning can be applied to more general case where Type-3 pairing is taken instead of Type-1; this means that the problem ℓ-BDHI$_3$ is also computationally hard.

Besides ℓ-BDHI$_3$, we also define its extended variant called $(\ell,1)$-BDHI$_3$. Although this problem is essentially equivalent to ℓ-BDHI$_3$ in terms of computational difficulty, it plays an important technical role and is useful in gaining clarity of the security analysis. Suppose that g_1, g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively and let $\alpha, \beta \xleftarrow{\$} \mathbb{F}_p^*$, then $(\ell,1)$-BDHI$_3$ is as follows

$$(\ell,1)\text{-BDHI}_3: \quad \begin{array}{l} \text{given} \quad \{g_i, g_i^{\alpha}, \ldots, g_i^{(\alpha^{\ell})}; g_2^{\beta}, g_2^{\beta\alpha}, \ldots, g_2^{(\beta\alpha^{\ell})}\},\ i=1,2, \\ \text{compute} \quad \hat{e}(g_1,g_2)^{\frac{\beta}{\alpha}}. \end{array}$$

The relation between ℓ-BDHI$_3$ and $(\ell,1)$-BDHI$_3$ is given below.

Lemma 1. *If $\mathcal{A}$ is a $(\ell,1)$-BDHI$_3$-adversary, then there exists a ℓ-BDHI$_3$-adversary $\mathcal{B}$ that runs in essentially the same time as $\mathcal{A}$, furthermore*

$$\mathbf{Adv}_n^{(\ell,1)\text{-}BDHI_3}(\mathcal{A}) \leq \mathbf{Adv}_n^{\ell\text{-}BDHI_3}(\mathcal{B}).$$

Proof. The adversary $\mathcal{B}$ is given $g_i, g_i^{\alpha}, \ldots, g_i^{(\alpha^{\ell})}$ on input. It chooses $\beta \xleftarrow{\$} \mathbb{F}_p^*$, uniformly at random and computes $\left(g_2^{\alpha^j}\right)^{\beta} = g_2^{\beta\alpha^j}$, for $i = 1,2$ and $j = 0, \ldots, \ell$. Next, $\mathcal{B}$ sends $\left\{g_i^{\alpha^j}; g_2^{\beta\alpha^j}\right\}_{j=0}^{\ell}$ to $\mathcal{A}$, and obtains $\tilde{e} = \hat{e}(g_1,g_2)^{\frac{\beta}{\alpha}}$ on output. Since $\mathcal{B}$ knows β it computes $\tilde{e}^{\frac{1}{\beta}} = \hat{e}(g_1,g_2)^{\frac{1}{\alpha}}$, which is desired value. This means that breaking ℓ-BDHI$_3$ is at least as hard as breaking $(\ell,1)$-BDHI$_3$. □

2.2. Existentially Unforgeable Signature Schemes. The notion of signature schemes which are existentially unforgeable under a chosen-message attack (euf-cma) was introduced in [5]. To remind the formal definition of euf-cma security let $\Pi = (\mathscr{G}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme , $\mathcal{A}$ a PPT-adversary and n the value of a security parameter. Assume that the system parameters $\mathsf{params} \leftarrow \mathscr{G}(1^n)$ have been generated and sent to $\mathcal{A}$. Consider the experiment $\mathbf{Exp}_{\mathcal{A},\Pi}^{\mathsf{euf\text{-}cma}}$:

(1) Generate $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(\mathsf{params})$;

(2) The adversary $\mathcal{A}$ is given pk and access to the signing oracle $\mathsf{Sign}_{\mathsf{sk}}(\cdot)$, requesting signatures on as many messages as it like (it is denoted by $\mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$). Let$\{m^i\}_{i=1}^q$ be the set of queries that $\mathcal{A}$ has asked the oracle;

(3) Eventually $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$;

(4) $\mathcal{A}$ succeeds if $\mathsf{Vrfy}_{\mathsf{pk}}(m^*, \sigma^*) = 1 \wedge m^* \notin \{m^i\}_{i=1}^q$. In this case the output of the experiment is defined to be 1. Otherwise, the experiment outputs 0.

We refer to such an adversary as an euf-cma-adversary. The *advantage* of $\mathcal{A}$ in attacking the scheme Π is defined as

$$\mathbf{Adv}_{\Pi,n}^{\mathsf{euf\text{-}cma}}(\mathcal{A}) = \Pr[\mathbf{Exp}_{\mathcal{A},\Pi}^{\mathsf{euf\text{-}cma}}(1^n) = 1].$$

A signature scheme is secure if no efficient adversary can succeed in the above game with non-negligible probability.

Definition 2. A signature scheme $\Pi = (\mathscr{G}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$, is called to be *existentially unforgeable under a chosen-message attack* if for all efficient probabilistic, polynomial-time adversaries $\mathcal{A}$, there is a negligible function negl such that

$$\mathbf{Adv}_{\Pi,n}^{\mathsf{euf\text{-}cma}}(\mathcal{A}) \leq \mathsf{negl}(n).$$

Signature schemes, which are existentially unforgeable under a chosen-message attack are often called euf-cma *secure*.

3. Construction of the scheme

In this section we present the construction of our scheme MulSig. The idea is such that there are M signers, having independent keys $(\mathsf{sk}_1, \mathsf{pk}_1)$, ..., $(\mathsf{sk}_M, \mathsf{pk}_M)$, and signing the same message. The crucial matter is that the signing algorithm consists of two separated stages, making pre-signatures and generating final aggregate signature. It must be highlighted that the second stage is conducted by the system, and consequently none of the signers has already control of signature generation process. In other words, a set of all pre-signatures became a seed for an aggregate signature. The details are given in the following construction of the scheme (see also Figure 1).

Parameters setup: $(\mathscr{G})$ On input 1^n, the setup algorithm generates parameters $\mathsf{params} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_i, \hat{e}, \mathsf{Hashes})$, where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are three multiplicative cyclic groups of prime order p, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a pairing of Type 3 and $\langle g_i \rangle = \mathbb{G}_i$ with $i = 1, 2$. According to the definition, $\mathbb{G}_1 \neq \mathbb{G}_2$ and no efficiently computable isomorphism is known between $\mathbb{G}_1$ and $\mathbb{G}_2$, in either direction. The component Hashes consists of three hash functions $H_1 : \{0,1\}^* \to \mathbb{F}_p^*$, $H_2 : \mathbb{G}_T \times \mathbb{G}_T \to \{0,1\}^n$, $H_3 : \{0,1\}^* \to \mathbb{F}_p^* \times \mathbb{F}_p^*$.

Parameters setup: ($\mathscr{G}$) On input 1^n, the setup algorithm generates parameters $\mathsf{params} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_i, \hat{e}, \mathsf{Hashes})$, where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are three multiplicative cyclic groups of prime order p, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a pairing of Type 3 and $\langle g_i \rangle = \mathbb{G}_i$ with $i = 1, 2$. According to the definition, $\mathbb{G}_1 \neq \mathbb{G}_2$ and no efficiently computable isomorphism is known between $\mathbb{G}_1$ and $\mathbb{G}_2$, in either direction. The component Hashes consists of three hash functions $H_1 : \{0,1\}^* \to \mathbb{F}_p^*$, $H_2 : \mathbb{G}_T \times \mathbb{G}_T \to \{0,1\}^n$, $H_3 : \{0,1\}^* \to \mathbb{F}_p^* \times \mathbb{F}_p^*$.

Key generation: (Gen) Each signer generates a pair of random secret keys $s, r \xleftarrow{\$} \mathbb{F}_p^*$ and computes a corresponding public key $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2)$, where $\mathsf{pk}_1 \leftarrow g_1^s, \mathsf{pk}_2 \leftarrow g_2^r$.

First signing round: ($\mathsf{Sign.Round1}$) Each signer separately performs the algorithm $\mathsf{Sign.Round1}$, which takes on input a message $\mathbf{m}$ and the secret key $\mathsf{sk} = (s, r)$. It computes and outputs the following value $t \leftarrow g_2^{(s+H_1(\mathbf{m}))^{-1} \cdot r}$.

Second signing round: ($\mathsf{Sign.Round2}$) This algorithm derives an aggregate signature. Upon reception of the first-round output $\{t_i\}_{i \in [M]}$ it chooses a uniformly random nonce $\mathsf{nonce} \xleftarrow{\$} \{0,1\}^n$, takes the message $\mathbf{m}$ and performs the following computations

$$R \leftarrow \prod_{i=1}^{M} \mathsf{pk}_{2,i}, \qquad \rho \leftarrow H_3(\mathsf{nonce} \| \mathbf{m}),$$
$$\sigma_i \leftarrow t_i^{\rho},\ i \in [M], \qquad \sigma_{M+1} \leftarrow \mathsf{nonce}\ \oplus H_2\left(\hat{e}(g_1, R)^{\rho}\right)$$

It outputs $\sigma = \left(\{\sigma_i\}_{i \in [M]}, \sigma_{M+1}\right)$.

Verification: (Vrfy) Given a set of associated public keys $\{\mathsf{pk}_i\}_{i \in [M]}$, a message $\mathbf{m}$, and a (purported) signature σ, it makes pre-computations

$$\xi_i \leftarrow \hat{e}\left(\mathsf{pk}_{1,i} \cdot g_1^{H_1(\mathbf{m})}, \sigma_i\right),\ i \in [M],$$
$$\pi \leftarrow \prod_{i=1}^{M} \xi_i, \qquad \eta \leftarrow H_2(\pi) \oplus \sigma_{M+1}, \qquad \tau \leftarrow H_3(\eta \| \mathbf{m}).$$

The verifier accepts the signature if $\pi = \hat{e}\left(g_1, \prod_{i=1}^{M} \mathsf{pk}_{2,i}\right)^{\tau}$.

$\mathscr{G}(1^n)$

$\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p \in \mathcal{P}, g_1, g_2, \hat{e}$
Select three hash functions
$\mathsf{Hashes} = \{H_1, H_2, H_3\}$
$H_1 : \{0,1\}^* \to \mathbb{F}_p^*$,
$H_2 : \mathbb{G}_T \to \{0,1\}^n$
$H_3 : \{0,1\}^* \to \mathbb{F}_p^*$
$\mathsf{params} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_i, \hat{e}, \mathsf{Hashes})$
return params

$\mathsf{Gen}(1^n, \mathsf{params})$

$s, r \xleftarrow{\$} \mathbb{F}_p^*$
$\mathsf{pk}_1 \leftarrow g_1^s, \mathsf{pk}_2 \leftarrow g_2^r$
$\mathsf{sk} = (s, r),\ \mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2)$
return $(\mathsf{sk}, \mathsf{pk})$

$\mathsf{Sign.Round1}_{\mathsf{sk}}(\mathbf{m})$

$t \leftarrow g_2^{\frac{1}{s+H_1(\mathbf{m})} \cdot r}$
return t

$\mathsf{Sign.Round2}\left(\mathbf{m}, \{t_i\}_{i\in[M]}, \{\mathsf{pk}_i\}_{i\in[M]}\right)$

$R \leftarrow \prod_{i=1}^{M} \mathsf{pk}_{i,2}$
$\mathsf{nonce} \xleftarrow{\$} \{0,1\}^n$
$\rho \leftarrow H_3(\mathsf{nonce} \| \mathbf{m})$
$\sigma_i \leftarrow t_i^{\rho},\ i \in [M]$
$\sigma_{M+1} \leftarrow \mathsf{nonce} \oplus H_2\left(\hat{e}(g_1, R)^{\rho}\right)$
return $\sigma = \left(\{\sigma_i\}_{i\in[M]}, \sigma_{M+1}\right)$

$\mathsf{Vrfy}_{\{\mathsf{pk}\}}(\mathbf{m}, \sigma)$

$\xi_i \leftarrow \hat{e}\left(\mathsf{pk}_{i,1} \cdot g_1^{H_1(\mathbf{m})}, \sigma_i\right),\ i \in [M]$
$\pi \leftarrow \prod_{i=1}^{M} \xi_i$
$\eta \leftarrow H_2(\pi) \oplus \sigma_{M+1}$
$\tau \leftarrow H_3(\eta \| \mathbf{m})$
if $\pi = \hat{e}\left(g_1, \prod_{i=1}^{M} \mathsf{pk}_{i,2}\right)^{\tau}$
  return 1
else
  return 0

Figure 1. The multi-signature scheme MulSig.

In the next section we justify that the scheme MulSig is euf-cma-secure. The proof is made under the assumption that H_1 is modeled as a random oracle, meaning that it is unreal perfect mixing function.

4. Security proof

Theorem 3. *Let $\mathcal{A}$ be a* euf-cma*-adversary against* MulSig *in the random oracle model which makes at most q queries to H_1. Then there exists a q-BDHI$_3$-adversary $\mathcal{B}$ with advantage*

$$\mathbf{Adv}_n^{q\text{-}BDHI_3}(\mathcal{B}) \geq q^{-1} \cdot \mathbf{Adv}_{\Pi,n}^{\mathsf{euf\text{-}cma}}(\mathcal{A})$$

and a running time $\mathcal{O}(time(\mathcal{A}))$.

Proof. At first, we reduce the security of MulSig to hardness of the $(q,1)$-BDHI$_3$ problem. To this end, let $\mathcal{A}$ be an adversary, which attacks the scheme MulSig, and assume there are given parameters of $(q,1)$-BDHI$_3$, consisting of three cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ of prime order p, a Type 3 pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, and generators $g_i \in \mathbb{G}_i$ with $i, = 1, 2$. We shall construct an algorithm $\mathcal{B}'$ that uses $\mathcal{A}$ as a subroutine, and is aimed to attack $(q,1)$-BDHI$_3$. Therefore, according to the model, it obtains $\left\{g_i, g_i^{\alpha}, \ldots, g_i^{\alpha^q} \; ; \; g_2^{\beta}, g_2^{\beta\alpha}, \ldots, g_2^{\beta\alpha^q}\right\}$, $i = 1, 2$, on input, and must return the value $\hat{e}(g_1, g_2)^{\frac{\beta}{\alpha}}$. Since the adversary $\mathcal{A}$ serves as a black-box, the algorithm $\mathcal{B}'$ has to perfectly pretend the signing oracle; that is, the view of $\mathcal{A}$ when run by $\mathcal{B}'$ is identically distributed to the view of $\mathcal{A}$ in $\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\mathsf{MulSig}}$. Let $M \geq 2$ denote the number of signers.

Setup. The simulator sets the parameters $\mathsf{params} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \; p, \tilde{g}_i, \hat{e}, \mathsf{Hashes})$, where $\mathsf{Hashes} = \{H_1, H_2, H_3\}$ are as in Section 3 and $\tilde{g}_i$ will be defined below (see (4)). These parameters will be sent to $\mathcal{A}$. The hash function H_1 is modeled as a random oracle, and is able to be queried $q := q_{H_1}$ times by $\mathcal{A}$. In order to properly simulate the random oracle, elements $w_0, w_1, \ldots, w_{q-1} \xleftarrow{\$} \mathbb{F}_p^{\star}$ are chosen uniformly at random, and they will be served as a pool of answers to the oracle queries. Having done this, we define a polynomial $W \in \mathbb{F}_p[x]$ as follows

$$W(x) = (x + w_1)(x + w_2) \cdots (x + w_{q-1}) = \prod_{i=1}^{q-1} (x + w_i).$$

It turns to be useful to write the polynomial W as

$$W(x) = \sum_{i=0}^{q-1} a_i x^i, \quad \text{where } a_i \in \mathbb{F}_p.$$

Now, the generators $\tilde{g}_i$ can be defined; for $i = 1, 2$ we have

$$\tilde{g}_i := (g_i)^{a_0} \cdot (g_i^{\alpha})^{a_1} \cdots \left(g_i^{\alpha^{q-1}}\right)^{a_{q-1}} = g_i^{\sum_{i=0}^{q-1} a_i \alpha^i} = g_i^{W(\alpha)}.$$

Note that if $\tilde{g}_i = 1_{\mathbb{G}_i}$, then $1_{\mathbb{G}_i} = g_i^0 = g_i^{\prod_{i=1}^{q-1}(w_i+\alpha)}$ and there is easily derivable i_0 such that $w_{i_0} = -\alpha$.

The simulator chooses $s_2, r_2, \ldots, s_M, r_M \xleftarrow{\$} \mathbb{F}_p^*$, and sets the secret keys as

$$\begin{aligned} \mathsf{sk}_1 &= (\mathsf{sk}_{1,1}, \mathsf{sk}_{1,2}) = (\alpha - w_0, \beta), \\ \mathsf{sk}_j &= (\mathsf{sk}_{j,1}, \mathsf{sk}_{j,2}) = (s_j, r_j), \text{ for } j = 2, \ldots, M \end{aligned}$$

It should be born in mind that the exact values of α and β are hidden from the simulator, and, in particular, the values of $\mathsf{sk}_{1,1}$ and $\mathsf{sk}_{1,2}$ are unknown too. On the other hand, the simulator has full knowledge about remaining keys $\mathsf{sk}_2, \ldots, \mathsf{sk}_M$. Despite the drawback as for sk_1, it is possible to derive the associated public key.

$$\mathsf{pk}_{1,1} = \tilde{g}_1^{\mathsf{sk}_{1,1}} = \tilde{g}_1^{\alpha - w_0} = g_1^{\alpha \cdot W(\alpha)} \cdot \tilde{g}_1^{-w_0}.$$

It is easily seen that it is possible to compute the value of $\tilde{g}_1^{-w_0}$. For the first component of the above product, we have

$$g_1^{\alpha \cdot W(\alpha)} = (g_1^{\alpha})^{a_0} \cdot (g_1^{\alpha^2})^{a_1} \cdots (g_1^{\alpha^q})^{a_{q-1}}.$$

As $g_1^{\alpha^i}$'s and w_i's are known, $g_1^{\alpha \cdot W(\alpha)}$ is explicitly computable In the similar fashion we obtain

$$\mathsf{pk}_{1,2} = \tilde{g}_2^r = \left(g_2^{\beta}\right)^{a_0} \cdot \left(g_2^{\beta\alpha}\right)^{a_1} \cdots \left(g_2^{\beta\alpha^{q-1}}\right)^{a_{q-1}} = g_2^{\beta W(\alpha)}.$$

Therefore, formally, the simulator sets the public keys as

$$\begin{aligned} \mathsf{pk}_1 &= (\mathsf{pk}_{1,1}, \mathsf{pk}_{1,2}) = \left(g_1^{\alpha \cdot W(\alpha)} \cdot \tilde{g}_1^{-w_0}, g_2^{\beta W(\alpha)}\right), \\ \mathsf{pk}_j &= (\mathsf{pk}_{j,1}, \mathsf{pk}_{j,2}) = (g_1^{s_j}, g_2^{r_j}), \text{ for } j = 2, \ldots, M. \end{aligned}$$

$\mathbf{H_1}$-Query. The adversary $\mathcal{A}$ makes hash queries in this phase. Before getting any query, the simulator chooses $k^* \stackrel{\$}{\leftarrow} [0, q]$. Having done this it prepares a hash list L to record all queries and responses as follows

(1) At the beginning, the list L is empty;

(2) Let $\mathbf{m}_k$ be an k-th query to H_1:

  (a) If $\mathbf{m}_k$ has been already asked about, then the list L_1 consists of a pair $(\mathbf{m}_k, H_1(\mathbf{m}_k))$ and, in this case, the simulator outputs $H_1(\mathbf{m}_k)$.

  (b) Otherwise, there are considered two cases:

    (i) If $k = k^*$, then $(\mathbf{m}_{k^*}, w_0)$ is appended to the list L_1 and $H_1(\mathbf{m}_{k^*}) = w_0$ is given on output.

    (ii) If $k \neq k^*$, then then $(\mathbf{m}_k, w_0 + w_{\hat{k}})$, where $\hat{k} := k - k^* \pmod q$, is appended to the list L_1 and $H(\mathbf{m}_k) = w_0 + w_{\hat{k}}$ is given on output.

Query. The adversary $\mathcal{A}$ makes signature queries in this phase. For a signature query on $\mathbf{m}_k$, if $k = k^*$ then abort. Otherwise, $H(\mathbf{m}_k) = w_0 + w_{\hat{k}}$ and since the simulator knows secret keys $\mathsf{sk}_2, \ldots \mathsf{sk}_M$ then associated t_j's are computed according to $\mathsf{Sign.Round1}_{\mathsf{sk}_j}$ algorithm

$$t_{j,k} = \tilde{g}_2^{\frac{1}{s_j + w_0 + w_{\hat{k}}} \cdot r_j}.$$

Although the simulator does not know the exact values of $\mathsf{sk}_{1,1}$ and $\mathsf{sk}_{1,2}$, it is also possible to derive $t_{1,k}$. By (4), we have

$$\tilde{g}_2^{\beta} = \left(g_2^{\beta}\right)^{a_0} \cdot \left(g_2^{\beta\alpha}\right)^{a_1} \cdots \left(g_2^{\beta\alpha^{q-1}}\right)^{a_{q-1}} = g_2^{\beta W(\alpha)},$$

thus

$$t_{1,k} = \tilde{g}_2^{\frac{1}{\mathsf{sk}_1 + H_1(\mathbf{m}_k)} \cdot \mathsf{sk}_{1,2}} = \tilde{g}_2^{\frac{1}{\alpha + w_{\hat{k}}} \cdot \beta} = g_2^{\frac{W(\alpha)}{\alpha + w_{\hat{k}}} \cdot \beta}.$$

Let us put

$$W_{\hat{k}}(x) := \frac{W(x)}{x + w_{\hat{k}}} = \frac{1}{x + w_{\hat{k}}} \cdot \prod_{i=1}^{q-1}(x + w_i) = \prod_{\substack{i=1 \\ i \neq \hat{k}}}^{q-1}(x + w_i).$$

It is seen that there are easily computable b_i's, such that we have

$$W_{\hat{k}}(x) = \sum_{i=0}^{q-2} b_i x^i.$$

This implies that $t_{1,k}$ has the following form

$$t_{1,k} = g_2^{\beta W_{\hat{k}}(\alpha)} = \prod_{i=0}^{q-2} \left(g_2^{\beta\alpha^i}\right)^{b_i}.$$

In a second step, the simulator chooses $\mathsf{nonce} \xleftarrow{\$} \{0,1\}^n$ and finds a value of H_3 at a concatenation $\mathsf{nonce} \| \mathbf{m}_k$, we get

$$\rho \leftarrow H_3(\mathsf{nonce} \| \mathbf{m}_k).$$

Eventually, the simulator computes

$$\sigma_{j,k} \leftarrow t_{j,k}^{\rho}, \quad \text{for } j = 1, \ldots, M,$$
$$\sigma_{M+1,k} \leftarrow \mathsf{nonce} \ \oplus H_2\left(\hat{e}(g_1, g_2^{\beta} \cdot g_2^{r_2} \cdots g_2^{r_M})^{\rho}\right).$$

The tuple $(\{\sigma_{j,k}\}_{j \in [M]}, \sigma_{M+1,k})$ is a valid forgery on $\mathbf{m}_k$ and is sent to $\mathcal{A}$.

Forgery. The adversary $\mathcal{A}$ outputs a forged signature $\sigma^* = (\{\sigma_j^*\}_{j\in[M]}, \sigma_{M+1}^*)$ on a message $\mathbf{m}^*$ that has not been queried. If $\mathbf{m}^* \neq \mathbf{m}_{k^*}$, abort. Otherwise, we have $H_1(\mathbf{m}^*) = w_0$. According to the simulation, the first component of σ^* is of the form

$$\sigma_1^* = \left(\tilde{g}_2^{\frac{1}{\mathsf{sk}_{1,1}+H_1(\mathbf{m}^*)}\cdot \mathsf{sk}_{1,2}}\right)^{\rho} = \left(\tilde{g}_2^{\frac{\beta}{\alpha}}\right)^{\rho} = \left(g_2^{\frac{\beta W(\alpha)}{\alpha}}\right)^{\rho}.$$

Launching the verification algorithm Vrfy, it is easy to compute ρ; having this we obtain

$$t_1^* := (\sigma_1^*)^{\frac{1}{\rho}} = g_2^{\frac{\beta W(\alpha)}{\alpha}}.$$

These values are used to get

$$E_{\alpha,1} = \hat{e}\,(\tilde{g}_1, t_1^*) = \hat{e}\left(g_1^{W(\alpha)}, g_2^{\frac{\beta W(\alpha)}{\alpha}}\right) = \hat{e}\,(g_1, g_2)^{\frac{\beta W^2(\alpha)}{\alpha}}.$$

Note that $E_{\alpha,1}$ differs from the required solution of $(q,1)$-BDHI$_3$ problem and must be reformulated to the most suitable form. To this end, we let

$$\tilde{\tilde{g}}_{1,1} := (g_1)^{a_1}\cdot(g_1^{\alpha})^{a_2}\cdots\left(g_1^{\alpha^{q-2}}\right)^{a_{q-1}} = g_1^{\sum_{i=0}^{q-2} a_{i+1}\alpha^i} = g_1^{\frac{W(\alpha)-a_0}{\alpha}}.$$

Further define (see also (4))

$$E_{\alpha,2} = \hat{e}\left(\tilde{\tilde{g}}_1, \tilde{g}_2^{\beta}\cdot(g_2^{\beta})^{a_0}\right).$$

Since $\tilde{g}_2^{\beta}\cdot(g_2^{\beta})^{a_0} = g_2^{\beta(W(\alpha)+a_0)}$, so we use the bilinearity of $\hat{e}$ to get the following conditions

$$E_{\alpha,2} = \hat{e}\,(g_1, g_2)^{\frac{\beta(W^2(\alpha)-a_0^2)}{\alpha}} = \hat{e}\,(g_1, g_2)^{\frac{\beta W^2(\alpha)-\beta a_0^2}{\alpha}}$$

Finally, in order to obtain the solution of $(q,1)$-BDHI$_3$, it is just sufficient to divide $E_{\alpha,1}$ by $E_{\alpha,2}$. Indeed, the simulator $\mathcal{B}'$ computes

$$\left(\frac{E_{\alpha,1}}{E_{\alpha,2}}\right)^{\frac{1}{a_0}} = \hat{e}\,(g_1, g_2)^{\frac{\beta}{\alpha}}$$

as the solution to the $(q,1)$-BDHI$_3$ problem instance.

Probability of successful simulation. If the simulator $\mathcal{B}'$ successfully guesses k^*, then all signatures queried by $\mathcal{A}$ are simulatable, and signature forgery is made on the message $\mathbf{m}^*$. Therefore, the probability of breaking the problem $(q,1)$-BDHI$_3$ is q^{-1} for $q = q_{H_1}$ queries.

To sum up, we have shown that there exists a $(q,1)$-BDHI$_3$-adversary $\mathcal{B}'$ with advantage

$$\mathbf{Adv}_n^{(q,1)\text{-BDHI}_3}(\mathcal{B}') \geq g^{-1} \cdot \mathbf{Adv}_{\mathsf{MulSig},n}^{\mathsf{euf\text{-}cma}}(\mathcal{A}).$$

The running time of $\mathcal{B}'$ is $\mathcal{O}(\text{time}(\mathcal{A}))$. By Lemma 1, there is an adversary $\mathcal{B}$, with essentially the same running time as $\mathcal{B}'$, and such that

$$\mathbf{Adv}_n^{q\text{-BDHI}_3}(\mathcal{B}) \geq \mathbf{Adv}_n^{(q,1)\text{-BDHI}_3}(\mathcal{B}').$$

Then, as an immediate consequence of these estimates, we obtain

$$\mathbf{Adv}_{\mathsf{MulSig},n}^{\mathsf{euf\text{-}cma}}(\mathcal{A}) \leq q \cdot \mathbf{Adv}_n^{q\text{-BDHI}_3}(\mathcal{B}).$$

This finishes the proof. □

Conclusion 1. Since the q-BDHI$_3$ problem is computationally hard, then $\mathbf{Adv}_n^{q\text{-BDHI}_3}(\mathcal{B}) \leq \mathsf{negl}(n)$, therefore $\mathbf{Adv}_{\mathsf{MulSig},n}^{\mathsf{euf\text{-}cma}}(\mathcal{A})$ is negligible as well. This implies that MulSig is euf-cma-secure according to Definition 2.

References

[1] A. M. Antonopoulos, Mastering Bitcoin: Programming the open blockchain, *O'Reilly Media, Inc.*, 2017.

[2] M. Bellare and G. Neven, Multi-signatures in the plain public-key model and a general forking lemma, In: Proceedings of the 13th ACM conference on Computer and communications security, 2006, 390–399.

[3] D. Boneh and X. Boyen, Efficient selective-ID secure identity-based encryption without random oracles, In: International conference on the theory and applications of cryptographic techniques, 223–238.

[4] M. Drijvers, K. Edalatnejad, B. Ford, E. Kiltz, J. Loss, G. Neven and I. Stepanovs, On the security of two-round multi-signatures, In: 2019 IEEE Symposium on Security and Privacy (SP), 2019, 1084–1101.

[5] S. Goldwasser, S. Micali and R. L. Rivest, A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks, *SIAM J. Comput.* **17**, no. 2 (1988), 281–308.

[6] F. Guo, W. Susilo and Y. Mu, Introduction to security reduction, *Springer*, 2018.

[7] K. Itakura and K. Nakamura, A public-key cryptosystem suitable for digital multisignatures, *NEC Research & Development* **71** (1983), 1–8.

[8] G. Maxwell, A. Poelstra, Y. Seurin and P. Wuille, Simple schnorr multi-signatures with applications to bitcoin, *Designs, Codes and Cryptography* **87**, no. 9 (2019), 2139–2164.

[9] S. Micali, K. Ohta and L. Reyzin, Accountable-subgroup multisignatures, In: Proceedings of the 8th ACM Conference on Computer and Communications Security, 2001, 245–254.

[10] M. Michels and P. Horster, On the risk of disruption in several multiparty signature schemes, In: International Conference on the Theory and Application of Cryptology and Information Security, 1996, 334–345.

[11] A. Poelstra, Musig: A new multisignature standard, 2019, https://blockstream.com/2019/02/18/musig-a-new-multisignature-standard/.

[12] C.-P. Schnorr, Efficient signature generation by smart cards, *Journal of cryptology* **4**, no. 3 (1991), 161–174.

[13] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi and B. Ford, Keeping authorities "honest or bust" with decentralized witness cosigning, In: 2016 IEEE Symposium on Security and Privacy (SP), 2016, 526–545.

MARIUSZ JURKIEWICZ
DEPARTMENT OF CYBERNETICS
MILITARY UNIVERSITY OF TECHNOLOGY
2 GEN. S. KALISKI ST, WARSAW
POLAND

E-mail: mariusz.jurkiewicz@wat.edu.pl