Insecurity of MuSig and BN Multi-Signatures with Delayed Message Selection

Sela Navot[1]

March 2024

Abstract

This note reveals a vulnerability of MuSig and BN multi-signatures when used with delayed message selection. Despite the fact that both schemes can be correctly implemented with pre-processing of the first two signing rounds before the message to sign is selected, we show that they are insecure (i.e. not existentially unforgeable against chosen message attacks) when the message selection is deferred to the third signing round and when parallel signing sessions are permitted. The attack, which uses the algorithm by Benhamouda et al. to solve the ROS problem [4], is practical and runs in polynomial time.

1 Introduction

Multi-signature schemes allow a group of signers to provide a succinct joint signature for an agreed upon message. However, pairing-free discrete-log based multi-signature schemes require multiple communication signing rounds between the signers. To make them closer to non-interactive schemes, some schemes allow all but one of the communication rounds to be completed before the message to be signed or the identity of all the signers have been determined. A prominent example is MuSig2 [9], which consists of two signing rounds and is proved secure when the first round is pre-processed before the selection of the message and the signing group, and is thus nearly round optimal.

BN [2] and MuSig [7] are practical *three* round multi-signature schemes. In those schemes the message to be signed is not used by the signers until the third signing round, and hence they can be executed correctly when the first two rounds are pre-processed before the message is determined. This raises the question of whether these two schemes are secure in such a setting. This note shows that they are not, and presents a practical polynomial time attack against those schemes when the second round is pre-processed before the selection of the message.

MuSig, BN signatures, and message dependence. As mentioned, BN and MuSig are similar multi-signature schemes consisting of three signing rounds, where the signers do not use the message to be signed until the third signing round. This suggest that the scheme may be used in a setting where the choice of the message to be signed is delayed to after the second signing round.

Prior literature contains some ambiguities regarding whether this is the case. The MuSig paper [7], in particular, does not explicitly state at which round the message needs to be determined. While they mention that their security definition is identical to that of [2] and that the first two signing round of MuSig are identical to those of the BN-Scheme [2], which implies implicitly that the message is selected before the first signing round (even though it is not used for two more rounds), this is never made explicit in the description of MuSig. A later paper proving the security of MuSig [1] presents a more detailed description of the scheme in which the message is selected

[1] University of Washington, Seattle, Washington. Email: `senavot@cs.washington.edu`.

before the first signing round, but there is no justification as to why early message selection is needed. We show that message selection before the second signing round is essential to the security of MuSig.

The BN-Scheme description [2] is more detailed and precise, and it explicitly states that the message is selected before the first signing round. There is no justification, however, as to why the message needs to be determined two signing rounds before it is first used by the signers. We show that as is the case with MuSig, determining the message before the second signing round is necessary for security.

We emphasise that nothing wrong was stated or proved in the mentioned papers. However, due to practical interest in delaying message selection, we use this note to highlight the vulnerabilities it causes and, more generally, the value of precise protocol descriptions.

<u>The attacks.</u> In a blog post [8], Nick shows that if the first two signing rounds are pre-processed before the message is selected then MuSig is vulnerable to an attack in sub-exponential time using Wagner's algorithm [11]. We expand on this result by presenting a similar sub-exponential attack against BN multi-signatures when used with pre-processing of the first two signing rounds. Then, we present a similar attack that can be executed in polynomial time against BN and MuSig, which completely breaks the existential unforgeable against a chosen message attacks of those schemes (when used with delayed message selection). The polynomial attacks use the ideas from the sub-exponential attacks and the algorithm solving the ROS problem from [4].

In all of our attacks the adversary initiates multiple signing sessions concurrently (say ℓ sessions) and observes the execution of the first two signing rounds. Then, the adversary chooses the message to be signed in each of the signing sessions and completes the session, and is able to use the resulting multi-signatures to construct a signature on a message of their choice. For the polynomial time attacks it is sufficient that $\ell \geq \lceil \log(p) \rceil$, where p is the order of the underlying group.

Our polynomial time attacks not only break existential unforgeability, but allow the adversary to forge a multi-signature for a message of their choice. Furthermore, the adversary in the attacks against MuSig only needs to control the execution of the protocol and pick the messages to be signed in the ℓ legitimate sessions, but does not need to corrupt any of the signers. For our attacks against the BN scheme, the adversary must corrupt all but one of the signers. In all attacks the messages for which the adversary obtains legitimate signatures can also be of the adversaries choice (and thus the adversary can choose messages the signers are willing to sign), as long as they are selected after the second round from a set of multiple possible messages.

<u>This note.</u> As Bellare and Dai claim in [1], we believe that much of the ambiguity and security issues found in multi-signatures stem from lack of detail in the security definitions and syntax of multi-signature schemes. This includes the ambiguity regarding at which round the message to sign is selected in MuSig, which we address in this report. Hence, we begin with a detailed specifications, syntax, and unforgeability definitions for multi-signature schemes, as well as a detailed description of the secure and the insecure versions of MuSig.

Next, we present our attacks, which is our main contribution in this report. The attacks section is intended to stand alone and can be read before the specifications and security definition sections.

2 Preliminaries

<u>Notation.</u> For a positive integer n, we use $\mathbb{Z}_n$ to refer to the ring of integers modulo n equipped with modular addition and multiplication. $\mathbb{G}$ denotes a large finite groups with a known order and we use multiplicative notation for group operations unless otherwise noted. Logarithms are to the base 2 unless otherwise noted.

In our psuedo-code, we use $\leftarrow$ to denote assignment and use $\leftarrow\!\$$ for randomized assignment. In particular, we write $x \leftarrow\!\$\, S$ to denote assigning a uniformly random element of a finite set S to x and $x \leftarrow\!\$\, R(x_1, \dots)$ to denote executing a randomized algorithm R with input $x_1, \dots$ and a uniformly random coin and assigning the output to x. We use $\perp$ to denote an error value, and we use subscripts for array indexing. All variables are assumed to be uninitialized until assigned a value.

GAMES FRAMEWORK. For our correctness and security definitions, we use the game based framework from [3], which is used in the context of multi-signatures in [1].

Those definitions are described as an INIT and FIN procedures, any number of oracles, and a security parameter. When a randomized algorithm A (often called adversary) plays a game Gm, which we denote by Gm(A), A is executed with the output of INIT as its input. A may call the oracles (all procedures except INIT and FIN), and such calls are counted towards the runtime of A. Then, FIN is executed with the output of A and returns the output of the game. We say that A is a polynomial time adversary if its time and space complexity is polynomial with respect to the security parameter of the game.

We only consider games that have an underlying group (i.e. the group $\mathbb{G}$ used to instantiate the scheme), and unless otherwise noted the security parameter is $\log(p)$ where p is the order of $\mathbb{G}$.

3 Multi-Signatures

3.1 Specifications

A multi-signature scheme allows a group of signers to provide a succinct joint signature for an agreed upon message. More specifically, a valid multi-signature by a group of n signers intends to prove that each of the n signers have participated in the signing protocol in order to sign this specific message with this group of signers. Note the distinction of multi-signatures from signature aggregation techniques which compress signatures from multiple signers into a single signature but the signers do not need to know the identity of all signers in the group in the time of signing.

Multi-signature schemes are expected to be unforgeable in the plain public key model, which denotes the setting where each signer has a public key and is not required to prove ownership of an associated secret key or participate in a distributed key generation protocol [2]. This allows signers to use the same public key to sign multi-signatures with different groups.

We will provide formal syntax for multi-signatures later in this section.

KEY AGGREGATION. A multi-signature scheme supports key aggregation if a multi-signature can be verified with respect to a single short key (the aggregate key of the signing group), as opposed to requiring the complete list of all signers public keys for verification. In particular, in the case of MuSig [7] and MuSig2 [9], the resulting multi-signature is an ordinary Schnorr Signature that can be verified with respect to the aggregate key, making the scheme useful on platforms that support Schnorr Signatures.

FORMAL SYNTAX AND CORRECTNESS. A multi-signature scheme MS is a collection of algorithms $\mathsf{MS.Kg}$, $(\mathsf{MS.Sign}_r)_{r=1}^{\mathsf{MS.nr}}$, and $\mathsf{MS.Verify}$, where nr is the number of signing rounds specified by the scheme. A scheme also specifies the last interactive round $\mathsf{MS.lir}$, after which it is possible to construct a multi-signature without knowledge of the signers secret information (often the last round that needs to be completed by all signers). It is required for a secure scheme that by or on the last interactive round the message to sign and public keys of the signing group have been determined. A scheme may also have a $\mathsf{MS.KeyAgg}$ algorithm accompanied by $\mathsf{MS.AggVer}$ for key aggregation (of the verification key) and for verification using the aggregated key respectively, in which case it is said that MS supports key aggregation. The intent of the algorithms is as follows:

Key generation: The randomized algorithm $\mathsf{MS.Kg}$ is used for key generation by each signing party individually. It takes no input and outputs a secret-public key pair.

Signing: A collection of algorithms $(\mathsf{MS.Sign}_r)_{r=1}^{\mathsf{nr}}$ specify the procedure for each signing round that is run by each signing party, and $\mathsf{MS.nr}$ (the number of signing rounds) is specified by the scheme. Each round takes a subset of the following as input: a message, a vector of public keys, the output of previous signing rounds, and some other information saved in the state of at most one signer (including the secret key). It also takes the signing session index to identify which signing session this round corresponds to. The algorithm produces an output, as well as updates the state of the signing party. The signature is the output of the last round $\mathsf{Sign}_{\mathsf{nr}}$. These algorithms may be randomized.

Key aggregation: If the scheme supports key aggregation, the algorithm $\mathsf{MS.KeyAgg}$ takes a list of n public keys $(vk_i)_{i=1}^{n}$ as input and outputs a single aggregate verification key.

Verification: If MS does not support key aggregation then it has an algorithm $\mathsf{MS.Verify}$ that takes a list of public keys, a signature, and a message as input and returns a boolean value signifying whether the signature is valid. If MS supports key aggregation it has the algorithm $\mathsf{MS.AggVer}$ with the same functionality that takes an aggregated public key as input instead of a list of public keys. In this case, a standard Verify algorithm can be obtained by composing the aggregated verification and the key aggregation algorithms, i.e. setting $\mathsf{MS.Verify}((vk_i)_{i=1}^{n}, m, \sigma) = \mathsf{MS.AggVer}(\mathsf{MS.KeyAgg}((vk_i)_{i=1}^{n}), m, \sigma)$. Hence, without loss of generality we will only consider $\mathsf{MS.Verify}$ in the correctness and security definitions.

The signers are entities that may run any of the algorithms above, and have a state st, which may change throughout the protocol. In particular, the signer holds their private key st_{sk} and public key st_{vk}, as well as the information and status st_s regarding each signing execution $s$which must include the following components:

$\mathsf{st}_s.n$: the number of parties in the current execution.

$(\mathsf{st}_s.vk_j)_{j=1}^{n}$: the public key of the signers in the group.

$\mathsf{st}_s.\mathsf{msg}$: the message being signed in the current execution.

$\mathsf{st}_s.\mathsf{rnd}$: the current round of the execution. It is assumed and required that a signer refuses requests to run the algorithm Sign_ℓ if $\ell \neq \mathsf{st.rnd}+1$ and that an execution of a signing round increments this field by one.

$\mathsf{st}_s.\mathrm{me}$: the index of the party in the public key vector.

$\mathsf{st}_s.\mathsf{rej} \in \{\mathsf{true}, \mathsf{false}\}$: whether the execution of the protocol have been aborted. If any call of the form $(\sigma, st_s) \leftarrow \mathsf{Sign}_r(m, st_s)$ returns $\bot$, it is assumed and expected that $\mathsf{st}_s.\mathsf{rej}$ is set to false.

The state may also include other information such as the output of previous signing rounds or the discrete log of a nonce.

In an honest execution of a multi-signature scheme, each party runs the key generation algorithm independently. To sign a message, the signing rounds are executed by each party in sequential order, with the output of each signing round from all participating signers used as part of the input for consequent signing rounds. A multi-signature scheme satisfies (perfect) correctness if an honest execution of the scheme always produces a valid signature. We provide an algorithm for an honest execution of a multi-signature scheme and a correctness game definition in figure 1.

```
Algorithm $\mathsf{Exec}_{\mathsf{MS}}((vk_i)_{i=1}^n, (sk_i)_{i=1}^n, m)$:
For $i = 1, \ldots, n$ do:
    $i.\mathsf{st}.sk \leftarrow sk_i$; $i.\mathsf{st}.vk \leftarrow vk_i$; $i.\mathsf{st}.\mathrm{me} \leftarrow i$
    $i.\mathsf{st}_1.n \leftarrow n$; $(i.\mathsf{st}_1.vk_j)_{j=1}^n \leftarrow (vk_j)_{j=1}^n$
$\mathrm{partialSigs} \leftarrow (0)_{i=1}^n$ // output of current round
For $r = 1, \ldots, \mathsf{MS.nr}$ do:
    For $s = 1, \ldots, n$ do:
        $(\sigma_s, s.\mathsf{st}) \leftarrow\$ \mathsf{MS.Sign}_r(s.\mathsf{st}, (vk_i)_{i=1}^n, m, \mathrm{partialSigs})$
    $\mathrm{partialSigs} \leftarrow (\sigma_i)_{i=1}^n$
Return $\sigma_1$
```

```
Game $\mathbf{G}^{\text{ms-cor}}_{\mathsf{MS},n,m}$
FIN:
For $i = 1, \ldots, n$ do:
    $(vk_i, sk_i) \leftarrow\$ \mathsf{MS.Kg}()$
$\sigma \leftarrow\$ \mathsf{Exec}_{\mathsf{MS}}((vk_i)_{i=1}^n, (sk_i)_{i=1}^n, m, 1)$
Return $\mathsf{MS.Verify}((vk_i)_{i=1}^n, m, \sigma)$
```

Figure 1: **Left:** Procedure specifying an honest execution of the signing protocol for a multi-signature scheme MS. Note that signing rounds may only use a subset of the provided input. **Right:** A game specifying the correctness of a scheme. A scheme MS satisfies perfect correctness if $\Pr[\mathbf{G}^{\text{ms-cor}}_{\mathsf{MS},n,m} \Rightarrow \mathsf{true}]$ for each natural number n and message m that is supported by MS.

3.2 Existential Unforgeability

We use a game based Existential Unforgeability Against Chosen Message Attack (MS-EUF-CMA) definition, which allows the scheme to define which signing rounds are message dependent.

In the game, there is one honest signer with an honestly generated signing key (unknown to the adversary) and verification key. An adversary can interact with the honest signer via a signing oracle by providing input of the adversary's choice, including the message to be signed and the group of signers to sign the message with. Those interactions can happen concurrently. The adversary wins if they can provide a non-trivial valid multi-signature, where non-trivial means that it is valid for a message and signing group that the honest signer did not participate in the signing protocol to provide a signature for. A formal definition of this game is presented in figure 2.

A scheme is existentially unforgeable if no efficient adversary can win the described game with non-negligible probability.

At which signing round is a forgery trivial. As mentioned in the syntax section, a scheme specifies its last interactive round $\mathsf{MS.lir}$. It is expected that after querying the signing oracle to complete the last interactive signing round then the adversary can produce a multi-signature, but not before. Thus, only at a call to the signing oracle for the last interactive round we record that a legitimate multi-signature has been provided for the corresponding message and signing group.

Note that our definition is stricter than that of [1] and [7], which consider a forgery trivial if the adversary has initiated a signing session with the message and signing group for which the forgery is valid. Their definition does not apply to schemes in which the message is selected after the first signing round. We believe that this is one of the reasons that those papers did not consider the security of MuSig under delayed message selection.

Setting for our attacks. Note that our attacks against MuSig with delayed message selection can be done by weaker adversaries than those allowed in the existential unforgeability definition. In particular, the adversary only needs to observe parallel signing sessions and control which message will be signed, but can succeed against any group of honest signers even if none of them is corrupt. Our attacks against the BN-Scheme requires the adversary to corrupt all but one of the signers. In both attacks the adversary can forge a multi-signature for a message of their choice.

```
Game $\mathbf{G}^{\text{ms-euf-cma}}[\mathsf{MS}]$

$\textsc{Init}()$:
  $(vk, sk) \leftarrow\$ \mathsf{MS.Kg}$
  $\mathrm{S} \leftarrow$ Empty Dictionary
  $u \leftarrow 0$ // Signing Session Index
  Return $vk$

$\textsc{NewSignSession}()$:
  $u \leftarrow u + 1$
  $\mathsf{st}_u.\mathsf{rnd} \leftarrow 1$
  Return $u$

$\textsc{SignO}_{\mathsf{rnd}}(s,$ SOME SUBSET OF $\{m, (k, (vk_i)_{i=1}^n), \mathrm{partialSigs}\})$:
  // Defined for each $\mathsf{rnd} \in \{1, \ldots, \mathsf{MS.nr}\}$
  If $\mathsf{st}_s.\mathsf{rnd} \neq \mathsf{rnd}$:
    Return $\perp$
  If $\mathsf{st}_s.m$ uninitialized and $m$ is provided:
    $\mathsf{st}_s.m \leftarrow m$
  If $\mathsf{st}_s.k$ and $\mathsf{st}_s.n$ and $(\mathsf{st}_s.vk_i)_{i=1}^n$ uninitialized:
    If $(k, (vk_i)_{i=1}^n)$ provided:
      If $vk_k \neq vk$ return $\perp$
      $\mathsf{st}_s.n \leftarrow n$; $\mathsf{st}_s.k \leftarrow k$
      For $i$ from 1 to $n$: $\mathsf{st}_s.vk_i \leftarrow vk_i$
  $(\sigma, \mathsf{st}_s) \leftarrow\$ \mathsf{MS.Sign}_{\mathsf{rnd}}(\mathsf{st}_s.m, (\mathsf{st}_s.vk_i)_{i=1}^n, \mathsf{st}_s.sk, \mathrm{partialSigs})$
  If $\mathsf{rnd} = \mathsf{MS.lir}$ : // on last interactive signing round
    If $\sigma \neq \perp$:
      $\mathrm{S} \leftarrow \mathrm{S} \cup \{(\mathsf{st}_s.m, (\mathsf{st}_s.vk_i)_{i=1}^n)\}$
  Return $\sigma$

$\textsc{Fin}(k, (vk_i)_{i=1}^n, m, \sigma)$:
  If $vk_k \neq vk$:
    Return $\mathsf{false}$
  If $(m, (vk_i)_{i=1}^n) \in \mathrm{S}$:
    Return $\mathsf{false}$
  Return $\mathsf{MS.Verify}((vk_i)_{i=1}^n, m, \sigma)$
```

Figure 2: Game used to define the strong unforgeability of a multi-signature scheme MS. It is secure if $\mathbb{P}[\mathbf{G}^{\text{ms-euf-cma}}[\mathsf{MS}](A) \Rightarrow \mathsf{true}]$ is negligible for every randomized polynomial time adversary A.

4 The Schemes

4.1 MuSig

We will first describe the scheme informally. A formal description of the scheme using our syntax for multi-signatures can be found in figure 3 for both the secure and the insecure variant with delayed message selection.

We emphasise that we only consider variants of the three-round MuSig scheme, and not the original two-round scheme which is long known to be insecure (irreparable bug in proof found by [6], efficient attack by [4]).

THE SCHEME. The scheme involves a multiplicative group $\mathbb{G} = \langle g \rangle$ of prime order p and three hash functions: H_{com}, H_{sign}, and H_{agg} with codomain $\mathbb{Z}_p$ that are used for commitments, signing, and key aggregation respectively.

In key generation, each signing party generates a private key $sk \leftarrow\$ \mathbb{Z}_p$ and a public key $vk \leftarrow g^{sk}$. The aggregate public key for a group of n signers with public keys $vk_1, \ldots, vk_n$ is computed by

$$\tilde{vk} \leftarrow \prod_{i=1}^{n} vk_i^{H_{\text{agg}}(i, vk_1, \ldots, vk_n)}$$

In the first signing rounds, each signer k chooses $r_k \leftarrow\$ \mathbb{Z}_p$, computes $R_k \leftarrow g^{r_k}$, and sends a commitment $t_k \leftarrow H_{\text{com}}(R_k)$ to all the other signers. In the second round, the signer receives the commitments from all other signers $t_1, \ldots, t_n$, and sends R_k to all other signers. In the third round, the signer receives nonces $R_1, \ldots, R_n$ from all the signers and verifies the commitments by checking that $t_i = H_{\mathrm{com}}(R_i)$ for each i. Then, they compute $R \leftarrow \prod_{i=1}^n R_i$, the aggregate public key $\tilde{vk}$ as described above, and a challenge $c \leftarrow H_{\mathsf{sign}}(\tilde{vk}, R, m)$. Then, they outputs a signature share $z_k \leftarrow r_k + sk_k \cdot c \cdot H_{\mathrm{agg}}(k, vk_1, \ldots, vk_n)$. Now, any of the signer can output the multi-signature (R, z) where $R \leftarrow \prod_{i=1}^n R_i$ and $z \leftarrow \sum_{i=1}^n z_i$.

A signature (R, z) is valid with respect to an aggregated verification key $\tilde{vk}$ and a message m if and only if

$$g^z = R \cdot \tilde{vk}^{H_{\mathsf{sign}}(\tilde{vk}, R, m)}$$

Perfect correctness is easy to verify. Furthermore, note that the verification of a MuSig multi-signature with respect to an aggregated key $\tilde{vk}$ is identical to the verification of a standard Schnorr signature, adding to the appeal of the MuSig scheme.

Which signing rounds are message dependent. The signers in MuSig do not use the message in the first two signing rounds. Thus, it is natural to ask whether the message needs to be determined at the initiation of the protocol? Alternatively, is it possible to pre-process the first two signing rounds, and thus have only one interactive signing round when a message to sign is detemined, which gives us an almost round optimal scheme? This property is claimed by MuSig2 [9] and MuSig2-H [10], for example. The MuSig paper [7] does not provide an explicit answer to this question.

We show that the answer is no. The signers must associate each signing execution with a message and a signing group at least before the second round (the "reveal" round of the nonce shares). In other words, the signers must store the message to be signed before the second round even though it is not used until the third round. Otherwise, if an adversary can see the partial nonces after the second signing round and only then choose the message, they can forge a signature for any message of their choice and completely break existential unforgeability.

MuSig security proofs. MuSig is proven secure in [7][5][1] under the discrete log assumption when the hash functions are modeled as a random oracle. None of those papers consider whether MuSig can be used with delayed message selection.

4.2 BN Multi-Signatures

The BN scheme [2] is an older scheme containing three interactive signing round. It is very similar to MuSig but without the key aggregation, and is also proven secure using the discrete log assumption in the plain public key model when the message is determined before the second signing round. We will describe the scheme informally, but a thorough description of the scheme can be found in [2].

The scheme involves two hash functions: H_{com} and H_{sign} with codomain $\mathbb{Z}_p$ that are used for commitments and signing respectively, and a multiplicative group $\mathbb{G} = \langle g \rangle$ of prime order p.

Key generation is identical to MuSig: each signing party generates a private key $sk \leftarrow_\$ \mathbb{Z}_p$ and a public key $vk \leftarrow g^{sk}$.

The first two signing rounds are also identical to MuSig: in the first round each signer k chooses $r_k \leftarrow_\$ \mathbb{Z}_p$, computes $R_k \leftarrow g^{r_k}$, and sends a commitment $t_k \leftarrow H_{\mathrm{com}}(R_k)$ to all the other signers. In the second round, the signer receives the commitments from all other signers $t_1, \ldots, t_n$, and sends R_k to all other signers. The difference from MuSig is in the third round, where the challenge each signer uses to sign is different from MuSig. In the third round, the signer k receives nonces $R_1, \ldots, R_n$ from all the signers and verifies the commitments by checking that $t_i = H_{\mathrm{com}}(R_i)$ for each i. Then, they compute $R \leftarrow \prod_{i=1}^n R_i$, and a challenge $c_k \leftarrow H_{\mathsf{sign}}(X_k, R, X_1, \ldots, X_n, m)$.

```
$\boxed{\text{MuSig}[H_{\text{com}}, H_{\text{agg}}, H_{\mathsf{sign}}, \mathbb{G}, g, p]:}$

$\dbox{\text{InsecureMuSig}[H_{\text{com}}, H_{\text{agg}}, H_{\mathsf{sign}}, \mathbb{G}, g, p]:}$

$\mathsf{nr} = 4$

$\mathsf{lir} = 3$

$\mathsf{KeyGen}()$:
    $u \leftarrow 0$ // signing sessions counter
    $x \leftarrow\$ \mathbb{Z}_p$; $X \leftarrow g^x$
    $\mathsf{st}.sk \leftarrow x$; $\mathsf{st}.vk \leftarrow X$
    Return $(sk = x, vk = X)$

$\mathsf{KeyAgg}(vk_1, \ldots, vk_n)$:
    Return $\prod_{i=1}^{n} vk_i^{H_{\text{agg}}(i, vk_1, \ldots, vk_n)}$

$\mathsf{Sign}_1(\boxed{k, m, vk_1, \ldots, vk_n})$:
    $u \leftarrow u + 1$; $\mathsf{st}.u \leftarrow \emptyset$
    $\mathsf{st}_u.\mathsf{rnd} \leftarrow 1$
    $\boxed{\mathsf{Save\_Session\_Params}(k, m, vk_1, \ldots, vk_n, u)}$
    $\mathsf{st}_u.r \leftarrow\$ \mathbb{Z}_p$; $\mathsf{st}_u.R \leftarrow g^{\mathsf{st}_u.r}$
    $\mathsf{st}_u.t \leftarrow H_{\text{com}}(\mathsf{st}_u.R)$
    $\mathsf{st}_u.\mathsf{rnd} \leftarrow 2$
    Return $\mathsf{st}_u.t$

$\mathsf{Sign}_2(t_1, \ldots, t_n, j, \dbox{k})$:
    If $\mathsf{st}_j.\mathsf{rnd} \neq 2$: Return $\bot$
    $\dbox{\mathsf{st}_j.k \leftarrow k;\ \mathsf{st}_j.n \leftarrow n}$
    If $t_{\mathsf{st}_j.k} \neq \mathsf{st}_j.t$ or $n \neq \mathsf{st}_j.n$:
        Return $\bot$
    For $i$ from 1 to $n$:
        $\mathsf{st}_j.t_i \leftarrow t_i$
    $\mathsf{st}_j.\mathsf{rnd} \leftarrow 3$
    Return $\mathsf{st}_j.R$

$\mathsf{Sign}_3(R_1, \ldots, R_n, j, \dbox{m, vk_1, \ldots, vk_n})$:
    If $\mathsf{st}_j.\mathsf{rnd} \neq 3$: Return $\bot$
    If $\exists i$ such that $\mathsf{st}_j.t_i \neq H_{\text{com}}(R_i)$:
        Return $\bot$
    $\dbox{\mathsf{Save\_Session\_Params}(\mathsf{st}_j.k, m, vk_1, \ldots, vk_n, j)}$
    $\tilde{vk} \leftarrow \prod_{i=1}^{n} \mathsf{st}_j.vk_i^{H_{\text{agg}}(i, \mathsf{st}_j.vk_1, \ldots, \mathsf{st}_j.vk_n)}$
    $\tilde{R} \leftarrow \prod_{i=1}^{n} R_i$
    $c \leftarrow H_{\mathsf{sign}}(\tilde{R}, \tilde{X}, \mathsf{st}_j.m)$
    $z \leftarrow \mathsf{st}_j.r + \mathsf{st}.sk \cdot c \cdot H_{\text{agg}}(\mathsf{st}_j.k, \mathsf{st}_j.vk_1, \ldots, \mathsf{st}_j.vk_n)$
    $\mathsf{st}_j.\mathsf{rnd} \leftarrow 4$
    Return $(R, z)$

$\mathsf{Sign}_4(R, z_1, \ldots, z_n, j)$:
    If $\mathsf{st}_j.\mathsf{rnd} \neq 4$: Return $\bot$
    $\mathsf{st}_j.\mathsf{rnd} \leftarrow 5$
    Return $(R, \sum_{i=1}^{n} z_i)$

$\mathsf{AggVer}(m, \sigma, \tilde{vk})$:
    $(R, z) \leftarrow \sigma$
    Return $[g^z = R \cdot \tilde{vk}^{H_{\mathsf{sign}}(\tilde{X}, R, m)}]$

$\mathsf{Save\_Session\_Params}(k, m, vk_1, \ldots, vk_n, j)$:
    // private helper method storing parameters
    If $vk_k \neq \mathsf{st}.vk$: Return $\bot$
    $\mathsf{st}_j.m \leftarrow m$
    $\mathsf{st}_j.n \leftarrow n$
    For $i$ from 1 to $n$: $\mathsf{st}_j.vk_i \leftarrow vk_i$
    $\mathsf{st}_j.k \leftarrow k$
```

Figure 3: A description of the MuSig scheme over a group $\mathbb{G} = \langle g \rangle$ of order p. The secure variant where all rounds are message dependent contains all but the dashed boxes, and the insecure variant with delayed message selection contains all but the solid boxes. The fourth signing round is often omitted since it can be performed by any observer of the protocol.

Then, they outputs a signature share $z_k \leftarrow r_k + sk_k \cdot c_k$. Now, any of the signer can output the multi-signature (R, z) where $R \leftarrow \prod_{i=1}^{n} R_i$ and $z \leftarrow \sum_{i=1}^{n} z_i$.

Verification requires the message m, the signature $\sigma = (R, z)$, and all the signers public keys $(X_1, \ldots, X_n)$. Now, the verifier can compute $c_i \leftarrow H_{\mathsf{sign}}(X_i, R, X_1, \ldots, X_n, m)$ for each $i \in \{1, \ldots, n\}$ and output true if and only if $g^z = R \prod_{i=1}^{n} X_i^{c_i}$.

As with MuSig, the message is not used in the protocol until the third signing round of the scheme. However, as our attack shows, it is needed for security that the message being signed is selected before the second signing round.

5 The Attacks

Here we present our attacks against MuSig and BN Multi-Signatures when used with delayed message selection. We will first present a sub-exponential attack against each scheme. Then, we build on the ideas from the sub-exponential attack and use the algorithm from [4] to design a more efficient polynomial time attack in the same setting. Note that the sub-exponential has no advantage over the more efficient polynomial time attack, and therefore is only presented to introduce the ideas used in the more complicated polynomial time attack.

Suppose the first two rounds of MuSig and BN Multi-Signatures (the commitments and revealing the nonces rounds) are executed before message selection. A formal description of this insecure version of MuSig and comparison with the secure version is provided in figure 3. We will describe the attacks when executed with two signers, but it is straightforward to generalize it to a setting with more signers.

5.1 Sub-Exponential Attack using Wagner's Algorithm

Wagner [11] presented an algorithm that solves the generalized birthday problem in $\mathbb{Z}_p$ in sub-exponential time. We will present a simple attack that uses Wagner's Algorithm as a black box against MuSig with delayed message selection. The variation of the birthday problem that is useful to us is the following:

> **The generalized birthday problem:** Given $c \in \mathbb{Z}_p$ and ℓ lists (of arbitrary length) $L_1, \ldots, L_\ell$ of elements drawn uniformly and independently at random from $\mathbb{Z}_p$, find $c_1 \in L_1, \ldots, c_\ell \in L_\ell$ such that $\sum_{i=1}^{\ell} c_i \equiv c \pmod{p}$.

Wagner shows that this problem can be solved with high probability in $O(\ell \cdot 2^{\lceil \log p \rceil / (1 + \lfloor \log(\ell+1) \rfloor)})$ time and space complexity. This results in sub-exponential run-time for large ℓ, though still impractical for large enough p. For this attack we also assume that the hash functions used by the scheme behave like random functions, though this assumption is not needed for the more complicated polynomial time attack.

The Attack Against MuSig

This attack against MuSig is presented in [8], and is included here for the sake of completion.

Let S_1 and S_2 be the signers with private keys x_1 and x_2 and public key X_1 and X_2 respectively. Let $\tilde{X} = X_1^{H_{\text{agg}}(1, X_1, X_2)} \cdot X_2^{H_{\text{agg}}(2, X_1, X_2)}$ denote the aggregate verification key. Let ℓ be an integer that can be adjusted to optimize the runtime, and let m be the message for which the adversary wishes to forge a signature.

Now, the adversary begins ℓ signing sessions and observes the first two signing rounds to obtain an aggregate nonce $R_i = R_{i,1} \cdot R_{i,2}$ for each $i \in \{1, \ldots, \ell\}$. The adversary calculates $R \leftarrow \prod_{i=1}^{\ell} R_i$ and a challenge $c = H_{\mathsf{sign}}(\tilde{X}, R, m)$. Now, define ℓ lists $L_1, \ldots, L_\ell$ such that $L_{i,j} = H_{\mathsf{sign}}(\tilde{X}, R_i, m_j)$ where the m_j are some arbitrary messages that are different for different j.

The adversary can now use Wagner's algorithm to find elements $c_i = L_{i,j_i} \in L_i$ for each list such that $\sum_{i=1}^{\ell} c_i = c \pmod{p}$. Next, the adversary finishes each signing session i to sign the message m_{j_i}, obtaining a valid multi-signature $\sigma_i = (R_i, z_i)$ for the message m_{j_i}. Now, the adversary outputs the signature $\sigma = (R, \sum_{i=1}^{\ell} z_i)$ where summation is mod p, which we claim is a valid signature for the message m under the aggregate verification key $\tilde{X}$.

<u>Validity of forged signature.</u> We wish to verify that $\sigma = (R, \sum_{i=1}^{\ell} z_i)$ is a valid signature for the message m under the aggregate verification key $\tilde{X}$. Thus, we must show that

$$g^{\sum_{i=1}^{\ell} z_i} = R \cdot \tilde{X}^c$$

Note that (by the correctness of the signing protocol), $\sigma_i = (R_i, z_i)$ is a valid signature for the message m_{j_i} for each $i \in \{1, \ldots, \ell\}$, and thus $g^{z_i} = R_i \cdot \tilde{X}^{c_i}$. This means that

$$\prod_{i=1}^{\ell} g^{z_i} = \prod_{i=1}^{\ell} R_i \tilde{X}^{c_i}$$

or equivalently

$$g^{\sum_{i=1}^{\ell} z_i} = R\tilde{X}^{\sum_{i=1}^{\ell} c_i}$$

However, by construction (using Wagner's Algorithm) we know that $\sum_{i=1}^{\ell} c_i = c$, and therefore we conclude that

$$g^{\sum_{i=1}^{\ell} z_i} = R\tilde{X}^c$$

which is what we wanted to prove.

The Attack Against BN Multi-Signatures

As before, let S_1 and S_2 be the signers with private keys x_1 and x_2 and public key X_1 and X_2 respectively. Also suppose that S_2 is corrupt and thus their secret key x_2 is known to the adversary. Let ℓ be an integer that can be adjusted to optimize the runtime, and let m be the message for which the adversary wishes to forge a signature.

Now, the adversary begins ℓ signing sessions and executes the first two signing rounds as described in the protocol to obtain nonce shares $R_{i,1}$ and $R_{i,2} = g^{r_{i,2}}$ and an aggregate nonces $R_i = R_{i,1}R_{i,2}$ for $i \in \{1, \ldots, \ell\}$. The adversary calculates $R \leftarrow \prod_{i=1}^{\ell} R_i$ and challenges $c_1 = H_{\mathsf{sign}}(X_1, R, X_1, X_2, m)$ and $c_2 = H_{\mathsf{sign}}(X_2, R, X_1, X_2, m)$. Now, define ℓ lists $L_1, \ldots, L_\ell$ such that $L_{i,j} = H_{\mathsf{sign}}(X_1, R_i, X_1, X_2, m_j)$ where the m_j are arbitrary messages that are different for different j.

The adversary can now use Wagner's algorithm to find elements $c_{i,1} = L_{i,j_i} \in L_i$ for each list such that $\sum_{i=1}^{\ell} c_{i,1} = c_1 \pmod p$. Next, the adversary finishes each signing session i to sign the message m_{j_i}, obtaining a valid partial signature from the honest signer $z_{i,1} = r_{i,1} + c_{i,1}x_1$, and calculates $z_1 = \sum_{i=1}^{\ell} z_{i,1}$ where summation is mod p. They also calculate $z_2 = \sum_{i=1}^{\ell} r_{i,2} + x_2 c_2$. Now, the adversary outputs $\sigma = (R, z_1 + z_2)$ where the summation is mod p, which we claim is a valid multi-signature for the message m.

<u>Validity of forged signature.</u> We wish to show that $\sigma = (R, z_1 + z_2)$ is a valid multi-signature for the message m under the group of signers S_1 and S_2. Thus, we must show that

$$g^{z_1+z_2} = R \cdot X_1^{H_{\mathsf{sign}}(X_1,R,X_1,X_2,m)} \cdot X_2^{H_{\mathsf{sign}}(X_2,R,X_1,X_2,m)}$$

Starting from the left hand side, we have that

$$\begin{aligned} g^{z_1+z_2} &= g^{\sum_{i=1}^{\ell} z_{i,1} + z_2} \\ &= g^{\sum_{i=1}^{\ell}(r_{i,1}+c_{i,1}x_1)+z_2} \\ &= g^{\sum_{i=1}^{\ell} r_{i,1} + x_1 \sum_{i=1}^{\ell} c_{i,1} + z_2} \end{aligned}$$

But by construction $\sum_{i=1}^{\ell} c_{i,1} = c_1$ and thus

$$\begin{aligned}
&= g^{\sum_{i=1}^{\ell} r_{i,1} + x_1 c_1 + z_2} \\
&= g^{\sum_{i=1}^{\ell} r_{i,1} + x_1 c_1 + \sum_{i=1}^{\ell} r_{i,2} + x_2 c_2} \\
&= g^{\sum_{i=1}^{\ell} r_{i,1} + r_{i,2}} \cdot g^{x_1 c_1} \cdot g^{x_2 c_2} \\
&= R \cdot X_1^{H_{\mathsf{sign}}(X_1, R, X_1, X_2, m)} \cdot X_2^{H_{\mathsf{sign}}(X_2, R, X_1, X_2, m)}
\end{aligned}$$

which is what we wanted to prove.

5.2 Polynomial Time Attack

In this attack, the adversary opens ℓ signing sessions for some $\ell \geq \lceil \log_2(p) \rceil$, observes the execution of the first two interactive signing rounds in all ℓ sessions, and then picks which message will be signed in each session. Such an attack, we claim, allows the adversary to forge a signature for an arbitrary message of their choice, breaking the unforgeability of the insecure MuSig and the BN-scheme with delayed message selection. In the case of MuSig, the adversary only needs to control the execution of the protocol and pick the messages to be signed, whereas to break the BN-Scheme the adversary needs to corrupt all but one of the signers.

The algorithm for the attack is similar to that presented in [4] against the original two-round variant of MuSig, and is similar to and a little more complicated than the attack in section 5.1. The construction uses the newly discovered algorithm for solving the ROS problem from [4].

The Attack Against MuSig

Let S_1 and S_2 be the signers with private keys x_1 and x_2 and public key X_1 and X_2 respectively. Let $\tilde{X} = X_1^{H_{\text{agg}}(1, X_1, X_2)} \cdot X_2^{H_{\text{agg}}(2, X_1, X_2)}$ denote the aggregate verification key. Let $\ell \geq \lceil \log_2(p) \rceil$ be an integer, let $m_{\ell+1}$ be some message for which the adversary wishes to forge a signature, and for each $j \in \{1, \ldots, \ell\}$ choose distinct messages m_j^0 and m_j^1 that the signers would be willing to sign.

Now, the adversary begins ℓ signing sessions and observes the first two signing rounds to obtain an aggregate nonce $R_j = R_{j,1} \cdot R_{j,2}$ for each $j \in \{1, \ldots, \ell\}$. Then the adversary calculates challenges $c_j^b \leftarrow H_{\mathsf{sign}}(R_j, \tilde{X}, m_j^b)$ for each $j \in \{1, \ldots, \ell\}$ and $b \in \{0, 1\}$. Now, define the group homomorphisms $\rho_+ : (\mathbb{Z}_p)^\ell \to \mathbb{Z}_p$ and $\rho_\times : (\mathbb{G})^\ell \to \mathbb{G}$ as follows:

$$\rho_+(x_1, \ldots, x_\ell) = \sum_{j=1}^{\ell} \frac{2^{j-1} x_j}{c_j^1 - c_j^0}$$

$$\rho_\times(g_1, \ldots, g_\ell) = \prod_{j=1}^{\ell} g_j^{\frac{2^{j-1}}{c_j^1 - c_j^0}}$$

Let $R_{\ell+1} \leftarrow \rho_\times(R_1, \ldots, R_\ell)$ and let $c_{\ell+1} \leftarrow H_{\mathsf{sign}}(R_{\ell+1}, \tilde{X}, m_{\ell+1})$. Let $d \leftarrow c_{\ell+1} - \rho_+(c_1^0, \ldots, c_\ell^0)$ and write it in binary as $\sum_{j=1}^{\ell} 2^{j-1} b_j$ for some $b_1, \ldots, b_\ell \in \{0, 1\}$, which is possible since $\ell \geq \lceil \log_2(p) \rceil$. Now, for each $j \in \{1, \ldots, \ell\}$ complete the signing session j with the message $m_j^{b_j}$ to obtain a valid multi-signature (R_j, z_j). We claim that $\sigma \leftarrow (R_{\ell+1}, \rho_+(z_1, \ldots, z_\ell))$ is a valid multi-signature for the message $m_{\ell+1}$ under the aggregate verification key $\tilde{X}$, and is thus a forgery that breaks the unforgeability of the scheme.

<u>Validity of forged signature.</u> We wish to verify that $\sigma = (R^{\ell+1}, \rho_+(z_1, \ldots, z_\ell))$ is a valid

signature for $m_{\ell+1}$ under the aggregate verification key $\tilde{X}$. Thus, we must show that

$$g^{\rho_+(z_1,\ldots,z_\ell)} = R_{\ell+1}\tilde{X}^{c_{\ell+1}}$$

Note that (R_j, z_j) is a valid schnorr signature for the message $m_j^{b_j}$ under the verification key $\tilde{X}$ for each j, and thus $g^{z_j} = R_j\tilde{X}^{c_j^{b_j}}$. Hence,

$$\rho_\times(g^{z_1},\ldots,g^{z_\ell}) = \rho_\times(R_1\tilde{X}^{c_1^{b_1}},\ldots,R_\ell\tilde{X}^{c_\ell^{b_\ell}})$$

or equivalently,

$$g^{\rho_+(z_1,\ldots,z_\ell)} = \rho_\times(R_1,\ldots,R_\ell)\cdot\tilde{X}^{\rho_+(c_1^{b_1},\ldots,c_\ell^{b_\ell})}$$

By construction $R_{\ell+1} = \rho_\times(R_1,\ldots,R_\ell)$ and lemma 5.1 shows that $\rho_+(c_1^{b_1},\ldots,c_\ell^{b_\ell}) = c_{\ell+1}$. Hence,

$$g^{\rho_+(z_1,\ldots,z_\ell)} = R_{\ell+1}\cdot\tilde{X}^{c_{\ell+1}}$$

which is what we wanted to prove.

Lemma 5.1 *By the construction above,* $c_{\ell+1} = \sum_{j=1}^{\ell}\frac{2^{j-1}c_j^{b_j}}{c_j^1-c_j^0}$.

This lemma is at the heart of the attack, and is precisely the idea that allows [4] to solve the ROS problem.

Proof of Lemma 5.1: By definition $\sum_{j=1}^{\ell}2^{j-1}b_j = c_{\ell-1} - \rho_+(c_1^0,\ldots,c_\ell^0)$. Hence, to prove the lemma it is sufficient to show that $\sum_{j=1}^{\ell}\frac{2^{j-1}c_j^{b_j}}{c_j^1-c_j^0} - \rho_+(\bar{c}_1^0,\ldots,c_\ell^0) = \sum_{j=1}^{\ell}2^{j-1}b_j$.

Starting from the left hand side, we have that

$$\begin{aligned}\sum_{j=1}^{\ell}\frac{2^{j-1}c_j^{b_j}}{c_j^1-c_j^0} - \rho_+(c_1^0,\ldots,c_\ell^0) &= \sum_{j=1}^{\ell}\frac{2^{j-1}c_j^{b_j}}{c_j^1-c_j^0} - \sum_{j=1}^{\ell}\frac{2^{j-1}c_j^0}{c_j^1-c_j^0}\\ &= \sum_{i=1}^{\ell}\frac{2^{j-1}(c_j^{b_j}-c_j^0)}{c_j^1-c_j^0}\end{aligned}$$

However, for each j it holds that $\frac{2^{j-1}(c_j^{b_j}-c_j^0)}{c_j^1-c_j^0}$ is 0 whenever b_j is 0 and is 2^{j-1} whenever b_j is 1. Hence, for each j it holds that $\frac{2^{j-1}(c_j^{b_j}-c_j^0)}{c_j^1-c_j^0} = 2^{j-1}b_j$ and thus

$$\sum_{j=1}^{\ell}\frac{2^{j-1}c_j^{b_j}}{c_j^1-c_j^0} - \rho_+(c_1^0,\ldots,c_\ell^0) = \sum_{j=1}^{\ell}2^{j-1}b_j$$

which is what we wanted to prove. ▮

The Attack Against BN Multi-Signatures

As before, let S_1 and S_2 be the signers with private keys x_1 and x_2 and public keys X_1 and X_2 respectively. Also suppose that S_2 is corrupt and thus their secret key x_2 is known to the adversary. Let $\ell \geq \lceil\log_2(p)\rceil$ be an integer, let $m_{\ell+1}$ be some message for which the adversary wishes to forge a multi-signature, and for each $j \in \{1,\ldots,\ell\}$ choose distinct messages m_j^0 and m_j^1 that the honest signer would be willing to sign.

Now, the adversary begins ℓ signing sessions and observes the first two signing rounds to obtain nonce shares $R_{j,1}$ and $R_{j,2} = g^{r_{j,2}}$ and an aggregate nonce $R_j = R_{j,1} \cdot R_{j,2}$ for each $j \in \{1, \ldots, \ell\}$. Then, the adversary calculates challenges $c_{j,1}^b \leftarrow H_{\mathsf{sign}}(X_1, R_j, X_1, X_2, m_j^b)$ for each $j \in \{1, \ldots, \ell\}$ and $b \in \{0, 1\}$. Now, define the group homomorphisms $\rho_+ : (\mathbb{Z}_p)^\ell \to \mathbb{Z}_p$ and $\rho_\times : (\mathbb{G})^\ell \to \mathbb{G}$ as follows:

$$\rho_+(x_1, \ldots, x_\ell) = \sum_{j=1}^{\ell} \frac{2^{j-1} x_j}{c_{j,1}^1 - c_{j,1}^0}$$

$$\rho_\times(g_1, \ldots, g_\ell) = \prod_{j=1}^{\ell} g_j^{\frac{2^{j-1}}{c_{j,1}^1 - c_{j,1}^0}}$$

Let $R_{\ell+1} \leftarrow \rho_\times(R_1, \ldots, R_\ell)$, let $c_{\ell+1,1} \leftarrow H_{\mathsf{sign}}(X_1, R_{\ell+1}, X_1, X_2, m)$, and also let $c_{\ell+1,2} \leftarrow H_{\mathsf{sign}}(X_2, R_{\ell+1}, X_1, X_2, m)$. Let $d \leftarrow c_{\ell+1,1} - \rho_+(c_{1,1}^0, \ldots, c_{\ell,1}^0)$ and write it in binary as $\sum_{j=1}^{\ell} 2^{j-1} b_j$, which is possible since $\ell \geq \lceil \log_2(p) \rceil$.

Now, for each $j \in \{1, \ldots, \ell\}$ complete the third signing round with the message $m_j^{b_j}$ to obtain a signature share $z_{j,1} = r_{j,1} + c_{j,1}^{b_j} \cdot x_1$. Now, they can calculate $z_{\ell+1,1} \leftarrow \rho_+(z_{1,1}, \ldots, z_{\ell,1})$. Additionally, the adversary can calculate $z_{\ell+1,2} = \rho_+(r_{1,2}, \ldots, r_{\ell,2}) + x_2 \cdot c_{\ell+1,2}$.

We claim that $\sigma \leftarrow (R_{\ell+1}, z_{\ell+1,1} + z_{\ell+1,2})$ is a valid multi-signature for the message $m_{\ell+1}$ under the group S_1 and S_2, and thus this attack breaks the existential unforgeability of the scheme.

<u>Validity of forged signature.</u> We wish to verify that $\sigma = (R_{\ell+1}, z_{\ell+1,1} + z_{\ell+1,2})$ is a valid multi-signature for the message $m_{\ell+1}$ and the group of signers S_1 and S_2. Hence, we must show that

$$g^{z_{\ell+1,1} + z_{\ell+1,2}} = R_{\ell+1} \cdot X_1^{c_{\ell+1,1}} \cdot X_2^{c_{\ell+1,2}}$$

Starting from the left hand side we have that

$$\begin{aligned}
g^{z_{\ell+1,1} + z_{\ell+1,2}} &= g^{\rho_+(z_{1,1}, \ldots, z_{\ell,1}) + \rho_+(r_{1,2}, \ldots, r_{\ell,2}) + x_2 \cdot c_{\ell+1,2}} \\
&= g^{\rho_+(r_{1,1}, \ldots, r_{\ell,1}) + x_1 \cdot \rho_+(c_{1,1}^{b_1}, \ldots, c_{\ell,1}^{b_\ell}) + \rho_+(r_{1,2}, \ldots, r_{\ell,2}) + x_2 \cdot c_{\ell+1,2}} \\
&= g^{\rho_+(r_{1,1} + r_{1,2}, \ldots, r_{\ell,1} + r_{\ell,2})} \cdot X_1^{\rho_+(c_{1,1}^{b_1}, \ldots, c_{\ell,1}^{b_\ell})} \cdot X_2^{c_{\ell+1,2}} \\
&= \rho_\times(R_1, \ldots, R_\ell) \cdot X_1^{\rho_+(c_{1,1}^{b_1}, \ldots, c_{\ell,1}^{b_\ell})} \cdot X_2^{c_{\ell+1,2}} \\
&= R_{\ell+1} \cdot X_1^{\rho_+(c_{1,1}^{b_1}, \ldots, c_{\ell,1}^{b_\ell})} \cdot X_2^{c_{\ell+1,2}}
\end{aligned}$$

By a lemma nearly identical to lemma 5.1, we have that $\rho_+(c_{1,1}^{b_1}, \ldots, c_{\ell,1}^{b_\ell}) = c_{\ell+1,1}$ and therefore

$$= R_{\ell+1} \cdot X_1^{c_{\ell+1,1}} \cdot X_2^{c_{\ell+1,2}}$$

which is what we wanted to prove.

Acknowledgments

I thank Stefano Tessaro for the discussions leading to this note, as well as his invaluable assistance in its writing and revision.

References

[1] M. Bellare and W. Dai. Chain reductions for multi-signatures and the HBMS scheme. In M. Tibouchi and H. Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 650–678, Singapore, Dec. 6–10, 2021. Springer, Heidelberg, Germany. 1, 2, 3, 5, 7

[2] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *ACM CCS 2006: 13th Conference on Computer and Communications Security*, pages 390–399, Alexandria, Virginia, USA, Oct. 30 – Nov. 3, 2006. ACM Press. 1, 2, 3, 7

[3] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany. 3

[4] F. Benhamouda, T. Lepoint, J. Loss, M. Orrù, and M. Raykova. On the (in)security of ROS. In A. Canteaut and F.-X. Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 33–53, Zagreb, Croatia, Oct. 17–21, 2021. Springer, Heidelberg, Germany. 1, 2, 6, 9, 11, 12

[5] D. Boneh, M. Drijvers, and G. Neven. Compact multi-signatures for smaller blockchains. In T. Peyrin and S. Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 435–464, Brisbane, Queensland, Australia, Dec. 2–6, 2018. Springer, Heidelberg, Germany. 7

[6] 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*, pages 1084–1101, San Francisco, CA, USA, May 19–23, 2019. IEEE Computer Society Press. 6

[7] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille. Simple schnorr multi-signatures with applications to bitcoin. In *Designs, Codes and Cryptography*, 2019. 1, 3, 5, 7

[8] J. Nick. Insecure shortcuts in musig, 2019. https://medium.com/blockstream/insecure-shortcuts-in-musig-2ad0d38a97da. 2, 9

[9] J. Nick, T. Ruffing, and Y. Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In T. Malkin and C. Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 189–221, Virtual Event, Aug. 16–20, 2021. Springer, Heidelberg, Germany. 1, 3, 7

[10] S. Tessaro and C. Zhu. Threshold and multi-signature schemes from linear hash functions. In *Advances in Cryptology – EUROCRYPT 2023*, Lyon, France, Apr. 23–27, 2023. 7

[11] D. Wagner. A generalized birthday problem. In M. Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303, Santa Barbara, CA, USA, Aug. 18–22, 2002. Springer, Heidelberg, Germany. 2, 9