M&M'S: Mix and Match Attacks on Schnorr-type Blind Signatures with Repetition

Khue Do, Lucjan Hanzlik, and Eugenio Paracucchi

CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
{`firstname.surname`}`@cispa.de`

Abstract. Blind signatures allow the issuing of signatures on messages chosen by the user so that they ensure *blindness* of the message against the signer. Moreover, a malicious user cannot output $\ell+1$ signatures while only finishing ℓ signing sessions. This notion, called *one-more* unforgeability, comes in two flavors supporting either *sequential* or *concurrent* sessions. In this paper, we investigate the security of a class of blind signatures constructed from Sigma-protocols with small challenge space $\mathcal{C}_\Sigma$ (i.e., polynomial in the security parameter), using k repetitions of the protocol to decrease the chances of a cheating prover. This class of schemes includes, among others, the Schnorr blind signature scheme with bit challenges and the recently proposed isogeny-based scheme CSI-Otter (Crypto'23), as well as potential blind signatures designed from assumptions with the well-known Sigma-protocol for the graph-isomorphism problem (e.g., Lattice Isomorphism Problem).
For this class of blind signatures, we show a *polynomial-time* attack that breaks one-more unforgeability for any $\ell \geq k$ concurrent sessions in time $O(k \cdot |\mathcal{C}_\Sigma|)$. Contrary to the ROS attack, ours is generic and does not require any particular algebraic structure. We also propose a computational trade-off, where, for any $t \leq k$, our attack works for $\ell = \frac{k}{t}$ in time $O(\frac{k}{t} \cdot |\mathcal{C}_\Sigma|^t)$. The consequences of our attack are as follows. Schemes in the investigated class of blind signatures should not be used concurrently without applying specific transformations to boost the security to support more signing sessions. Moreover, for the parameters proposed for CSI-Otter ($k = 128$ and $|\mathcal{C}_\Sigma| = 2$), the scheme becomes forgeable after 128 concurrent signing sessions for the basic attack and with only eight sessions in our optimized attack. We also show that for those parameters, it is even possible to compute two signatures in around 10 minutes with just one signing session using the computation power of the Bitcoin network. Thus, we show that, for sequential security, the parameter k must be at least doubled in the security parameter for any of the investigated schemes.

Keywords: Blind Signatures · Sigma Protocols · Group Actions · Cryptanalysis.

1 Introduction

Blind signatures allow a signer to sign messages picked by third parties without actually seeing them. This property called *blindness* enables many privacypreserving applications of blind signatures. The authenticity of the signer's signatures is based on a *one-more unforgeability* property, which informally states that one cannot create $\ell + 1$ valid signatures under different messages while only finishing the signing process ℓ times with the signer. If one-more unforgeability holds even if those ℓ signing queries are executed in parallel, we are talking about a *concurrently* secure blind signature scheme. Alternatively, we are talking about *sequential* unforgeable blind signatures, if the signer processes queries sequentially. Secure blind signatures can be constructed from different assumptions such as RSA [Cha82, BNPS03], discrete logarithm [PS00, AO00, FPS20, KLX22], pairings [Bol03], lattices [LNP22, HKLN20], and isogenies [KLLQ23]. The resulting schemes provide various properties, including optimal round-complexity (i.e., two-move) and security under stronger security notions.

One of the most prominent ways to construct blind signatures is to leverage the existing construction of standard digital signatures from the Sigma protocol (Σ-protocol) and the Fiat-Shamir transformation. Σ-protocols are three-move interactive protocols, where the prover creates a commitment and later answers a challenge from the verifier. If the challenge space is small (e.g., bit challenges), the prover and verifier repeat the protocol k-times in parallel to decrease the chances of a cheating prover. For Σ-protocols, one can use the Fiat-Shamir transformation to turn this protocol into a signature scheme, i.e., compute the verifier's challenge using a random oracle query on the prover's commitment and a message picked by the prover/signer. The transformation can also be applied to Σ-protocols with a small challenge space where all k commitments are queried at once to the random oracle, and the prover's responses are provided *separately* for each of the k protocol instances.

While this transformation works for standard signatures, it fails for blind signatures since the signer can break blindness by inspecting the commitment and the challenge value of the final signature. The signer can do it even if the user is the one querying the random oracle. A folklore approach to get around this problem is introducing a way to randomize the signer's commitment and output of the random oracle that the user can reverse to receive the final signature. Prominent examples using this approach are the Schnorr blind signature scheme, the Abe-Okamoto scheme in the discrete logarithm setting, and the recently proposed CSI-Otter scheme [KLLQ23] based on isogenies.

Unfortunately, the Schnorr blind signature scheme and many others following the above recipe are vulnerable to the so-called ROS attack [BLL$^+$21]. These schemes cannot be concurrently unforgeable for ℓ bigger than polylogarithmic in the security parameter. In practice, the scheme becomes unusable after approximately 8-10 concurrent signing queries, and the only way out is to either use transformations boosting the security to more signatures [KLR21, CAHL$^+$22] or use the scheme sequentially, which limits the number of real-world applications significantly. The attack requires that there exists an algebraic structure in the space of commitments that the attacker can leverage. Such a structure does not exist, e.g., for the CSI-Otter scheme, making it potentially secure against the

ROS attack and allowing for polynomially many signing queries, an open problem left by the authors [KLLQ23].

1.1 Our contribution

This paper investigates the security of blind signatures built using Σ-protocols with a small challenge space $\mathcal{C}_\Sigma$ (non-negligible soundness error $1/|\mathcal{C}_\Sigma|$). For such blind signature schemes, we show a new polynomial-time attack breaking the unforgeability for $\ell = k$, where k is the number of repetitions of the base protocol. Contrary to the ROS attack, we do not require any particular algebraic structure and rely solely on the honest verifier zero-knowledge property of the Σ-protocol and queries to the random oracle (i.e., computing hash digests).

The Attack. We use the fact that, from the user's point of view, the signer is running k instances of the Σ-protocol with challenges from $\mathcal{C}_\Sigma$. In the case of a honest user, there is no problem since we assume that the user will "glue" all k instances in the same random oracle query. However, a malicious user does not need to do it. Assume that $\mathbf{R} = (R_1, \ldots, R_k)$ is the commitment sent in the first step by the signer and $\mathbf{R}' = (R'_1, \ldots, R'_k)$ is the blinded commitment. A honest user can compute the challenge by computing $\mathbf{c}' = (c'_1, \ldots, c'_k) = \mathcal{H}(\mathbf{R}' \| m)$ and then challenges the signer with a blinded version of it. Given the signer's response, the honest user owns a signature under m.

A malicious user can generate a valid signature differently. Thanks to the honest verifier zero-knowledge property of the Σ-protocol, the adversary can simulate one of the k instances of the protocol. As a result of the simulation, it will receive a valid transcript (e^*, d^*, z^*), where $d^* \in \mathcal{C}_\Sigma$. The adversary can now compute the challenge $\mathbf{c}' = (c'_1, \ldots, c'_k) = \mathcal{H}((R'_1, \ldots, R'_{k-1}, e^*) \| m^*)$ for a random message m^* and repeat the process for a different message until $c'_k = d^*$. On average, a malicious user will only have to repeat this computation $|\mathcal{C}_\Sigma|$ times, e.g. two times for bit challenges.

The key observation, now, is that the adversary only needs the $k-1$ first elements of the signer's response to receive a valid signature. It works since the adversary simulated the last instance and knows a proper answer z^* for the challenge $\mathbf{c}'$ it picked. This means that for the protocol instance with commitment R_k, the adversary can arbitrarily choose a challenge and still be able to receive a valid signature. In itself, this is not interesting. However, suppose the adversary executes the same attack over k concurrent sessions. In that case, the malicious user ends up with k instances of the Σ-protocol for which it can arbitrarily pick the challenge while simultaneously still being able to compute k valid signatures. Thus, the adversary can use those instances to create a one-more signature.

To be more explicit, we provide an unblind version of the attack. Suppose $\mathbf{R}_1, \ldots, \mathbf{R}_k$ are the commitment the signer creates for the k concurrent sessions the adversary executes, where $\mathbf{R}_i = (R_{i,1}, \ldots, R_{i,k})$. The adversary uses the above technique to simulate k instances of the Σ-protocol receiving (e_i, d_i, z_i). It then finds messages $m_1, \ldots, m_k$ such that for every $i \in \{1, \ldots, k\}$ we have

$$\mathcal{H}((R_{i,1}, \ldots, R_{i,k-1}, e_i) \| m_i) = (c_{i,1}, \ldots, c_{i,k-1}, d_i).$$

The adversary now picks a random message m^* and computes the challenge $\mathbf{c}^* = \mathcal{H}((R_{1,k}, \ldots, R_{k,k}) \| m^*) \in \mathcal{C}_\Sigma^k$, where $\mathbf{c}^* = (c_1^*, \ldots, c_k^*)$. Now, for the ith session, it challenges the signer with $(c_{i,1}, \ldots, c_{i,k-1}, c_i^*)$ and receives k separate responses for each of the elements of the challenge. It is easy to see now that by replacing the last entries of the response with the z_i values, the adversary will receive valid signatures for messages $m_1, \ldots, m_k$. By combining those last entries, the adversary can construct a valid signature for challenge $\mathbf{c}^*$, which means a valid forgery for message m^*.

In expectation, the adversary will have to compute the hash function $k \cdot |\mathcal{C}_\Sigma|$ times and execute k concurrent sessions for the attack to be successful. It can also be performed with fewer sessions at the expense of computation, i.e. $\frac{k}{t}$ concurrent sessions with adversary's running time $O(\frac{k}{t} \cdot |\mathcal{C}_\Sigma|^t)$. In case we want to run the attack in the sequential settings, the running time will be $O(2 \cdot |\mathcal{C}_\Sigma|^{k/2})$, i.e., exponential in k but breaking unforgeability by creating two signatures while only querying one signature. In the above attack, the adversary requires the signer to create k (respectively $\frac{k}{t}$) signatures, where all must be made using concurrent sessions. We fix this shortcoming by extending the attack so that the adversary only needs two open sessions at a given time. However, the attack still requires the signer to create k (respectively $\frac{k}{t}$) signatures.

Implications. Our results show that for practical and concurrent applications, one should not use blind signatures constructed from Σ-protocols with a small challenge space. We summarize those protocols in table 1. Since the attack is generic and requires no unique algebraic structure, it works in various settings: Schnorr blind signature with bit challenges in the discrete logarithm setting, the Fiat-Shamir blind signatures (as in [HKL19]) in the RSA setting, CSI-Otter [KLLQ23] in the isogenies setting. Interestingly, the attack also works in the weaker synchronized parallel attack model from [Poi98], where we assume that the steps of the protocol for each session are executed in order of opened sessions, i.e., in our case, this means that the challenge for the first session must be sent before the challenge for the second session, etc.

While increasing the challenge space and consequently the number of instances k makes our attack more time-consuming, an efficient way to forge a signature will always exist if the signer uses the signing keys more than k times. In practice, to use such blind signatures concurrently, one must use boosting transformations similar to the case of Schnorr blind signatures and the ROS attack. Unfortunately, this leads to a significant increase in the communication complexity and signature size.

One of the main downsides of our results is that the attack targets all known Σ-protocols for the graph isomorphism problem. This significantly influences the area of post-quantum blind signatures, where we are still trying to find more efficient alternatives for schemes based on standard lattice-based problems. Potential blind signatures from the lattice isomorphism problem [DvW22] and the CSI-Otter scheme are vulnerable to our attack, leaving us only with relatively inefficient blind signatures from standard lattice assumptions that suffer from big signature sizes.

Σ-protocol	**Hardness Assumption**	**Challenge space**
[HKL19]	Factoring assumption	{0,1}
[BKV19], [DG19]	Isogeny-based assumption	{0,1}
[DvW22]	Lattice Isomorphism assumption	{0,1}
[LNSW13]	Lattice assumption	{0,1, 2}

Table 1. Σ-protocols with small challenge space.

Case study. We introduce our attack, targetting any blind signature scheme built from a Σ-protocol with small challenge space. To make the attack more explicit, we show how to apply it against the CSI-Otter isogeny-based blind signature scheme recently introduced at Crypto'23. With this, we solve an open problem left by the authors [KLLQ23]. In particular, we show that CSI-Otter is not concurrently secure in the polynomial regime, i.e., it does not support polynomially many concurrent sessions as conjectured by the authors.

We show that for the parameters proposed by the authors (i.e., $n = 128$), CSI-Otter becomes forgeable after issuing only 128 signatures. Sacrificing a bit of computation, we can make the scheme insecure even with only eight concurrent sessions. In such a case, the adversary must perform around half a megahash, which can be done in less than a second on commodity hardware (e.g., the rate of the M1 Pro processor is 5Mh/s). Moreover, we argue that CSI-Otter is not even sequentially secure for the proposed parameters. In such a case, our attack requires around $2 \cdot 2^{64}$ hash computations, which is in the realms of the bitcoin difficulty, i.e., can be computed by the bitcoin network in around 10 minutes. Thus, the challenge space needs to be increased for CSI-Otter to be practically secure in the sequential setting. In particular, for n-bit security, we need at least $k = 2 \cdot n$.

Summary of Contribution. This paper analyzes the security of a specific class of blind signature schemes, i.e., schemes constructed by applying a particular transformation to a Σ-protocol with a challenge space that is polynomial in the size of the security parameter. This class captures many existing schemes, including blind Schnorr signatures with bit challenges, the Fiat-Shamir factoring-based scheme (as in [HKL19]), and the recently proposed CSI-Otter isogeny-based scheme [KLLQ23].

– We show that this class of blind signatures is vulnerable to a *polynomial-time* attack that breaks the one-more unforgeability for any $\ell > k$ of concurrent sessions, where the challenge space is $\mathcal{C}_{\Sigma}^{k}$, e.g., $|\mathcal{C}_{\Sigma}| = 2$ and $k = 128$ for CSI-Otter [KLLQ23]. The attack is generic and does not require any particular algebraic structure. It can be improved so that, at a given time, the adversary

only needs to keep two sessions open concurrently while still requiring k signatures to create a forged one.
- We then show that there exists a tradeoff between the number of sessions/signatures needed and the adversary's running time. The implications are that CSI-Otter is forgeable for the above parameters with only eight concurrent sessions in under a second on commodity hardware.
- We show that even if such schemes are only used in the impractical sequential setting, their parameters must be considered carefully. In particular, the challenge space must be at least doubled in the security parameter $k \geq 2 \cdot n$. Otherwise, for the parameters proposed in [KLLQ23], the required computation needed by an adversary is comparable to the work required by Bitcoin miners (2^{66} hash operations in expectation), where the network of miners can solve such a problem in around ten minutes.

2 Background

2.1 Notation

For a set X, $|X|$ denotes the cardinality of X. For convenience, vectors will be denoted by a bold letter $\mathbf{v} = (v_1, \ldots, v_k)$. We will denote the security parameter with n, and for positive integers k we define $[k] := \{1, \ldots, k\}$. We will use $\|$ to denote the concatenation of strings. We use uppercase letters $\mathcal{A}$ or A to denote algorithms. We let $y \leftarrow \mathcal{A}(x)$ denote the output of $\mathcal{A}$ on input x and $y \leftarrow\$ \mathcal{A}(x)$ for the randomized algorithm. For a set X the notation $x \leftarrow\$ X$ means that x is uniformly sampled from X. We use st explicitly to denote the inner state of an algorithm while omitting it when the context is clear. We say that a function is negligible and denote it as $\mathsf{negl}(n)$ if it vanishes faster than any polynomial. For a prime number p, we denote the finite field with p elements as $\mathbb{F}_p$.

2.2 Sigma Protocols

Definition 1 (Σ-protocol). *A sigma protocol (Σ-protocol) is a three-move interactive protocol $\Sigma = (\mathcal{G}, \mathcal{P}, \mathcal{V}, \mathcal{C})$ that consists of the following p.p.t. algorithm:*

Key generation *On input the security parameter 1^n, the probabilistic algorithm $\mathcal{G}$ outputs a public key pk and a secret key sk.*

Prover *The **prover** $\mathcal{P} = (\mathsf{P}_1, \mathsf{P}_2)$ consists of two algorithms:*

1. $(R, \mathsf{st}) \leftarrow\$ \mathsf{P}_1(\mathsf{sk})$*: on input a secret key sk the probabilistic algorithm P_1 outputs a **commitment** R and an internal state st.*
2. $s \leftarrow \mathsf{P}_2(\mathsf{sk}, c, \mathsf{st})$*: on input a **challenge** $c \in \mathcal{C}$, a secret key sk and and a state st, the algorithm P_2 outputs a **response** s.*

Verifier *On input a public key pk, a commitment R, a challenge c, and a response s, the verification algorithm $\mathcal{V}$ outputs 1 to indicate the prover is valid, and 0 otherwise.*

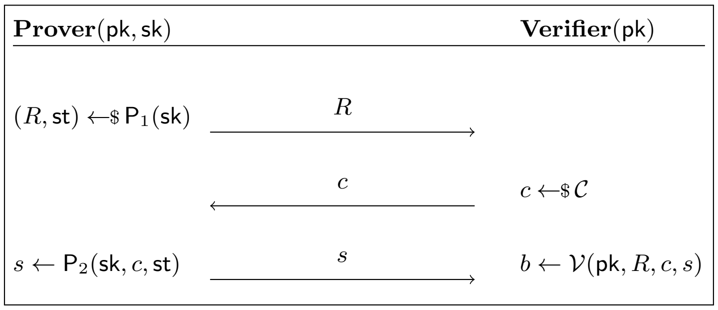

$\mathbf{Prover}(\mathsf{pk}, \mathsf{sk})$
$\mathbf{Verifier}(\mathsf{pk})$
$(R, \mathsf{st}) \leftarrow\!\$\, \mathsf{P}_1(\mathsf{sk})$
R
c
$c \leftarrow\!\$\, \mathcal{C}$
$s \leftarrow \mathsf{P}_2(\mathsf{sk}, c, \mathsf{st})$
s
$b \leftarrow \mathcal{V}(\mathsf{pk}, R, c, s)$

Fig. 1. Interaction in a sigma protocol.

We say that a Σ-protocol is correct if for every 1^n, for every key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}(1^n)$, and for every transcript (R, c, s) output from the interaction in figure 1 between a prover and a verifier, we have $\mathcal{V}(\mathsf{pk}, R, c, s) = 1$.

Security notions. We briefly capture the security of a Σ-protocol in the following notions: *honest verifier zero-knowledge* and *special soundness.*

- We say that a Σ-protocol is **honest verifier zero-knowledge (HVZK)** if there exists a p.p.t. simulator Sim such that for every key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}(1^n)$, and for every challenge $c \in \mathcal{C}$, it outputs a valid transcript $(R, c, s) \leftarrow \mathsf{Sim}(\mathsf{pk}, c)$ that is indistinguishable from a real transcript.
- We say that a Σ-protocol is **(2-)special sound** if there exists a deterministic polynomial time extractor Ext such that, for every key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}(1^n)$, recovers the secret sk given two valid transcripts (R, c_1, s_1) and (R, c_2, s_2) sharing the same commitment and different challenges.

We will assume hereafter that any Σ-protocol we encounter satisfies the properties enunciated in the above security notions. Recall also that we can turn any Σ-protocols into a signature scheme using the Fiat-Shamir Transform [FS87, PS00].

2.3 Blind Signature Schemes

Definition 2 (Three-Moves Blind Signature Scheme). *A three-moves blind signature scheme $\mathsf{BS} = (\mathcal{G}, \mathcal{S}, \mathcal{U}, \mathcal{V})$ consists of the following p.p.t. algorithms:*

Key generation. *On input the security parameter 1^n, the probabilistic algorithm $\mathcal{G}$ outputs a public key pk and a secret key sk.*

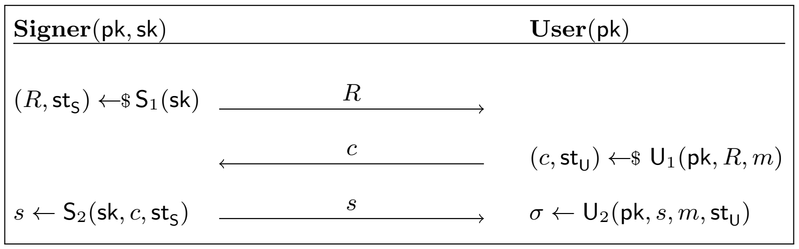
Signer. *The* ***signer*** *$\mathcal{S} = (\mathsf{S}_1, \mathsf{S}_2)$ consists of two algorithms:*

1. *$(R, \mathsf{st}_\mathsf{S}) \leftarrow\!\$\, \mathsf{S}_1(\mathsf{sk})$: on input a secret key sk, the probabilistic algorithm S_1 outputs an internal state st_S and a commitment R.*
2. *$s \leftarrow \mathsf{S}_2(\mathsf{sk}, c, \mathsf{st}_\mathsf{S})$: on input a challenge $c \in \mathcal{C}$, a secret key sk and an internal state st, the algorithm S_2 outputs a response s.*

User. *The* ***user*** *$\mathcal{U} = (\mathsf{U}_1, \mathsf{U}_2)$ consists of two algorithms:*

1. $(c, \mathsf{st}_\mathsf{U}) \leftarrow\$ \mathsf{U}_1(\mathsf{pk}, m, R)$*: on input a public key* pk*, a message* m *and a signer commitment* R*, the probabilistic algorithm* U_1 *outputs a challenge* $c \in \mathcal{C}$ *and an internal state* st.
2. $\sigma \leftarrow \mathsf{U}_2(\mathsf{pk}, s, \mathsf{st}_\mathsf{U})$*: on input a public key* pk*, a signer response* s *and an internal state* st_U*, the probabilistic algorithm* U_2 *outputs a signature* σ *on a message* m.

Verification. *On input a public key* pk*, a message* m*, and a signature* σ*, the verification algorithm* $\mathcal{V}$ *outputs* 1 *to indicate the prover is valid, and* 0 *otherwise.*



Signer$(\mathsf{pk}, \mathsf{sk})$
User(pk)
$(R, \mathsf{st}_\mathsf{S}) \leftarrow\$ \mathsf{S}_1(\mathsf{sk})$
R
c
$(c, \mathsf{st}_\mathsf{U}) \leftarrow\$ \mathsf{U}_1(\mathsf{pk}, R, m)$
$s \leftarrow \mathsf{S}_2(\mathsf{sk}, c, \mathsf{st}_\mathsf{S})$
s
$\sigma \leftarrow \mathsf{U}_2(\mathsf{pk}, s, m, \mathsf{st}_\mathsf{U})$

Fig. 2. Interaction in a Three-move Blind Signature Scheme.

We say that a three-moves blind signature scheme BS *is correct if for every* 1^n*, for every key pair* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}(1^n)$*, for every message and signature pair* (m, σ) *generated from the interaction in figure 2 between an user and a signer, we have* $\mathcal{V}(\mathsf{pk}, m, \sigma) = 1$*. We will omit "three-moves" and will say blind signature for brevity.*

Security notions. We capture the security of a Blind Signature Scheme BS by two security notions: *blindness* and *one-more unforgeability*. For blindness we will omit its formal definition and refer to [HKL19] since we do not discuss blindness of the schemes and it is out of this paper's scope.

Definition 3 (One-more-unforgeability). *Let* $\mathsf{BS} = (\mathcal{G}, \mathcal{U}, \mathcal{S}, \mathcal{V})$ *be a blind signature and* n *be the security parameter. We define the* ℓ*-one more unforgeability game* ℓ-$\mathbf{OMUF}_\mathsf{BS}$ *with an adversary* $\mathcal{A}$ *(in the role of the user) as follows:*

Setup. *Sample a pair of keys* $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathcal{G}(1^n)$*. Initialize* $\ell_{\mathtt{closed}} := 0$ *and run* $\mathcal{A}$ *on input* pk.

Online Phase. $\mathcal{A}$ *is given access to oracles* sign_1 *and* sign_2*, which behave as follows.*

Oracle sign_1**:** *the oracle samples a fresh session identifier* sid*. It sets* $\mathsf{open}_\mathsf{sid} := \mathtt{true}$ *and generates* $(R_\mathsf{sid}, \mathsf{st}_\mathsf{sid}) \leftarrow\$ \mathsf{S}_1(\mathsf{sk})$*. Then it returns the response* R_sid *to* $\mathcal{A}$.

Oracle sign_2**:** *If* $\ell_{\mathtt{closed}} < \ell$*, the oracle takes as input a challenge* c *and a session identifier* sid*. If* $\mathsf{open}_\mathsf{sid} = \mathtt{false}$*, it returns* $\perp$*. Otherwise, it* *sets $\ell_{\texttt{closed}}++$ and $\texttt{open}_{\mathsf{sid}} := \texttt{false}$. Then it computes the response $s \leftarrow\$ \mathsf{S}_2(\mathsf{sk}, \mathsf{st}_{\mathsf{sid}}, c)$ and returns s to $\mathcal{A}$.*

Output Determination. *When $\mathcal{A}$ outputs distinct tuples $(m_1, \sigma_1), \ldots, (m_k, \sigma_k)$, return 1 if $k \geq \ell_{\texttt{closed}} + 1$ and $\mathcal{V}(\mathsf{pk}, \sigma_i, m_i) = 1$ for all $1 \leq i \leq k$. Otherwise, return 0.*

We define the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}_{\mathcal{A}}^{\ell\text{-}\mathbf{OMUF}_{\mathsf{BS}}}(n) = \Pr\left[\ell\text{-}\mathbf{OMUF}_{\mathsf{BS}}{}^{\mathcal{A}} = 1\right],$$

where the probability goes over the randomness of the game as well as the randomness of the adversary $\mathcal{A}$. We say the scheme BS is ℓ-one-more unforgeable if for any adversary $\mathcal{A}$ that makes at most ℓ queries to sign_1, the following holds:

$$\mathsf{Adv}_{\mathcal{A}}^{\ell\text{-}\mathbf{OMUF}_{\mathsf{BS}}}(n) \leq \mathsf{negl}(n).$$

3 Mix-and-Match Attacks

This section presents three attacks against the one-more-unforgeability of blind signature schemes based on Σ-protocols. The first two attacks center on concurrent security, and the third focuses on sequential security. We also briefly demonstrate the first two attacks in figures 3 and 4.

3.1 Schorr-type Blind Signatures

Many blind signature schemes, such as the Schnorr Blind Signature [CP93], the Abe-Okamoto scheme [AO00] or the more recent CSI-Otter [KLLQ23], are based on an underlying sigma protocol. We will call the schemes following this paradigm **Schnorr-type blind signatures**.

Definition 4. *Let $\mathsf{BS} = (\mathcal{G}, \mathcal{S}, \mathcal{U}, \mathcal{V})$ be a blind signature. We say that BS is of Schnorr-type if there exists a sigma protocol $\Sigma = (\mathcal{G}_\Sigma, \mathcal{P}_\Sigma, \mathcal{V}_\Sigma, \mathcal{C}_\Sigma)$ and an hash function $\mathcal{H} : \{0,1\}^\star \to \mathcal{C}_\Sigma$ such that:*

1. *$\mathcal{G} = \mathcal{G}_\Sigma$, and $\mathcal{V}$ is as in the Fiat-Shamir's construction.*
2. *$\mathcal{S} = \mathcal{P}_\Sigma$. So, in particular, we have that $\mathsf{S}_1 = \mathsf{P}_1$ and $\mathsf{S}_2 = \mathsf{P}_2$.*

We also require the following property on the user $\mathcal{U} = (\mathsf{U}_1, \mathsf{U}_2)$:

3. *there exists algorithms $\mathsf{U}_1.\mathsf{BlindCom}$, $\mathsf{U}_1.\mathsf{BlindChal}$ and $\mathsf{U}_2.\mathsf{BlindResp}$ such that:*

```
$\mathsf{U}_1(\mathsf{pk}, m, R)$
1 : $(R', \mathsf{st}_\mathsf{U}) \leftarrow\$ \mathsf{U}_1.\mathsf{BlindCom}(R)$
2 : $c' \leftarrow \mathcal{H}(R' \| m)$
3 : $c \leftarrow \mathsf{U}_1.\mathsf{BlindChal}(c', \mathsf{st}_\mathsf{U})$
4 : **return** $(c, \mathsf{st}_\mathsf{U})$

$\mathsf{U}_2(\mathsf{pk}, s, \mathsf{st}_\mathsf{U})$
1 : $s' \leftarrow \mathsf{U}_2.\mathsf{BlindResp}(s, \mathsf{st}_\mathsf{U}))$
2 : $\sigma \leftarrow (R', s')$
3 : **return** $\sigma$
```

We will denote such a blind signature as BS_Σ.

Like sigma protocols, given a Schnorr-type blind signature BS_Σ with challenge space $\mathcal{C}$, we can construct (by taking parallel repetitions) another blind signature scheme with challenge space $\mathcal{C}^k$ for any $k \in \mathbb{N}$. We will denote it as BS_Σ^k.

3.2 Main attack

In this attack, the adversary initiates k concurrent sessions and obtains $k+1$ valid signatures.

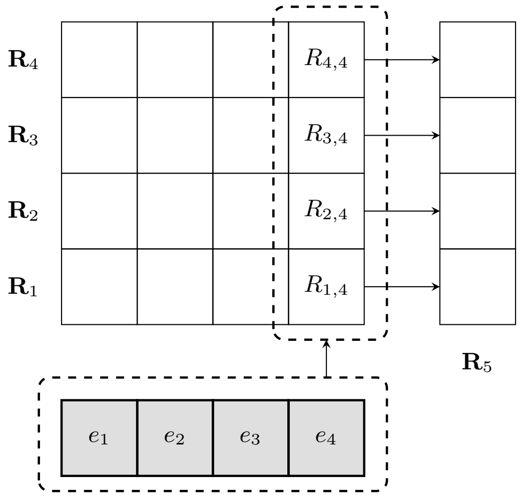


Fig. 3. The main attack depicted for the special case $k = 4$. The rows of the matrix are the signer's commitments $\mathbf{R}_1, \ldots, \mathbf{R}_4$ of the four sessions. The row at the bottom represents the simulated instances of the Σ-protocol used to replace the last instances in the signer's commitment, which are used to generate a forgery with commitment $\mathbf{R}_5$.

Let $\mathsf{BS} = \mathsf{BS}_\Sigma^k = (\mathcal{G}, \mathcal{S}, \mathcal{U}, \mathcal{V})$ be a k-parallel repetition of a Schnorr-type blind signature where k is a positive integer[1], and let $\Sigma = (\mathcal{G}_\Sigma, \mathcal{P}_\Sigma, \mathcal{V}_\Sigma, \mathcal{C}_\Sigma)$ be the underlying sigma protocol. We have, by construction, that the challenge space of BS is equal to $\mathcal{C}_\Sigma^k$ and the signer $\mathcal{S}$ is equal to $\mathcal{P}_\Sigma^k$. The attacker $\mathcal{A}$ will proceed as follows.

1. It opens k concurrent sessions $\mathsf{BS}^{(1)}, \ldots, \mathsf{BS}^{(k)}$ with $\mathcal{S}$. Let $\mathbf{R}_i$ be the commitment sent by $\mathcal{S}$ in the ith session. Since $\mathcal{S} = \mathcal{P}_\Sigma^k$, we can write

$$\mathbf{R}_i = (R_{i,1}, \ldots, R_{i,k})$$

where $(R_{i,j}, \mathsf{st}_{ij}) \leftarrow_\$ \mathsf{P}_1(\mathsf{sk})$ for any i, j.

[1] More precisely, this number is a function of the security parameter n.

2. From that, the attacker constructs a new fake signer commitment as

$$\mathbf{R}_{k+1} = (R_{1,k}, \dots, R_{k,k}).$$

Let $\mathbf{R}'_{k+1} \leftarrow \mathsf{U}_1.\mathsf{BlindCom}(\mathbf{R}_{k+1})$, $\mathbf{c}_{k+1} \leftarrow \mathsf{U}_1.\mathsf{BlindChal}(\mathbf{c}'_{k+1})$ be the blinded commitment and challenge respectively, where $\mathbf{c}'_{k+1} = \mathcal{H}(\mathbf{R}'_{k+1} \| m_{k+1})$ for a message $m_{i+1} \in \{0,1\}^\star$ picked at random.
In addition, it computes k valid transcripts[2] (e_i, d_i, z_i) of the underlying sigma protocol for random challenges d_i.

3. To construct a valid response for the challenge $\mathbf{c}'_{k+1}$, $\mathcal{A}$ interacts with $\mathcal{S}$ as follows. For each opened session $\mathsf{BS}^{(i)}$, the attacker replaces $\mathbf{R}_i$ with $\widetilde{\mathbf{R}}_i = (R_{i,1}, \dots, R_{i,k-1}, e_i)$ and blinds it as

$$\mathbf{R}'_i = (R'_{i,1}, \dots, R'_{i,k-1}, e_i), \tag{1}$$

where, for $j \in [k-1]$, $R'_{i,j} \leftarrow \mathsf{U}_1.\mathsf{BlindCom}(R_{i,j})$. The attacker then finds a message m_i such that the last entry of $\mathbf{c}'_i = \mathcal{H}(\mathbf{R}'_i \| m_i)$ is equal to d_i. For random messages m_i this happens after $O(|\mathcal{C}_\Sigma|)$ queries of the hash function[3]. In this way, it will be able to ask the signer the response for the challenge of the forgery. $\mathcal{A}$ then sends to the signer the blinded challenge

$$\mathbf{c}_i = (c_{i,1}, \dots, c_{i,k-1}, c_{k+1,i}),$$

where $c_{i,j} \leftarrow_\$ \mathsf{U}_1.\mathsf{BlindChal}(c'_{i,j})$, and receives the response $\mathbf{s}_i$ from $\mathcal{S}$, where $\mathbf{s}_i \leftarrow \mathsf{S}_2(\mathsf{sk}, \mathbf{c}_i)$. The attacker now blinds the response as

$$\mathbf{s}'_i = (s'_{i,1}, \dots, s'_{i,k-1}, z_i), \tag{2}$$

where $s'_{i,j} \leftarrow_\$ \mathsf{U}_2.\mathsf{BlindResp}(s_{i,j})$ and close session $\mathsf{BS}^{(i)}$. Finally, the attacker sets the response for $\mathbf{c}'_{k+1}$ to

$$\mathbf{s}'_{k+1} = (s'_{1,k}, \dots, s'_{k,k}). \tag{3}$$

4. To conclude, $\mathcal{A}$ outputs $k+1$ signatures:

$$(m_i, (\mathbf{R}'_i, \mathbf{s}'_i)), \text{ for } i \in [k+1].$$

Lemma 1. *For all $i = 1, \dots, k+1$ the pair $(\mathbf{R}'_i, \mathbf{s}'_i)$ is a valid signature for the message m_i.*

Proof. We distinguish two cases: $i \leq k$ and $i = k+1$.

[2] This can be done because of the honest verifier zero-knowledge property of the sigma protocol.

[3] Define $x = |\mathcal{C}_\Sigma|$, the probability of fail after Q queries is equal to $((x-1)/x)^Q$. This probability is negligible for $Q = O(x)$, with constant $128 \log 2$.

$i \le k$. From equation 1 we have $\mathbf{R}'_i = (R'_{i,1}, \dots, R'_{i,k-1}, e_i)$ and from 2 that $\mathbf{s}'_i = (s'_{i,1}, \dots, s'_{i,k-1}, z_i)$. Let $\mathbf{c}'_i = \mathcal{H}(\mathbf{R}'_i \| m_i)$. For $j \in [k-1]$, $R'_{i,j}$ and $s'_{i,j}$ are honestly generated following BS_Σ, therefore

$$\mathcal{V}_\Sigma(\mathsf{pk}, R'_{i,j}, c'_{i,j}, s'_{i,j}) = 1 \quad \forall j \in [k-1]$$

In addition, by construction, $c'_{i,k} = d_i$ and (e_i, d_i, z_i) is a valid transcript for Σ, hence $\mathcal{V}_\Sigma(\mathsf{pk}, e_i, d_i, z_i) = 1$. Therefore

$$\mathcal{V}_{\mathsf{BS}^k_\Sigma}(\mathsf{pk}, m_i, (\mathbf{R}'_i, \mathbf{s}'_i)) = 1.$$

$i = k+1$. Let $\mathbf{R}'_{k+1} \leftarrow \mathsf{U}_1.\mathsf{BlindCom}(\mathbf{R}_{k+1})$, $\mathbf{c}'_{k+1} = \mathcal{H}(\mathbf{R}'_{k+1} \| m_{k+1})$ and also let $\mathbf{c}_{k+1} \leftarrow \mathsf{U}_1.\mathsf{BlindChal}(\mathbf{c}'_{k+1})$ as in step 2 above. By equation 3 we have $\mathbf{s}'_{k+1} = (s'_{1,k}, \dots, s'_{k,k})$ where $s_{j,k}$ is the blinded response of the signer for the challenge $c_{k+1,j}$ and commitment $R_{j,k}$. Therefore

$$\mathcal{V}_\Sigma(\mathsf{pk}, R'_{k+1,1}, c'_{k+1,j}, s'_{j,k}) = 1 \quad \forall j \in [k-1].$$

and hence $\mathcal{V}_{\mathsf{BS}^k_\Sigma}(\mathsf{pk}, m_{k+1}, (\mathbf{R}'_{k+1}, \mathbf{s}'_{k+1})) = 1$

We have just proved the following:

Theorem 1. *Let $\mathsf{BS} = \mathsf{BS}^k_\Sigma$ be a parallel repetition of a Schnorr-type blind signature scheme supporting at most k concurrent sessions. There exists an adversary $\mathcal{A}$ against the ℓ-one-more unforgeability game, for $\ell = k$, such that*

$$\mathsf{Adv}_{\mathcal{A}}^{\ell\text{-}\mathbf{OMUF}_{\mathsf{BS}}}(n) = 1$$

and asking $O(k \cdot |\mathcal{C}_\Sigma|)$ queries to the hash function, where $\mathcal{C}_\Sigma$ is the challenge space of the base sigma protocol.

As previously mentioned in the introduction, the above attack can be generalized as follows. Instead of brute force one challenge, $\mathcal{A}$ will go for t consecutive challenges for some $1 \le t \le k$. In this way, by opening $s = \lceil k/t \rceil$ concurrent sessions and making $O(|\mathcal{C}_\Sigma|^t)$ queries to the hash function for each of them, it will get $s+1$ signatures.

Theorem 2. *Let $\mathsf{BS} = \mathsf{BS}^k_\Sigma$ be a parallel repetition of a Schnorr-type blind signature scheme supporting at most $s = \lceil k/t \rceil$ concurrent sessions for some $1 \le t \le k$. There exists an adversary $\mathcal{A}$ against the ℓ-one-more unforgeability game, for $\ell = s$, such that,*

$$\mathsf{Adv}_{\mathcal{A}}^{\ell\text{-}\mathbf{OMUF}_{\mathsf{BS}}}(n) = 1$$

and asking $O(s \cdot |\mathcal{C}_\Sigma|^t)$ queries to the hash function, where $\mathcal{C}_\Sigma$ is the challenge space of the base sigma protocol.

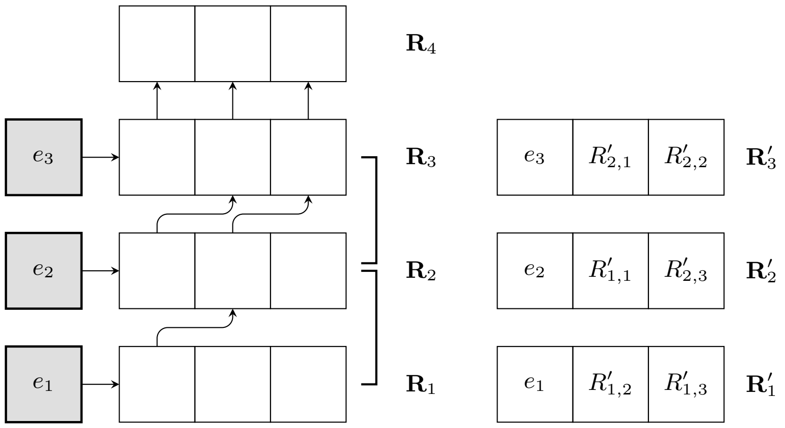


Fig. 4. The 2 out of k attack when $k = 3$. The vectors $\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3$ are the signer commitments. The attacker will replace them with $\mathbf{R}'_1, \mathbf{R}'_2, \mathbf{R}'_3$ in order to get the forgery. On the right side, we depict the commitments used for the 3 standard signatures, while the left side represents the order of the challenges asked to the signer. At the top, we have the forgery with commitment $\mathbf{R}_4$ that in this scenario corresponds to $\mathbf{R}_3$.

3.3 Two out of k attack

In this attack, the adversary employs $k-1$ pairs of concurrent sessions, totaling k sessions, to get $k+1$ signatures. More precisely, we will construct an adversary $\mathcal{A}$ against the k-one-more unforgeability for a Schnorr-type blind signature that supports at most two concurrent sessions at a given time. Let $\mathsf{BS} = \mathsf{BS}^k_\Sigma$ be as in the previous section. For simplicity, we will present the adversary $\mathcal{A}$ in the particular case $k = 3$. However, the attack generalized easily, and we will prove its correctness for an arbitrary k. The attacker proceeds as follows.

1. It first opens two concurrent sessions $\mathsf{BS}^{(1)}, \mathsf{BS}^{(2)}$. Let $\mathbf{R}_1 = (R_{1,1}, R_{1,2}, R_{1,3})$ and $\mathbf{R}_2 = (R_{2,1}, R_{2,2}, R_{2,3})$ be the commitments sent by the signer in the first and second session respectively. Also, let $\widetilde{\mathbf{R}}_1 \leftarrow \mathsf{U}_1.\mathsf{BlindCom}(\mathbf{R}_1)$ and $\widetilde{\mathbf{R}}_2 \leftarrow \mathsf{U}_1.\mathsf{BlindCom}(\mathbf{R}_2)$ be the blinded commitments.
2. $\mathcal{A}$ now computes two valid transcripts (e_1, d_1, z_1) and (e_2, d_2, z_2) of the underlying sigma protocol for random challenges d_1 and d_2. Set $\mathbf{R}'_1 = (e_1, \widetilde{R}_{1,2}, \widetilde{R}_{1,3})$ and $\mathbf{R}'_2 = (e_2, \widetilde{R}_{1,1}, \widetilde{R}_{2,3})$. Then, the attacker searches for two messages m_1, m_2 such that $\mathbf{c}'_1 = \mathcal{H}(\mathbf{R}'_1 \| m_1)$ and $\mathbf{c}'_2 = \mathcal{H}(\mathbf{R}'_2 \| m_2)$ have the first component equal to d_1 and d_2 respectively. As before, this can be achieved with $O(|\mathcal{C}_\Sigma|)$ queries to the hash function.
3. The attacker computes $\widetilde{\mathbf{c}}_1 \leftarrow \mathsf{U}_1.\mathsf{BlindChal}(\mathbf{c}'_1)$ and $\widetilde{\mathbf{c}}_2 \leftarrow \mathsf{U}_1.\mathsf{BlindChal}(\mathbf{c}'_2)$, and sends to the signer, in the first session, the challenge $\mathbf{c}_1 = (\widetilde{c}_{2,2}, \widetilde{c}_{1,2}, \widetilde{c}_{1,3})$. Upon receiving the response $\mathbf{s}_1$ from the signer, $\mathcal{A}$ blinds it by computing $\widetilde{\mathbf{s}}_1 \leftarrow \mathsf{U}_2.\mathsf{BlindResp}(\mathbf{s}_1)$ and sets

$$\mathbf{s}'_1 = (z_1, \widetilde{s}_{1,2}, \widetilde{s}_{1,3}).$$

It outputs $(m_1, (\mathbf{R}'_1, \mathbf{s}'_1))$ and closes $\mathsf{BS}^{(1)}$.

4. Now $\mathcal{A}$ opens the third session $\mathsf{BS}^{(3)}$. Note that two concurrent sessions are open at the moment. Let $\mathbf{R}_3 = (R_{3,1}, R_{3,2}, R_{3,3})$ be the commitment sent by the signer and let $\widetilde{\mathbf{R}}_3 \leftarrow \mathsf{U}_1.\mathsf{BlindCom}(\mathbf{R}_3)$ be the blinded commitment. It creates a valid transcript of the underlying sigma protocol (e_3, d_3, z_3) for a random challenge d_3 and sets $\mathbf{R}'_3 = (e_3, \widetilde{R}_{2,1}, \widetilde{R}_{2,2})$. As before, searches for a message m_3 such that $\mathbf{c}'_3 = \mathcal{H}(\mathbf{R}'_3 \| m_3)$ has the first component equal to d_3.
5. Similarly to Step 3 above, the attacker computes $\widetilde{\mathbf{c}}_3 \leftarrow \mathsf{U}_1.\mathsf{BlindChal}(\mathbf{c}'_3)$ and sends to the signer, in the second session, the challenge $\mathbf{c}_2 = (\widetilde{c}_{3,2}, \widetilde{c}_{3,3}, \widetilde{c}_{2,3})$. Upon receiving the response $\mathbf{s}_2$ from the signer in session two, $\mathcal{A}$ computes the blinded response as $\widetilde{\mathbf{s}}_2 \leftarrow \mathsf{U}_2.\mathsf{BlindResp}(\mathsf{s}_2)$ and sets
$$\mathbf{s}'_2 = (z_2, \widetilde{s}_{1,1}, \widetilde{s}_{2,3}).$$
It outputs $(m_2, (\mathbf{R}'_2, \mathbf{s}'_2))$ and closes $\mathsf{BS}^{(2)}$.
6. Now only one session, $\mathsf{BS}^{(3)}$, is open. It sets $\mathbf{R}'_4 = \widetilde{\mathbf{R}}_3$ and $\mathbf{c}'_4 = \mathcal{H}(\mathbf{R}'_4 \| m_4)$. $\mathcal{A}$ will ask the server in $\mathsf{BS}^{(3)}$ to respond for this challenge. It hence computes $\widetilde{\mathbf{c}}_4 \leftarrow \mathsf{U}_1.\mathsf{BlindChal}(\mathbf{c}'_4)$ and sends to $\mathcal{S}$, in the third session, the challenge $\mathbf{c}_3 = (\widetilde{c}_{4,1}, \widetilde{c}_{4,2}, \widetilde{c}_{4,3})$. Upon receiving the response $\mathbf{s}_3$ from the signer, $\mathcal{A}$ blinds it as $\widetilde{\mathbf{s}}_3$ and sets
$$\mathbf{s}'_3 = (z_3, \widetilde{s}_{2,1}, \widetilde{s}_{2,2}),$$
and
$$\mathbf{s}'_4 = (\widetilde{s}_{3,1}, \widetilde{s}_{3,2}, \widetilde{s}_{3,3})$$
It outputs $(m_3, (\mathbf{R}'_3, \mathbf{s}'_3))$, the forgery $(m_4, (\mathbf{R}'_4, \mathbf{s}'_4))$, and closes $\mathsf{BS}^{(3)}$.

Lemma 2. *For all $i = 1, \ldots, k+1$ the pair $(\mathbf{R}'_i, \mathbf{s}'_i)$ is a valid signature for the message m_i.*

Proof. We will distinguish two cases like in lemma 1.

$i \le k$. We have by construction
$$\mathbf{R}'_i = (e_i, \underbrace{\widetilde{R}_{i-1,1}, \ldots, \widetilde{R}_{i-1,i-1}}_{i-1 \text{ terms}}, \underbrace{\widetilde{R}_{i,i+1}, \ldots, \widetilde{R}_{i,k}}_{k-i \text{ terms}}),$$
$$\mathbf{c}'_i = \mathcal{H}(\mathbf{R}'_i \| m_i) = (c'_{i,1}, \ldots, c'_{i,k}),$$
and
$$\mathbf{s}'_i = (z_i, \underbrace{\widetilde{s}_{i-1,1}, \ldots, \widetilde{s}_{i-1,i-1}}_{i-1 \text{ terms}}, \underbrace{\widetilde{s}_{i,i+1}, \ldots, \widetilde{s}_{i,k}}_{k-i \text{ terms}}).$$
Since $c'_{i,1} = d_i$ by construction, we have $\mathcal{V}_\Sigma(\mathsf{pk}, e_i, d_i, z_i) = 1$. Moreover, for all $1 \le j \le i-1$, we have $\mathcal{V}_\Sigma(\mathsf{pk}, \widetilde{R}_{i-1,j}, c_{i,j}, \widetilde{s}_{i-1,j}) = 1$. Similarly, $\mathcal{V}_\Sigma(\mathsf{pk}, \widetilde{R}_{i,j}, c_{i,j}, \widetilde{s}_{i,j}) = 1$ for all $i+1 \le j \le k$. We therefore have
$$\mathcal{V}_{\mathsf{BS}^k_\Sigma}(\mathsf{pk}, m_i, (\mathbf{R}'_i, \mathbf{s}'_i)) = 1.$$

$i = k + 1$. In this case we have

$$\mathbf{R}'_{k+1} = (\widetilde{R}_{k,1}, \ldots, \widetilde{R}_{k,k}),$$

$$\mathbf{c}'_{k+1} = \mathcal{H}(\mathbf{R}'_i \| m_i) = (c'_{k,1}, \ldots, c'_{k,k})$$

and

$$\mathbf{s}'_{k+1} = (\widetilde{s}_{k,1}, \ldots, \widetilde{s}_{k,k}).$$

From the description of the attack we have $\mathcal{V}_{\Sigma}(\mathsf{pk}, \widetilde{R}_{k,j}, c_{k,j}, \widetilde{s}_{k,j}) = 1$ for any $j \in [k]$ and hence

$$\mathcal{V}_{\mathsf{BS}^k_{\Sigma}}(\mathsf{pk}, m_{k+1}, (\mathbf{R}'_{k+1}, \mathbf{s}'_{k+1})) = 1.$$

We have just proved the following theorem:

Theorem 3. *Let $\mathsf{BS} = \mathsf{BS}^k_{\Sigma}$ be (a parallel repetition of) a Schnorr-type blind signature scheme supporting at most two concurrent sessions. There exists an adversary $\mathcal{A}$ against the ℓ-one-more unforgeability game, for $\ell = k$, such that*

$$\mathsf{Adv}^{\ell\text{-}\mathbf{OMUF}_{\mathsf{BS}}}_{\mathcal{A}}(n) = 1$$

and asking $O(k \cdot |\mathcal{C}_{\Sigma}|)$ queries to the hash function, where $\mathcal{C}_{\Sigma}$ is the challenge space of the base sigma protocol.

3.4 One out of one attack

In this attack, the adversary, by opening only one session with the signer, obtains two signatures. This, however, comes with a trade-off of exponential computational cost. More precisely, suppose the attacker cannot open concurrent sessions, i.e., sequential setting. We will show how the attacker can produce two valid signatures after one session with the signer. Let $\mathsf{BS} = \mathsf{BS}^k_{\Sigma}$ be the blind signature scheme. Suppose, for simplicity, the number of repetition k is an even number and set $t = k/2$. We construct $\mathcal{A}$ as follows.

1. The attacker opens one session BS with the signer. Let $\mathbf{R} = (R_1, \ldots, R_k)$ be the signer commitment and let:

$$\mathbf{R}' = (R'_1, \ldots, R'_k),$$

   be the blinded commitment.
2. For any $1 \le i \le k$, $\mathcal{A}$ will generate k valid transcripts (r_i, d_i, z_i) for the underlining Σ-protocol, and define

$$\mathbf{R}'_1 = (r_1, \ldots, r_t, R'_1, \ldots, R'_t),$$

   and

$$\mathbf{R}'_2 = (r_{t+1}, \ldots, r_k, R'_{t+1}, \ldots, R'_k).$$

It will then search for two messages m_1 and m_2 such that:

$$\mathbf{c}'_1 = \mathcal{H}(\mathbf{R}'_1 \| m_1) = (d_1, \ldots, d_t, c'_{1,t+1}, \ldots, c'_{1,k})$$

and

$$\mathbf{c}'_2 = \mathcal{H}(\mathbf{R}'_2 \| m_2) = (d_{t+1}, \ldots, d_k, c'_{2,t+1}, \ldots, c'_{2,k}).$$

This will require $O(|\mathcal{C}_\Sigma|^t)$ queries to the hash funcion $\mathcal{H}$.

3. The attacker now sets $\widetilde{\mathbf{c}} = (c'_{1,t+1}, \ldots, c'_{1,k}, c'_{2,t+1}, \ldots, c'_{2,k})$, then blinds it as $\mathbf{c} \leftarrow \mathsf{U}_1.\mathsf{BlindChal}(\widetilde{\mathbf{c}})$ and sends this to the signer. Upon receiving the response $\mathbf{s}$, it blinds it as $\widetilde{\mathbf{s}} \leftarrow \mathsf{U}_2.\mathsf{BlindResp}(\mathbf{s})$ and sets

$$\mathbf{s}'_1 = (z_1, \ldots, z_t, \widetilde{s}_1, \ldots, \widetilde{s}_t),$$

and

$$\mathbf{s}'_2 = (z_{t+1}, \ldots, z_k, \widetilde{s}_{t+1}, \ldots, \widetilde{s}_k).$$

4. The attacker will close the session and output $(m_1, (\mathbf{R}'_1, \mathbf{s}'_1))$, and $(m_2, (\mathbf{R}'_2, \mathbf{s}'_2))$.

Lemma 3. *The pairs $(\mathbf{R}'_1, \mathbf{s}'_1)$ and $(\mathbf{R}'_2, \mathbf{s}'_2)$ are valid signatures for the messages m_1 and m_2 respectively.*

Proof. The proof of this theorem is analogous to that of lemma 1, and hence it will be omitted.

Theorem 4. *Let $\mathsf{BS} = \mathsf{BS}^k_\Sigma$ be a parallel repetition of a Schnorr-type blind signature scheme not supporting any concurrent sessions. There exists an adversary $\mathcal{A}$ against the ℓ-one-more unforgeability game, for $\ell = 1$, such that*

$$\mathsf{Adv}^{\ell\text{-}\mathbf{OMUF}_{\mathsf{BS}}}_{\mathcal{A}}(n) = 1$$

and asking $O(2 \cdot |\mathcal{C}_\Sigma|^t)$ queries to the hash function, where $\mathcal{C}_\Sigma$ is the challenge space of the base sigma protocol and $t = \lceil k/2 \rceil$.

4 Cryptanalysis of CSI-Otter

As a concrete example, we will apply our attack to the isogeny-based blind signature scheme CSI-Otter [KLLQ23] as one of the state-of-the-art blind signatures based on Σ-protocols.

4.1 Cryptographic Group Actions

The scheme we will describe is based on *cryptographic group actions with twist* [OZ23, DHK+23, CLM+18]. We will, therefore, provide some background.

Definition 5 (Group action). *We say that a group $(G, +)$ acts on a set X if there exists a function $\star : G \times X \to X$ satisfying the following properties.*

1. *Identity: for any $x \in X$, we have $0 \star x = x$ where 0 is the identity element of G.*

2. *Compatibility: for any $g, h \in G$ and any $x \in X$, we have $(g+h) \star x = g \star (h \star x)$.*
3. *Transitivity: for every $x_1, x_2 \in X$, there exists a group element $g \in G$ such that $x_2 = g \star x_1$.*
4. *Free: for each group element $g \in G$, g is the identity element if and only if there exists some set element $x \in X$ such that $x = g \star x$.*

We will say, for brevity, that $(G, X, \star)$ is a group action.

Let $(G, X, \star)$ be a group action and $x_0 \in X$ a set element. It follows from the definition that, every $x \in X$ can be written as $x = g \star x_0$ for a (unique) $g \in G$. We define the *twist* of a set element $x = g \star x_0$ as $x^{-1} = (-g) \star x_0$.

The specific group action used in CSI-Otter is the CSIDH-512 action defined in [CLM+18]. In particular, X is the set of supersingular elliptic curves[4] over $\mathbb{F}_p$ with $\mathbb{F}_p$-rational endomorphism ring isomorphic[5] to an order $\mathcal{O} \subseteq \mathbb{Q}(\sqrt{-p})$; the group G is the ideal class group $\mathcal{C}\ell(\mathcal{O})$ acting on X via isogeny; and $x_0 = E_0$ is the curve[6] defined by:

$$E_0 : y^2 = x^3 + x.$$

We also remark that, for this specific action, the twit can be efficiently computed for every $x \in X$ [Sil86]. For the cryptographic definitions of security, we refer to [ADMP20].

4.2 The Scheme

Let $(G, X, \star)$ be a cryptographic group action with twist, and let $x_0 \in X$ be a set element. Recall that x^{-1} represents the twist of $x \in X$. We now describe the blind signature scheme presented in [KLLQ23]. In the underlying sigma protocol, the prover $\mathcal{P}$ holds a secret key $\mathsf{sk} = (\delta, a_\delta) \in \{0,1\} \times G$ and a public key $\mathsf{pk} = (y_0, y_1) = (a_0 \star x_0, a_1 \star x_0)$. The interaction is described in figure 5. This Σ-protocol is correct, special sound and honest verifier zero-knowledge [BKP20]. For the latter property, we can construct a simulator Sim that, given a challenge $c \in \{-1, 1\}$ and a public key (y_0, y_1), samples random $(c_0, c_1) \leftarrow\$ (\{-1,1\})^2$ and $(r_0, r_1) \leftarrow\$ G^2$ conditioned on $c_0 \cdot c_1 = c$. It then sets $R_b = r_b \star y_b^c$ for $b \in \{0, 1\}$, and outputs the simulated transcript $((R_0, R_1),\ c,\ (r_0, r_1, c_0, c_1))$.

The blind signature scheme $\mathsf{BS}_\Sigma = (\mathcal{G}, \mathcal{U}, \mathcal{S}, \mathcal{V})$ is defined as follows:

$\mathcal{G}(1^n)$: On input the security parameter 1^n, it samples a bit $\delta \leftarrow\$ \{0,1\}$, a pair $(a_0, a_1) \leftarrow\$ G^2$ and outputs a public key $\mathsf{pk} = (y_0, y_1) = (a_0 \star x_0, a_1 \star x_0)$ and secret key $\mathsf{sk} = (\delta, a_\delta)$.

$\mathsf{S}_1(\mathsf{sk})$: The signer first samples $r_\delta \leftarrow\$ G$ and sets $R_\delta = r_\delta \star x_0$. It then samples $(c_{1-\delta}, r_{1-\delta}) \leftarrow\$ \{-1,1\} \times G$ and sets $R_{1-\delta} = r_{1-\delta} \star y_{1-\delta}^{c_{1-\delta}}$. It then outputs the signer state $\mathsf{st}_\mathsf{S} = (r_\delta, c_{1-\delta}, r_{1-\delta})$ and the signer commitment $R = (R_0, R_1)$.

[4] Modulo isomorphism over $\mathbb{F}_p$.

[5] We also require that an element $\pi \in \mathcal{O}$ maps to the Frobenius endomorphism trough that isomorphism.

[6] This is in fact supersingular for primes $p \equiv 3 \pmod 4$.

Prover$(\mathsf{pk}, \mathsf{sk})$		**Verifier**(pk)
$r_\delta \leftarrow\$ G$		
$R_\delta \leftarrow r_\delta \star x_0$		
$(c_{1-\delta}, r_{1-\delta}) \leftarrow\$ \{-1,1\} \times G$		
$R_{1-\delta} \leftarrow r_{1-\delta} \star y_{1-\delta}^{c_{1-\delta}}$	$\xrightarrow{(R_0, R_0)}$	
	$\xleftarrow{c}$	$c \leftarrow\$ \{-1,1\}$
$c_\delta \leftarrow c \cdot c_{1-\delta}$		
$r_\delta \leftarrow r_\delta - a_\delta \cdot c_\delta$	$\xrightarrow{(r_0, r_1, c_0, c_1)}$	Accept if $c = c_0 \cdot c_1$ and
		$\forall b \in \{0,1\},\ r_b \star y_b = R_b$.

Fig. 5. The interaction between $\mathcal{P}$ and $\mathcal{V}$.

$\mathsf{U}_1(\mathsf{pk}, m, R)$: The user parses $(R_0, R_1) \leftarrow R$, samples $(e_b, t_b) \leftarrow\$ \{-1,1\} \times G$, and computes $R'_b = t_b \star (R_b)^{e_b}$ for $b \in \{0,1\}$. It then computes $c' = \mathcal{H}(R'_0 \| R'_1 \| m)$ and outputs the user state $\mathsf{st_U} = (e_b, t_b)_{b \in \{0,1\}}$ and user message $c = c' \cdot e_0 \cdot e_1$.

$\mathsf{S}_2(\mathsf{sk}, c, \mathsf{st_S})$: The signer parses $(r_\delta, c_{1-\delta}, r_{1-\delta}) \leftarrow \mathsf{st_S}$, sets $c_\delta = c \cdot c_{1-\delta} \in \{-1,1\}$, and updates $r_\delta \leftarrow r_\delta - a_\delta \cdot c_\delta \in G$. It then outputs the signer response $s = (c_b, r_b)_{b \in \{0,1\}}$.

$\mathsf{U}_2(\mathsf{pk}, s, \mathsf{st_U})$: The user parses $(e_b, t_b)_{b \in \{0,1\}} \leftarrow \mathsf{st_U}$, $(c_b, r_b)_{b \in \{0,1\}} \leftarrow s$ and sets $(c'_b, r'_b) = (c_b \cdot e_b, t_b + r_b \cdot e_b)$ for $b \in \{0,1\}$. It then checks if

$$c'_0 \cdot c'_1 = \mathcal{H}(r'_0 \star y_0^{c'_0} \| r'_1 \star y_1^{c'_1} \| m). \tag{4}$$

If it holds, it outputs a signature $\sigma = (c'_b, r'_b)_{b \in \{0,1\}} \in (\{-1,1\} \times G)^2$, and otherwise a $\perp$.

$\mathcal{V}(\mathsf{pk}, m, \sigma)$: The verifier outputs 1 if equation 4 holds, and otherwise 0.

This is, in fact, a Schnorr-type blind signature. It is clear from the construction that $\mathsf{S}_1 = \mathsf{P}_1$ and $\mathsf{S}_2 = \mathsf{P}_2$, and we have the following table for the user.

```
U_1(pk, m, R = (R_0, R_1))
1 : for $b \in \{0,1\}$
2 :     $(e_b, t_b) \leftarrow\$ \{-1,1\} \times G$
3 :     $R'_b \leftarrow t_b \star (R_b)^{e_b}$
4 : $R' \leftarrow (R'_0, R'_1)$
5 : $c' \leftarrow \mathcal{H}(R' \| m)$
6 : $c \leftarrow c' \cdot e_0 \cdot e_1$
7 : $\mathsf{st_U} \leftarrow (e_b, t_b)_{b \in \{0,1\}}$
8 : return $(c, \mathsf{st_U})$
```

```
U_2(pk, s = (c_b, r_b)_{b∈{0,1}}, st_U)
1 : for $b \in \{0,1\}$
2 :     $(c'_b, r'_b) \leftarrow c_b \cdot e_b, t_b + r_b \cdot e_b$
3 : $s' \leftarrow (c'_b, r'_b)_{b \in \{0,1\}}$
4 : $c^* \leftarrow \mathcal{H}(r'_0 \star y_0^{c'_0} \| r'_1 \star y_1^{c'_1} \| m)$
5 : if $c'_0 \cdot c'_1 = c^*$
6 :     return $\perp$
7 : $\sigma \leftarrow s'$
8 : return $\sigma$
```

Blue lines represent the function $\mathsf{U}_1.\mathsf{BlindCom}$; the line yellow represents $\mathsf{U}_1.\mathsf{BlindChal}$ and the violet $\mathsf{U}_2.\mathsf{BlindResp}$. It follows, now, that the blind signature BS^k_Σ is vulnerable to our attacks since the base challenge space $\mathcal{C}_\Sigma$ has cardinality 2 and k is linear in the security parameter n.

5 Discussion

This section discusses potential ways of circumventing our attack in the concurrent setting, their downsides, and implications on the efficiency and practicality of the blind signature vulnerable to our attack. We also discuss implications on sequential security, particularly a concrete treatment of those schemes in the post-quantum setting.

5.1 Concurrent security

For our mix-and-match attack to work, the adversary must be able to initiate and finalize k signing sessions where the signer supports at least two concurrent sessions at a time. The parameter k depends on the challenge space $\mathcal{C}_\Sigma$ of the underlying Σ-protocol, e.g., in the case of CSI-Otter, the authors use the challenge space $\mathcal{C}_\Sigma = \{-1, 1\}$ and set the number of repetitions to $k = 128$. This parameter is usually polynomial in the security parameter to ensure that the soundness error for the augmented Σ-protocol is negligible while keeping it small to ensure efficiency. For almost all practical applications, the parameter $k = 128$ ensures that the soundness error of the Σ-protocol is negligible for 128-bit security. Thus, in most cases, any blind signature scheme from Σ-protocol with small challenge space will be forgeable after just around a hundred signed messages.

A potential way to circumvent our attack is to increase k since it is allowed to be polynomial in the security parameter. This way, the number of supported signatures will also be polynomial in the security parameter, where the polynomial can be freely chosen. Unfortunately, this approach will significantly decrease the efficiency of the used scheme. Let us discuss this on the CSI-Otter example. Doubling the parameter k to 256 will also double the number of signatures, allowing for one forgery, i.e., from 128 to 256 (note that we are considering the main attack here). At the same time, the communication complexity of the exchange and the signature size will also be doubled. We usually set parameters for a standard digital signature scheme so that one key pair can be used to sign around 2^{30}. To achieve a similar property while protecting against our main attack, we must set $k = 2^{30}$, leading to signatures in the gigabytes and inefficient computations.

Increasing the parameter k provides a simple fix that increases the number of concurrent sessions the blind signature can provide. Asymptotically, the scheme offers the required security, but it is impractical if we want a genuinely usable scheme supporting many signatures. Note that a signer can continuously refresh the key pair, which is not a problem for standard digital signatures but decreases the anonymity set in the blind signature setting.

An alternative approach would be to employ boosting transformations [KLR21, CAHL+22] known and used for the Schnorr blind signature scheme vulnerable to the ROS attack. Those solutions employ cut-and-choose techniques that allow the reduction to limit the number of signing queries while providing more questions to the adversary, i.e., in the case of Schnorr signatures, the adversary is allowed to do polynomially many queries in the security parameter while at the same time, the reduction only queries the signing oracle at most a logarithmic number of times. Similar techniques could be employed for the class of blind signatures considered in this paper. Contrary to increasing the parameter k, this approach increases the communication and signature size by a smaller factor than in the previous solution, making it more efficient. However, it is still impractical and not comparable to other blind signature schemes, e.g., applying this technique to CSI-Otter would still make it less efficient than state-of-the-art schemes based on lattices.

5.2 Sequential security

Using the threshold variant of our attack, an adversary can compute two valid signatures using only one signing session in $O(2^{k/2})$ number of hash evaluations. Thus, launching a practical attack in the sequential setting against CSI-Otter with the proposed parameters $k = 128$ requires around $2 \cdot 2 \cdot 2^{64}$ hash evaluation in expectation. This amount of computation is less than the work required by miners to add a new block to the Bitcoin blockchain. In particular, this means that the Bitcoin mining pool can, with non-negligible probability, forge CSI-Otter signatures even in the sequential setting in around 10 minutes for the parameter proposed by the authors.

Our results show that assuming the underlying Σ-protocol has a negligible soundness error for a parameter k_Σ, e.g., $k_\Sigma = n$, then, we need to set the parameter k for the blind signature to $k = 2 \cdot k_\Sigma$, i.e., in the case of $k_\Sigma = n$ this will lead to $k = 2 \cdot n$. In the case of CSI-Otter, this means that to get around the above sequential attack, the parameter k must be set to $2 \cdot 128 = 256$.

The above considerations are only for the classical setting and do not treat the hash function in the post-quantum scenario. In this setting, we can accelerate the adversary's computation using Grover's algorithm on quantum computers [Gro96]. In particular, the problem the adversary must solve is to find a message m for which $\mathcal{H}(\mathbf{R}||m) = ab$ for fixed commitments $\mathbf{R}$ and prefix a. As shown in [RBL+21] the adversary's work will be reduced to $O(2^{\frac{k}{4}})$. Therefore, the parameter k must be increased four times for the scheme to be secure against a quantum attacker.

5.3 Revisiting CSI-Otter Parameters

We will now discuss concrete parameters for CSI-Otter [KLLQ23]. The authors in the original paper propose two variants: one with challenge space $\{-1, 1\}$ with $k = 128$ repetitions and an optimized variant with a challenge space of size 4 with $k = 64$ repetitions. In both cases, our main attack can forge a new signature with

128 and 64 concurrent sessions, only evaluating the hash function several hundred times. By sacrificing a bit of computation (i.e., around one hour on a commodity M1 Pro processor with a five megahash/s rate), the same concurrent attack can be executed with just 4 concurrent sessions in total for the basic version and 2 concurrent sessions for the optimized version. For the above considerations, we still assume a pre-quantum attacker.

It is evident that CSI-Otter is not concurrently secure for the proposed parameters and design. Increasing the parameter k does not solve the problem since increasing it x times also increases the communication and signature size x times, which might blow up the signature's size to Gigabytes to allow only 2^{20} signatures, making the scheme impractical. In table 2, we provide an optimistic estimation for the scheme's efficiency after applying the proposed treatments. As argued in the introduction, CSI-Otter and, more generally, blind signatures from Σ-protocols with small challenge space should not be used concurrently without applying transformations to boost their security. Therefore, CSI-Otter can only be used as a blind signature scheme in the sequential setting.

The authors introduced the optimized variant with a challenge space of size 4 and $k = 64$ repetitions to reduce the signature size by two at the expense of the size of the secret key and communication from the signer. Further improvements with an even bigger space and smaller repetitions should be possible. Unfortunately, our attacks show that this decreases the scheme's security, allowing the adversary to forge signatures with fewer concurrent sessions and making a sequential attack more feasible. In particular, for the above case, an adversary only needs to compute $2 \cdot 4 \cdot 2^{32} = 2^{35}$ (in expectation) to forge a fresh signature with just one signing query. For the basic version of CSI-Otter, this is $2 \cdot 2 \cdot 2^{64} = 2^{66}$. The provided parameters should ensure 128-bit security, and they do if the scheme were a standard digital signature scheme. However, our attack shows that to achieve n-bit security for the blind signature scheme, we must set $k = 2 \cdot n$. Thus, the repetitions must be more than 256 for 128-bit security in both cases, making the optimized variant no longer beneficial. The number of 256 repetitions must be further increased if we want to consider the post-quantum setting where we can apply Grover's algorithm mentioned in the above section discussing sequential security.

6 Conclusion

We described three attacks against a class of three-move blind signature schemes based on Σ-protocol with small challenge space and parallel repetition. Unlike the ROS attack, the proposed attacks do not require any particular algebraic structure. This property is essential, as the lack of algebraic structure provides ambiguous arguments when assessing the concurrent security of the blind signature scheme. Moreover, because our attacks are generic and only rely on hash evaluation, there is a need for more rigorous treatment for the security parameter, especially when considered in the quantum setting.

	Bandwidth.S	Bandwidth.U	$\lvert\mathsf{sk}\rvert$	$\lvert\mathsf{pk}\rvert$	$\lvert\sigma\rvert$
Basic version ($k = 128$, $\lvert\mathcal{C}_\Sigma\rvert = 2$)	8.19 Mb	8.19 Kb	16 B	128 B	4.1 Mb
Optimized version ($k = 64$, $\lvert\mathcal{C}_\Sigma\rvert = 4$)	32.8 Mb	8.19 Kb	16 B	512 B	2.05 Mb

Table 2. An optimistic estimation for the performance of the two versions of CSI-Otter [KLLQ23] after applying a potential boosting transformation (e.g., [KLR21] with some modification) resulting in approximately 128 times larger signature size and communication bandwidth. The results include our proposed countermeasures against the post-quantum sequential attack.

We described an example case study of the CSI-Otter isogeny-based scheme and show that our mix-and-match attacks apply to this scheme. However, we highlighted that the class of aforementioned blind signature schemes is not restricted to this scheme and applies to other existing lattice-based or RSA-based constructions. CSI-Otter requires a more rigorous treatment of parameters to achieve practical sequential security. In contrast, one must apply boosting transformations for concurrent security, significantly worsening the scheme's performance, communication, and signature size, making it less attractive than other post-quantum secure candidates (e.g., scheme from lattice assumptions).

We conclude by observing that, in the isogeny-based setting, Σ-protocols with exponentially large challenge space exist, e.g., SQISign [DKL+20]. It is unclear, however, whether it is possible to turn those Σ-protocols into a blind signature or not. We leave this question to potential future work.

References

ADMP20. N. Alamati, L. De Feo, H. Montgomery, and S. Patranabis. Cryptographic Group Actions and Applications. In S. Moriai and H. Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 411–439. Springer, Heidelberg, December 2020.

AO00. M. Abe and T. Okamoto. Provably Secure Partially Blind Signatures. In M. Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 271–286. Springer, Heidelberg, August 2000.

BKP20. W. Beullens, S. Katsumata, and F. Pintore. Calamari and Falafl: Logarithmic (Linkable) Ring Signatures from Isogenies and Lattices. In S. Moriai and H. Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 464–492. Springer, Heidelberg, December 2020.

BKV19. W. Beullens, T. Kleinjung, and F. Vercauteren. CSI-FiSh: Efficient Isogeny Based Signatures Through Class Group Computations. In S. D. Galbraith and S. Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 227–247. Springer, Heidelberg, December 2019.

BLL+21. 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, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 33–53. Springer, Heidelberg, October 2021.

BNPS03. M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme. *Journal of Cryptology*, 16(3):185–215, June 2003.

Bol03. A. Boldyreva. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In Y. Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Heidelberg, January 2003.

CAHL$^+$22. R. Chairattana-Apirom, L. Hanzlik, J. Loss, A. Lysyanskaya, and B. Wagner. PI-Cut-Choo and Friends: Compact Blind Signatures via Parallel Instance Cut-and-Choose and More. In Y. Dodis and T. Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 3–31. Springer, Heidelberg, August 2022.

Cha82. D. Chaum. Blind Signatures for Untraceable Payments. In D. Chaum, R. L. Rivest, and A. T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982.

CLM$^+$18. W. Castryck, T. Lange, C. Martindale, L. Panny, and J. Renes. CSIDH: An Efficient Post-Quantum Commutative Group Action. In T. Peyrin and S. Galbraith, editors, *ASIACRYPT 2018, Part III*, volume 11274 of *LNCS*, pages 395–427. Springer, Heidelberg, December 2018.

CP93. D. Chaum and T. P. Pedersen. Wallet Databases with Observers. In E. F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, Heidelberg, August 1993.

DG19. L. De Feo and S. D. Galbraith. SeaSign: Compact Isogeny Signatures from Class Group Actions. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 759–789. Springer, Heidelberg, May 2019.

DHK$^+$23. J. Duman, D. Hartmann, E. Kiltz, S. Kunzweiler, J. Lehmann, and D. Riepel. Generic Models for Group Actions. In A. Boldyreva and V. Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 406–435. Springer, Heidelberg, May 2023.

DKL$^+$20. L. De Feo, D. Kohel, A. Leroux, C. Petit, and B. Wesolowski. SQISign: Compact Post-quantum Signatures from Quaternions and Isogenies. In S. Moriai and H. Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 64–93. Springer, Heidelberg, December 2020.

DvW22. L. Ducas and W. P. J. van Woerden. On the Lattice Isomorphism Problem, Quadratic Forms, Remarkable Lattices, and Cryptography. In O. Dunkelman and S. Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 643–673. Springer, Heidelberg, May / June 2022.

FPS20. G. Fuchsbauer, A. Plouviez, and Y. Seurin. Blind Schnorr Signatures and Signed ElGamal Encryption in the Algebraic Group Model. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 63–95. Springer, Heidelberg, May 2020.

FS87. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In A. M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

Gro96. L. K. Grover. A Fast Quantum Mechanical Algorithm for Database Search. In *28th ACM STOC*, pages 212–219. ACM Press, May 1996.

HKL19. E. Hauck, E. Kiltz, and J. Loss. A Modular Treatment of Blind Signatures from Identification Schemes. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 345–375. Springer, Heidelberg, May 2019.

HKLN20. E. Hauck, E. Kiltz, J. Loss, and N. K. Nguyen. Lattice-Based Blind Signatures, Revisited. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 500–529. Springer, Heidelberg, August 2020.

KLLQ23. S. Katsumata, Y.-F. Lai, J. T. LeGrow, and L. Qin. CSI-Otter: Isogeny-Based (Partially) Blind Signatures From the Class Group Action With a Twist. In *Advances in Cryptology – CRYPTO 2023: 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20–24, 2023, Proceedings, Part III*, pages 729–761, Berlin, Heidelberg, 2023. Springer-Verlag.

KLR21. J. Katz, J. Loss, and M. Rosenberg. Boosting the Security of Blind Signature Schemes. In M. Tibouchi and H. Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 468–492. Springer, Heidelberg, December 2021.

KLX22. J. Kastner, J. Loss, and J. Xu. The Abe-Okamoto Partially Blind Signature Scheme Revisited. In S. Agrawal and D. Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 279–309. Springer, Heidelberg, December 2022.

LNP22. V. Lyubashevsky, N. K. Nguyen, and M. Plançon. Efficient Lattice-Based Blind Signatures via Gaussian One-Time Signatures. In G. Hanaoka, J. Shikata, and Y. Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 498–527. Springer, Heidelberg, March 2022.

LNSW13. S. Ling, K. Nguyen, D. Stehlé, and H. Wang. Improved Zero-Knowledge Proofs of Knowledge for the ISIS Problem, and Applications. In K. Kurosawa and G. Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 107–124. Springer, Heidelberg, February / March 2013.

OZ23. E. Orsini and R. Zanotto. Simple Two-Round OT in the Explicit Isogeny Model. Cryptology ePrint Archive, Paper 2023/269, 2023. `https://eprint.iacr.org/2023/269`.

Poi98. D. Pointcheval. Strengthened Security for Blind Signatures. In K. Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 391–405. Springer, Heidelberg, May / June 1998.

PS00. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.

RBL$^+$21. S. Ramos-Calderer, E. Bellini, J. I. Latorre, M. Manzano, and V. Mateu. Quantum search for scaled hash function preimages. *Quantum Inf. Process.*, 20(5):180, 2021.

Sil86. J. H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate texts in mathematics*. Springer, 1986.