Breaking Parallel ROS: Implication for Isogeny and Lattice-based Blind Signatures

Shuichi Katsumata[1], Yi-Fu Lai[2], Michael Reichle[3]

[1]PQShield and AIST
shuichi.katsumata@pqshield.com
[2]Ruhr-Universität Bochum
Yi-Fu.Lai@ruhr-uni-bochum.de
[3]ETH Zürich
michael.reichle@inf.ethz.ch

October 16, 2023

Abstract

Many of the three-round blind signatures based on identification protocols are only proven to be ℓ-concurrently unforgeable for $\ell = \mathsf{polylog}(\lambda)$. It was only recently shown in a seminal work by Benhamouda et al. (EUROCRYPT'21) that this is not just a limitation of the proof technique. They proposed an elegant polynomial time attack against the ℓ-concurrently unforgeability of the classical blind Schnorr protocol for $\ell = \mathsf{poly}(\lambda)$. However, there are still many blind signatures following a similar recipe to blind Schnorr where the attack by Benhamouda et al. does not apply. This includes for instance the isogeny-based blind signature CSI-Otter by Katsumata et al (CRYPTO'23), the lattice-based blind signatures Blaze+ by Alkeilani et al (ACISP'20) and BlindOR by Alkeilani et al (CANS'20).

In this work, we provide a simple and novel attack on blind signatures based on identification protocols performing *parallel repetition* to reduce the soundness error. Our attack translates to a polynomial time break for the ℓ-concurrent unforgeability of CSI-Otter, Blaze+, and BlindOR for $\ell = \mathsf{poly}(\lambda)$. More formally, we define an intermediate problem called Parallel Random inhomogeneities in an Overdetermined Solvable system of linear equations (pROS) problem and show that an attack against pROS implies an attack to the above blind signatures. One takeaway of our finding is that while parallel repetition allows to exponentially reduce the soundness error of an identification protocol, this has minimal effect on the resulting blind signature. Our attack is concretely very efficient and for instance breaks 4-concurrent unforgeability of CSI-Otter in time roughly 2^{34} hash computations. Furthermore, it can achieve the same result for Blaze and BlindOR with approximately 2^{43} hash computations, offering a 7% and an overwhelming chance of success, respectively.

1 Introduction

Blind signature is an interactive signing protocol between a signer and a user with an advanced privacy feature. Originally envisioned to be used for e-cash in the early 80's [Cha82], we now have numerous applications in direct anonymous attestation [BCC04], privacy-preserving authentication tokens [VPN22, HIP+22], cryptocurrencies and blockchains [CKLR18, YL19, BDE+22], to name a few.

Informally, *blindness* guarantees that a user with a message can obtain a signature from the signer, while the signer remains oblivious of the message it signed. Due to blindness, unforgeability can no longer be defined as in a standard signature; a challenger (playing the role of the signer) cannot decide if the

forgery is on a message that it has not signed before. To this end, we define ℓ-*concurrent unforgeability*[1], guaranteeing that if a signer finished ℓ signing sessions, then no user can output $\ell+1$ or more valid signatures. Importantly, we allow the user to *concurrently* open signing sessions. For instance, a malicious user may try to concurrently open ℓ signing sessions and mix them together to create $\ell+1$ signatures.

3-Round Protocols. One of the popular approaches to construct blind signatures is to base it on identification protocols. Starting with the blind Schnorr protocol based on the Schnorr identification protocol [CP93], this approach has been very successful, being instantiable from versatile assumptions including post-quantum assumptions such as lattices and isogenies [AO00, HKL19, HKLN20, AEB20b, AHJ21, KLLQ23]. Compared to other approaches based on pairing specific techniques [AGHO11, BFPV13, MSF10, SC12, KSD19] and/or non-interactive zero-knowledge proofs [Fis06, dK22, KRS23, BLNS23, FW22], this approach is easier to generalize, leads to simpler constructions, and tends to be more efficient.

One peculiarity of many of the blind signatures based on this approach is that they were only proven to be ℓ-concurrent unforgeable for $\ell = \mathsf{polylog}(\lambda)$ with λ the security parameter. Interestingly, while the proof could not tolerate $\ell = \mathsf{poly}(\lambda)$, it was unclear whether this was an artifact of the proof technique or because there was a yet to be discovered attack. This issue was undesirable even from a practical point view as we did not know whether these blind signatures will remain secure when instantiated with concrete parameters, say what happens if the signer opened 128 concurrent sessions?

The ROS Attack. Schnorr [Sch01] introduced the Random inhomogeneities in an Overdetermined Solvable system of linear equations (ROS_ℓ) problem in dimension ℓ, and showed that a ROS_ℓ solver can be used to break the ℓ-concurrent unforgeability of the Schnorr signature. Wagner [Wag02] soon after showed that the ROS_ℓ problem can be solved in subexponential time when ℓ grows asymptotically faster than $\mathsf{polylog}(\lambda)$. While this implies a subexponential timed attack on the ℓ-concurrent unforgeability of blind Schnorr for $\ell = \mathsf{poly}(\lambda)$, showing the (in)existence of a polynomial time attack remained elusive for nearly two decades.

It was only recently in a seminal work, Benhamouda et al. [BLL+21] proposed an elegant polynomial time attack against ROS_ℓ for $\ell = \mathsf{poly}(\lambda)$, finally *"partially"* closing the above issue: blind Schnorr is not ℓ-concurrent unforgeable for $\ell = \mathsf{poly}(\lambda)$. Their attack is very practical and for instance when $\ell = 128$, it only takes time roughly 2^{32} hash computations to break unforgeability.

Unaffected Schemes by the ROS Attack. The reason why we highlighted that the ROS attack by Benhamouda et al. is only a partial solution to the issue was because many of the post-quantum blind signatures [HKLN20, AEB20b, AHJ21, KLLQ23] remain unaffected by the attack. For instance, the lattice-based blind signature by Hauck et al. [HKLN20] is related to a slightly generalized variant of the ROS problem for which the ROS attack by Benhamouda et al. does not seem to immediately apply.

The lattice-based blind signatures Blaze+, BlindOR [AEB20b, AHJ21] are even more different. The base identification protocol underlying these blind signatures has a small challenge set, and therefore, performs parallel repetition to reduce its soundness error. Due to this parallel repetition, the underlying problem is no longer the original ROS problem considered by Schnorr [Sch01] and it is unclear whether the ROS attack of Benhamouda et al. applies. A related question is whether a blind Schnorr protocol constructed from a Schnorr identification protocol with parallel repetition can resurrect ℓ-concurrent unforgeability for $\ell = \mathsf{poly}(\lambda)$: while at the identification protocol layer, parallel repetition exponentially reduces the soundness error, how would this relate to ℓ-concurrent unforgeability?

Lastly, the recent isogeny-based blind signature CSI-Otter [KLLQ23] also relies on a base identification protocol with parallel repetition. Adding to the complexity is that unlike lattices and classical groups that are modules, isogenies have a strictly weaker algebraic structure called group actions. As stated by Katsumata et al. [KLLQ23], due to the lack of algebraic structures in isogenies, even defining an appropriate ROS problem underlying the security of CSI-Otter is non-trivial, and they left it as an open problem to examine the (in)security of their scheme for $\mathsf{poly}(\lambda)$ many concurrent sessions.

[1] In the literature, this is typically coined as *one-more unforgeability*. We use our terminology throughout the introduction to be precise on the value of ℓ and concurrency.

1.1 Contribution

In this work, we propose the *parallel* ROS ($\mathsf{pROS}_{\ell,\omega,\mathcal{C}}$) problem and show that it is solvable in polynomial time for appropriate parameters. As a consequence, we show that the lattice-based blind signatures Blaze+, BlindOR [AEB20b, AHJ21] and the isogeny-based blind signature CSI-Otter [KLLQ23] are *not* ℓ-concurrently unforgeable when $\ell = \mathsf{poly}(\lambda)$. Our attack is very practical and for instance we can break the 4-concurrent unforgeability of CSI-Otter in time roughly 2^{34} hash computations.

In more detail, the $\mathsf{pROS}_{\ell,\omega,\mathcal{C}}$ problem captures the hardness of the ℓ-concurrent unforgeability of blind signatures based on an identification protocol with challenge space $\mathcal{C}$ performing ω parallel repetitions. For instance, when $\omega = 1$ and $\mathcal{C} = \mathbb{Z}_p$, the $\mathsf{pROS}_{\ell,\omega,\mathcal{C}}$ problem is identical to the standard ROS_ℓ problem, capturing the hardness of the ℓ-concurrent unforgeability of blind Schnorr. As another example, the $\mathsf{pROS}_{\ell,\omega,\mathcal{C}}$ problem underlying the hardness of the ℓ-unforgeability of CSI-Otter [KLLQ23] is $\omega = \lambda$ and $\mathcal{C} = \{-1, 1\}$. To be more precise, as discussed above, since isogenies are algebraically different from lattices and classical groups, we define two types of pROS problems over different mathematical structures capturing each cases: $\mathsf{Group\text{-}pROS}_{\ell,\omega,\mathcal{C}}$ for group actions (with a twist) and $\mathsf{Ring\text{-}pROS}_{\ell,\omega,\mathcal{C}}$ for modules.

Our main technical contribution is showing informally the following results:

- $\mathsf{Group\text{-}pROS}_{\ell,\omega,\mathcal{C}}$ can be solved in time $\omega \cdot \mathsf{poly}(|\mathcal{C}|)$ for any $\ell \geq \omega = \mathsf{poly}(\lambda)$ and $\mathcal{C}$. When $|\mathcal{C}| = \mathsf{poly}(\lambda)$, the attack runs in polynomial time.
- $\mathsf{Ring\text{-}pROS}_{\ell,\omega,\mathcal{C}}$ can be solved in time $\omega \cdot \mathsf{poly}(\log(|\mathcal{C}|))$ for any $\ell \geq \omega \cdot \log(|\mathcal{C}|) = \mathsf{poly}(\lambda)$ and $\mathcal{C}$. Even if $|\mathcal{C}| = \mathsf{exp}(\lambda)$, the attack runs in polynomial time.

Importantly, parallel repetition only amplifies *linearly* the hardness of the ℓ-concurrent unforgeability. An immediate takeaway is that while parallel repetition allows to exponentially reduce the soundness error of an identification protocol, this has minimal effect on the resulting blind signature. Moreover, note that when $|\mathcal{C}| = \mathsf{poly}(\lambda)$, our attack against $\mathsf{Ring\text{-}pROS}_{\ell,\omega,\mathcal{C}}$ does not take advantage of the module structure as we can simply break $\mathsf{Ring\text{-}pROS}_{\ell,\omega,\mathcal{C}}$ using the algorithm for breaking $\mathsf{Group\text{-}pROS}_{\ell,\omega,\mathcal{C}}$. The technical overview of our attacks are provide in Section 3, where we further present a modified attack on $\mathsf{Group\text{-}pROS}_{\ell,\omega,\mathcal{C}}$, leading to very practical attacks.

Using the above attack on the Group-pROS and Ring-pROS problems, we are able to break the ℓ-concurrent unforgeability for $\ell = \mathsf{poly}(\lambda)$ of the following schemes:

- The isogeny-based blind signature CSI-Otter by [KLLQ23]. (See Section 4.1.) Concretely, we can break 4-concurrent unforgeability in time roughly 2^{34} hash computations.
- Two lattice-based blind signatures Blaze+, BlindOR [AEB20b, AHJ21]. (See Section 4.2.) Concretely, we can break 4-concurrent unforgeability of both schemes in time roughly 2^{46} hash computations with a success probability of roughly 7.3%.
- Blind Schnorr with parallel repetition. (See Section 4.3.) For, e.g., 4 parallel repetitions, we can break 1024-concurrent unforgeability in time roughly 2^{11} hash computations.

We would like to emphasize that our attack does not contradict the security proof of ℓ-concurrent unforgeability for $\ell = \mathsf{polylog}(\lambda)$ provided in previous works. Our asymptotic attack uses the fact that an adversary can initiate $\ell = \mathsf{poly}(\lambda)$ concurrent sessions, a setting which previous works do not consider. Though, it is worth highlighting that our concrete attack indicates that in practice, ℓ can be quite small (i.e., $\ell = 4$ for CSI-Otter and $\ell \leq 16$ for Blaze+, BlindOR) to practically break the schemes.

Future Works. While our attack presents a concretely efficient break to the ℓ-concurrent unforgeability of the lattice-based blind signatures Blaze+, BlindOR [AEB20b, AHJ21] and the isogeny-based blind signature CSI-Otter [KLLQ23], there may be ways to fix this using the techniques developed by Abe [Abe01, KLX22b] or Tessaro and Zhu [TZ22]. By tweaking the original blind Schnorr protocol in different ways, they are able to prove ℓ-concurrent unforgeability for $\ell = \mathsf{poly}(\lambda)$ in either the generic group model or in the algebraic

group model. Considering how simple and efficient Blaze+, BlindOR, and CSI-Otter are, it will be worthwhile to enhance their security with minimal modification while retaining efficiency.

We also leave analyzing the (in)security of the ℓ-concurrent unforgeability for $\ell = \mathsf{poly}(\lambda)$ of the lattice-based blind signature by Hauck et al. [HKLN20] as an important open problem. Assuming the underlying polynomial ring R_q used in [HKLN20] splits into many fields, it almost seems that our pROS attack applies. However, due to lattice specific reasons, this attack does not seem to work. As mentioned above, the ROS attack by Benhamouda et al. [BLL+21] does not immediately apply either.

2 Preliminary

Notation. We denote the set of natural numbers and integers by $\mathbb{N}$ and $\mathbb{Z}$, respectively. We define the ring of integers modulo N, i.e., $\mathbb{Z}_N$, with representatives in $[-q/2, q/2) \cap \mathbb{Z}$. For a positive integer k, we let $[k]$ denote the set $\{1, 2, \ldots, k\}$. For a distribution D, we write $x \leftarrow D$ to denote x is sampled according to D. For a finite set S, we denote $x \leftarrow S$ to sample x uniformly at random over S. We use $\odot$ to denote the component-wise multiplication of vectors in a multiplicative group $\mathcal{G}$. We sometimes use $\|$ to denote the concatenation of two strings. For an element g and vector $\mathbf{a} = (a_1, \ldots, a_n)$, we use $g^{\mathbf{a}}$ as a shorthand for $(g^{a_1}, \ldots, g^{a_n})$. Moreover, for any operation $*$ defined between two elements g and h and vectors $\mathbf{a} = (a_1, \ldots, a_n)$ and $\mathbf{b} = (b_1, \ldots, b_n)$, we use $g^{\mathbf{a}} * h^{\mathbf{b}}$ as a shorthand for $(g^{a_1} * h^{b_1}, \ldots, g^{a_n} * h^{b_1})$. We extend this notation to matrices naturally. For readability, we use the arrow notation (e.g. $\vec{d}$) or the bold lowercase letters (e.g. $\mathbf{e}$) to denote a vector. We use the bold uppercase letters (e.g. $\mathbf{A}$) to represent a matrix in the lattice context. For a matrix $\mathbf{A}$, we denote the i-th row of $\mathbf{A}$ to be $\mathbf{A}[i]$. Let $(\mathcal{G}, \cdot)$ be a group. Let $\mathbf{a} \in (\mathcal{G} \cup \{\perp\})^n$ and $\mathbf{b} \in \mathcal{G}^n$. We write $\|\mathbf{a}\|_\infty = 1$ if $\mathbf{a}$ has at most one non-bot entry α. Then, we define multiplication $\mathbf{a} \cdot \mathbf{b} := (\alpha \cdot b_k)_{k \in [n]}$. We extend this notation to matrices naturally.

2.1 Cyclic Effective Group Action Model

To describe the the isogeny-based blind signature CSI-Otter [KLLQ23], we adapt the cyclic effective group action model given in [DHK+23], which captures the essence of the isogeny group action used in the scheme.

Definition 2.1 (Cyclic Effective Group Action with Twists). *Let the group G act on the set $\mathcal{X}$ by $\diamond$. The tuple $(G, \mathcal{X}, \diamond, E_0)$ is said to be a* cyclic effective group action with twists *(CEGAwT) if*

1. *G is finite and cyclic of order N for some known $N \in \mathbb{N}$.*
2. *There exists a known generator $g \in G$ with known representation (i.e. $G = \langle g \rangle$).*
3. *There exist efficient algorithms for membership testing and to compute a unique representation for any element in $\mathcal{X}$.*
4. *The group action $(G, \mathcal{X}, \diamond)$ is regular.*
5. *E_0 is a distinguished element in $\mathcal{X}$ with known representation.*
6. *There exists an algorithm such that for any element $a \in \mathbb{N}$ and $x \in \mathcal{X}$ the action $g^a \diamond x$ is efficiently computable.*
7. *There exists an efficient twisting algorithm on input $x' = h \star E_0$ computing $h^{-1} \star E_0$.*

The isomorphism $G = \langle g \rangle \cong \mathbb{Z}_N$ gives a standard representation for G. This, in turn, naturally induecs a CEGAwT $(\mathbb{Z}_N, \mathcal{X}, \star, E_0)$ where the action $m \star x := g^a \diamond x$. The structure of $\mathbb{Z}_N$ naturally gives efficient algorithms for membership testing, random sampling, and equality testing. Hence, on input $n \star E_0$, the twisting algorithm returns $-n \star E_0$.

Throughout this paper, we will use $(\mathbb{Z}_N, \mathcal{X}, \star, E_0)$ to represent a CEGAwT. For the sake of convenience, for any element $x' = h \star E_0 \in \mathcal{X}$, we will use the shorthand $x'^{-1} := h^{-1} \star E_0$. The notation is well-defined since for any $h_1 \star E_0 = h_2 \star E_0$, we have $h_1^{-1} \star E_0 = h_2^{-1} \star E_0$.

2.2 Lattices

We give a brief overview of the lattice foundations required to describe Blaze+ and BlindOR [AEB20a, AEB20b, AHJ21]. For readability, we simplify notation and focus on the essential parts that enable us to describe our attack. For a detailed overview, we refer to [AEB20a, AEB20b, AHJ21]. For any positive integer n, consider the polynomial ring $R_q = \mathbb{Z}_q[X]/(X^n + 1)$. Generally, we assume that n is a power of two. We consider the lattice R_q^m, where $m \in \mathbb{N}$.

Let $D_{\mathbb{Z}^n,\sigma,\mathbf{c}}$ be the discrete Gaussian distribution over $\mathbb{Z}^n$ with standard deviation $\sigma > 0$ and center $\mathbf{c}$. Throughout, we set $\chi = D_{\mathbb{Z}^n,\sigma,\mathbf{0}}$, for some implicit $\sigma > 0$, and $\chi_{\mathsf{rs}} = D_{\mathbb{Z}^n,s^*,\mathbf{0}}$, for some implicit $s^* > \sigma$. We denote by $\mathsf{RejSamp}$ an algorithm that carries out rejection sampling (with implicit parameters).

2.3 Blind Signature

Below, we recall the standard definition of (three-move) blind signatures.

Definition 2.2 (Blind Signature Scheme). *A three-move* partially blind signature $\mathsf{BS} = (\mathsf{BS.KGen}, \mathsf{BS.S}, \mathsf{BS.U}, \mathsf{BS.Verify})$ *with an efficiently decidable public key space $\mathcal{PK}$ consists of the following PPT algorithms:*

$\mathsf{BS.KGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: *On input the security parameter 1^λ, the key generation algorithm outputs a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk})$.*

$\mathsf{BS.S} = (\mathsf{BS.S}_1, \mathsf{BS.S}_2)$: *The interactive signer algorithm consists of two phases:*

- $\mathsf{BS.S}_1(\mathsf{sk}) \to (\mathsf{st}_\mathsf{S}, \rho_{\mathsf{S},1})$: *On input a secret key sk, it outputs an internal signer state st_S and a first-sender message $\rho_{\mathsf{S},1}$.*[2]
- $\mathsf{BS.S}_2(\mathsf{st}_\mathsf{S}, \rho_\mathsf{U}) \to \rho_{\mathsf{S},2}$: *On input a signer state st_S and a user message ρ_U, it outputs a second-sender message $\rho_{\mathsf{S},2}$.*

$\mathsf{BS.U} = (\mathsf{BS.U}_1, \mathsf{BS.U}_2)$: *The interactive user algorithm consists of two phases:*

- $\mathsf{BS.U}_1(\mathsf{pk}, \mathsf{M}, \rho_{\mathsf{S},1}) \to (\mathsf{st}_\mathsf{U}, \rho_\mathsf{U})$: *On input a public key $\mathsf{pk} \in \mathcal{PK}$, a message M, and a first-sender message $\rho_{\mathsf{S},1}$, it outputs an internal user state st_U and a user message ρ_U.*
- $\mathsf{BS.U}_2(\mathsf{st}_\mathsf{U}, \rho_{\mathsf{S},2}) \to \sigma$: *On input a user state st_U and a second-signer message $\rho_{\mathsf{S},2}$, it outputs a signature σ.*

$\mathsf{BS.Verify}(\mathsf{pk}, \mathsf{M}, \sigma) \to 1$ **or** 0 : *In input a public key pk, a message M, and a signature σ, the verification algorithm outputs 1 to indicate the signature is valid, and 0 otherwise.*

We define *correctness, blindness, and one-more unforgeability* of a blind signature scheme.

Perfect correctness ensures that when both the signer and the user adhere to the protocol specifications, the user's obtained signature will verify correctly. In the lattice-based setting (e.g. [AEB20b, AHJ21, HKLN20]), it is often acceptable to relax this requirement slightly, allowing for a scenario where the verification algorithm accepts with an overwhelming probability. Furthermore, it is acceptable that the signer aborts the protocol in advance.

The blindness property ensures that the signer cannot establish a link between the signing process and obtained signatures. This property preserves the privacy of the user's messages. As we are interested in forgery attacks, this notion is not important in our context and we omit details.

In this work, we focus on the one-more unforgeability (OMUF) notion of a blind signature. OMUF roughly ensures that at most one valid signature is generated after each full completion of a signing interaction. Formally, we have the following.

[2] We assume without loss of generality that sk includes pk and st_S includes $(\mathsf{pk}, \mathsf{sk})$ and omit it when the context is clear. Below, we also assume that st_U includes M.

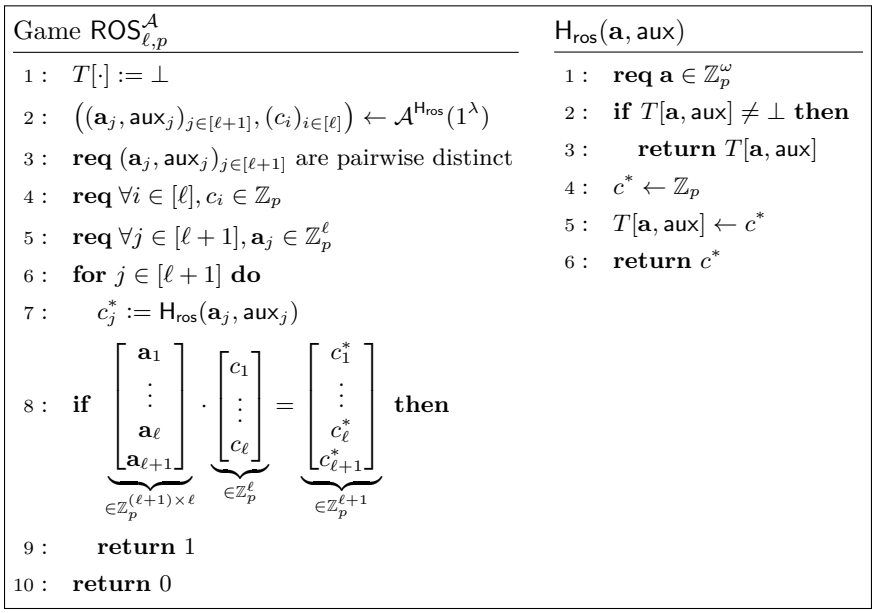

Figure 1: Classical ROS problem over $\mathbb{Z}_p$. **req** returns 0 if the requirement does not hold.

Definition 2.3 (One-More-Unforgeability). *We define ℓ-one-more unforgeability (ℓ-OMUF) for any $\ell \in \mathbb{N}$ of a three-move partially blind signature scheme BS via the following game between a challenger and an adversary $\mathcal{A}$:*

Setup. *The challenger samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{BS.KGen}(1^\lambda)$ and runs $\mathcal{A}$ on input pk. It further initializes $\ell_{\mathsf{closed}} = 0$ and $\mathsf{opened}_{\mathsf{sid}} = \mathtt{false}$ for all $\mathsf{sid} \in \mathbb{N}$.*

Online Phase. *$\mathcal{A}$ is given access to oracles S_1 and S_2, which behave as follows.*

- **Oracle S_1:** *The oracle samples a fresh session identifier sid. It sets $\mathsf{opened}_{\mathsf{sid}} \leftarrow$ true and generates $(\mathsf{st}_{\mathsf{S},\mathsf{sid}}, \rho_{\mathsf{S},1}) \leftarrow \mathsf{BS.S}_1(\mathsf{sk})$. Then it returns sid and the first-sender message $\rho_{\mathsf{S},1}$ to $\mathcal{A}$.*
- **Oracle S_2:** *On input of a user message ρ_{U} and a session identifier sid, if $\ell_{\mathsf{closed}} \geq \ell$ or $\mathsf{opened}_{\mathsf{sid}} = \mathtt{false}$, then it returns $\bot$. Otherwise, it sets $\mathsf{opened}_{\mathsf{sid}} = \mathtt{false}$. It then computes the second-signer message $\rho_{\mathsf{S},2} \leftarrow \mathsf{BS.S}_2(\mathsf{st}_{\mathsf{S},\mathsf{sid}}, \rho_{\mathsf{U}})$. If $\mathsf{BS.S}_2$ did not abort, then increments ℓ_{closed} and returns $\rho_{\mathsf{S},2}$ to $\mathcal{A}$. Else, returns $\bot$.*

Output Determination. *When $\mathcal{A}$ outputs distinct tuples $(\mathsf{M}_1, \sigma_1), \ldots, (\mathsf{M}_k, \sigma_k)$, we say $\mathcal{A}$* wins *if $k \geq \ell_{\mathsf{closed}} + 1$ and for all $i \in [k]$, $\mathsf{BS.Verify}(\mathsf{pk}, \mathsf{M}_i, \sigma_i) = 1$.*

We say BS is ℓ-one-more unforgeable *(or ℓ-OMUF) if the advantage of $\mathcal{A}$ defined as $\mathsf{Adv}^{\mathsf{OMUF}}_{\mathcal{A}}(\lambda) := \Pr[\mathcal{A}$ wins$]$ is negligible.*

2.4 ROS problem

The ROS problem captures a concurrent and algebraic attack on one-more unforgeability of the classical Schnorr blind signature [Sch01]. That is, if an adversary $\mathcal{A}$ can solve the $\mathsf{ROS}_{\ell,p}$ problem, then it can also break one-more unforgeability of the Schnorr blind signature. Here, the parameter p is the order of the underlying group and ℓ is the number of concurrent signing sessions. The problem is defined in Figure 1. Later, we define natural extensions of ROS for parallel repetitions.

3 Parallel ROS

We define two natural variants for the ROS problem for blind signatures based on parallel repetitions. The first variant Ring-pROS generalizes the standard ROS problem: the underlying algebraic structure is a ring $\mathcal{R}$ and we consider ω parallel repetitions. If we set $\mathcal{R} = \mathbb{Z}_p$ and $\omega = 1$, we obtain the classical ROS problem. The second variant Group-pROS is a harder variant of Ring-pROS: the underlying algebraic structure is a (multiplicative) group $\mathcal{G}$. Over groups, we have a single operation and thus, the adversary is much more restricted in its output (e.g., it cannot output arbitrary linear combinations).

Compared to ROS with a single repetition (i.e., $\omega = 1$), more parallel repetitions $\omega > 1$ require that we solve an overdetermined linear system with ω additional rows.

Perhaps surprisingly, we give two attacks on Ring-pROS and Group-pROS (with mild requirements). The first attack applies to both ROS variants and leverages that schemes with parallel repetitions often have a challenge space of polynomial size. The second attack generalizes the ROS attack from [BLL+21] and applies even for exponential challenge space. In Section 4, we apply our attacks to concrete schemes.

3.1 Definition of Parallel ROS

We define two problems Ring-pROS and Group-pROS.

3.1.1 For Rings

Let $(\mathcal{R}, +, \cdot)$ be a ring.

Parallel ROS Problem for Rings. We define a ROS problem for rings with parallel repetitions. Let $\ell \in \mathbb{N}$ be the number of concurrent sessions and $\omega \in \mathbb{N}$ be the number of parallel repetitions. Let $\mathcal{R}_c \subseteq \mathcal{R}$ be the challenge space. For any adversary $\mathcal{A}$ with oracle access to $\mathsf{H}_{\mathsf{ros}}$, the problem $\mathsf{Ring\text{-}pROS}^{\mathcal{A}}_{\ell,\omega,\mathcal{R},\mathcal{R}_c}$ is defined in Figure 2. We define the advantage of an adversary $\mathcal{A}$ against Ring-pROS as

$$\mathsf{Adv}^{\mathsf{Ring\text{-}pROS}}_{\mathcal{A},\ell,\omega,\mathcal{R},\mathcal{R}_c} := \Pr\left[b \leftarrow \mathsf{Ring\text{-}pROS}^{\mathcal{A}}_{\ell,\omega,\mathcal{R},\mathcal{R}_c}(\lambda) : b = 1\right].$$

If the parameters are clear by context, we sometimes write Ring-pROS for short.

Remark 3.1. Note that for $\omega = 1, \mathcal{R} = \mathcal{R}_c = \mathbb{Z}_p$, this is the standard ROS problem (cf. Figure 1).

3.1.2 For Group Actions

Let $(\mathcal{G}, \cdot)$ be a group.

Parallel ROS Problem for Group Actions. We define a ROS problem for groups with parallel repetitions. Let $\ell \in \mathbb{N}$ be the number of concurrent sessions and $\omega \in \mathbb{N}$ be the number of parallel repetitions. Let $\mathcal{G}_c \subseteq \mathcal{G}$ be the challenge space. For any adversary $\mathcal{A}$ with oracle access to $\mathsf{H}_{\mathsf{ros}}$, the problem $\mathsf{Group\text{-}pROS}^{\mathcal{A}}_{\ell,\omega,\mathcal{G},\mathcal{G}_c}$ is defined in Figure 3. We define the advantage of an adversary $\mathcal{A}$ against Group-pROS as

$$\mathsf{Adv}^{\mathsf{Group\text{-}pROS}}_{\mathcal{A},\ell,\omega,\mathcal{G},\mathcal{G}_c} := \Pr\left[b \leftarrow \mathsf{Group\text{-}pROS}^{\mathcal{A}}_{\ell,\omega,\mathcal{G},\mathcal{G}_c}(\lambda) : b = 1\right].$$

If the parameters are clear by context, we sometimes write Group-pROS for short.

3.2 Breaking Parallel ROS for Small Challenge Space

We provide an attack on Group-pROS with $\ell := \omega$ concurrent sessions if the challenge space is small, i.e., $|\mathcal{G}_c| = \mathsf{poly}(\lambda)$. Note that this also implies an attack on Ring-pROS (as the problem is more general) with the same parameters and complexity.

Overview. Our main observation is that since the challenge space is small, we can enforce specific values in a single coordinate of $\vec{c}_i^*$ by trying different auxiliary values. Roughly, we set up the first ℓ matrices $\mathbf{A}_j$

Game $\mathsf{Ring\text{-}pROS}^{\mathcal{A}}_{\ell,\omega,\mathcal{R},\mathcal{R}_c}(\lambda)$

$T[\cdot] := \bot$

$\left((\mathbf{A}_j, \mathsf{aux}_j)_{j\in[\ell+1]}, (\vec{c}_i)_{i\in[\ell]}\right) \leftarrow \mathcal{A}^{\mathsf{H}_{\mathsf{ros}}}(1^\lambda)$

req $(\mathbf{A}_j, \mathsf{aux}_j)_{j\in[\ell+1]}$ are pairwise distinct

req $\forall i \in [\ell], \vec{c}_i \in \mathcal{R}_c^\omega$

req $\forall j \in [\ell+1], \mathbf{A}_j \in \mathcal{R}^{\omega\times\omega\ell}$

for $j \in [\ell+1]$ **do**

  $\vec{c}_j^{\,*} := \mathsf{H}_{\mathsf{ros}}(\mathbf{A}_j, \mathsf{aux}_j)$

if $\underbrace{\begin{bmatrix}\mathbf{A}_1\\ \vdots \\ \mathbf{A}_\ell \\ \mathbf{A}_{\ell+1}\end{bmatrix}}_{\in\mathcal{R}^{\omega(\ell+1)\times\omega\ell}} \cdot \underbrace{\begin{bmatrix}\vec{c}_1\\ \vdots \\ \vec{c}_\ell\end{bmatrix}}_{\in\mathcal{R}_c^{\omega\ell}} = \underbrace{\begin{bmatrix}\vec{c}_1^{\,*}\\ \vdots \\ \vec{c}_\ell^{\,*} \\ \vec{c}_{\ell+1}^{\,*}\end{bmatrix}}_{\in\mathcal{R}^{\omega(\ell+1)}}$ **then**

  return 1

return 0

$\mathsf{H}_{\mathsf{ros}}(\mathbf{A}, \mathsf{aux})$

if $T[\mathbf{A}, \mathsf{aux}] = \bot$ **then**

  $\vec{c}^{\,*} \leftarrow \mathcal{R}_c^\omega$

  $T[\mathbf{A}, \mathsf{aux}] \leftarrow \vec{c}^{\,*}$

return $T[\mathbf{A}, \mathsf{aux}]$

Figure 2: Parallel ROS problem over ring $(\mathcal{R}, +, \cdot)$. In the above, **req** returns 0 if the requirement does not hold.

such that for each such matrix, we can find a trivial solution, i.e., the first $\omega\ell$ rows form an identity matrix (with $\bot$ symbols instead of zeros) and we pick the challenges as $\vec{c}_i = \vec{c}_i^{\,*}$. Next, we construct $\mathbf{A}_{\ell+1}$ with unit vectors as rows (with $\bot$ symbols instead of zeros) such that the i^{th} row picks a single coordinate from the i^{th} challenge vector $\vec{c}_i$. Then, we compute the challenge vector $\vec{c}^{\,*}_{\ell+1}$ and sample aux_i until each coordinate in $\vec{c}^{\,*}_{\ell+1}$ agrees with the picked coordinate in $\vec{c}_i$ for all $i \in [\omega]$. This constitutes a solution to the Group-pROS problem.

Finally, observe that we can replace all $\bot$ values with zeros to obtain a solution to the Ring-pROS problem.

The attack. First, for $i \in [\omega]$, we denote by $e_i^\bot := (\bot, \cdots, \bot, 1, \bot, \cdots \bot) \in (\mathcal{G}\cup\{\bot\})^k$ the vector of dimension $k \in \mathbb{N}$ with 1 at position i and all $\bot$ entries otherwise. Also, we write

$$\mathbf{I}_\omega^\bot := \begin{bmatrix} e_1^\bot \\ \vdots \\ e_\omega^\bot \end{bmatrix} \text{ and } \mathbf{I}_{\omega\ell}^\bot := \begin{bmatrix} \mathbf{I}_\omega^\bot & \cdots & \bot \\ \vdots & \ddots & \vdots \\ \bot & \cdots & \mathbf{I}_\omega^\bot \end{bmatrix}$$

for identity matrices of dimension ω and $\omega\ell$, respectively, where zeros are replaced with $\bot$. Above, $e_i^\bot$ is of dimension ω. Next, set for $j \in [\ell]$ and $e_1^\bot$ of dimension ℓ

$$\mathbf{A}_j = \left[\bot \mid \cdots \mid \mathbf{I}_\omega^\bot \mid \cdots \mid \bot\right], \mathbf{A}_{\ell+1} = \begin{bmatrix} e_1^\bot & \cdots & \bot \\ \vdots & \ddots & \vdots \\ \bot & \cdots & e_1^\bot \end{bmatrix} \in (\mathcal{G} \cup \{\bot\})^{\omega\times\omega\ell}.$$

Note that in the definition of $\mathbf{A}_j$, the matrix $\mathbf{I}_\omega^\bot$ is placed at columns $(j-1)\omega + 1$ to $j\omega$. Set $\vec{c}^{\,*}_{\ell+1} = \mathsf{H}_{\mathsf{ros}}(\mathbf{A}_{\ell+1}, \mathsf{aux}_{\ell+1})$ for some arbitrary $\mathsf{aux}_{\ell+1}$. Then, for $i \in [\ell]$, sample aux_i until we have that $c^*_{i,1} = c^*_{\ell+1,i}$, where $\vec{c}_i^{\,*} = \mathsf{H}_{\mathsf{ros}}(\mathbf{A}_i, \mathsf{aux}_i)$. Since the size of $\mathcal{G}_c$ is polynomial, this sampling is efficient. Then, we have for $\vec{d} := ((\vec{c}_1^{\,*})^\top, \cdots, (\vec{c}_\ell^{\,*})^\top)^\top$ that

$$\mathbf{A}_{\ell+1} \cdot \vec{d} = \begin{bmatrix} e_1^\bot & \cdots & \bot \\ \vdots & \ddots & \vdots \\ \bot & \cdots & e_1^\bot \end{bmatrix} \cdot \begin{bmatrix} \vec{c}_1^{\,*} \\ \vdots \\ \vec{c}_\ell^{\,*} \end{bmatrix} = \begin{bmatrix} c^*_{1,1} \\ \vdots \\ c^*_{\ell,1} \end{bmatrix} = \vec{c}^{\,*}_{\ell+1} \tag{1}$$

Game $\mathsf{Group\text{-}pROS}^{\mathcal{A}}_{\ell,\omega,\mathcal{G},\mathcal{G}_c}(\lambda)$

$T[\cdot] := \bot$

$\left((\mathbf{A}_j, \mathsf{aux}_j)_{j\in[\ell+1]}, (\vec{c}_i)_{i\in[\ell]}\right) \leftarrow \mathcal{A}^{\mathsf{H}_{\mathsf{ros}}}(1^\lambda)$

req $(\mathbf{A}_j, \mathsf{aux}_j)_{j\in[\ell+1]}$ are pairwise distinct

req $\forall i \in [\ell], \vec{c}_i \in \mathcal{G}_c^\omega$

req $\forall j \in [\ell+1], \mathbf{A}_j \in (\mathcal{G} \cup \{\bot\})^{\omega\times\omega\ell}$

for $j \in [\ell+1]$ **do**

  $\vec{c}_j^* := \mathsf{H}_{\mathsf{ros}}(\mathbf{A}_j, \mathsf{aux}_j)$

  for $k \in [\ell]$ **do**

  // check k-th row of $\mathbf{A}_j$ has exactly one non-$\bot$ entry

    if $\|\mathbf{A}_j[k]\|_\infty \neq 1$ **then**

      return 0

if $\underbrace{\begin{bmatrix}\mathbf{A}_1\\ \vdots \\ \mathbf{A}_\ell \\ \mathbf{A}_{\ell+1}\end{bmatrix}}_{\in(\mathcal{G}\cup\{\bot\})^{\omega(\ell+1)\times\omega\ell}} \cdot \underbrace{\begin{bmatrix}\vec{c}_1\\ \vdots \\ \vec{c}_\ell\end{bmatrix}}_{\in\mathcal{G}_c^{\omega\ell}} = \underbrace{\begin{bmatrix}\vec{c}_1^*\\ \vdots \\ \vec{c}_\ell^* \\ \vec{c}_{\ell+1}^*\end{bmatrix}}_{\in\mathcal{G}^{\omega(\ell+1)}}$ **then**

  return 1

return 0

$\mathsf{H}_{\mathsf{ros}}(\mathbf{A}, \mathsf{aux})$

if $T[\mathbf{A}, \mathsf{aux}] = \bot$ **then**

  $\vec{c}^* \leftarrow \mathcal{G}_c^\omega$

  $T[\mathbf{A}, \mathsf{aux}] \leftarrow \vec{c}^*$

return $T[\mathbf{A}, \mathsf{aux}]$

Figure 3: Parallel ROS problem over a group $(\mathcal{G}, \cdot)$. In the above, **req** returns 0 if the requirement does not hold. The main difference between the parallel ROS problem over a ring $\mathcal{R}$ is highlighted. Recall we extend the operation $\cdot$ to vectors with the understanding that we ignore the $\bot \notin \mathcal{G}$ entries.

Notably, the values $(\mathbf{A}_j, \mathsf{aux}_j)_{j\in[\ell+1]}$ and $(\vec{c}_i^*)_{i\in[\ell]}$ form a valid $\mathsf{Group\text{-}pROS}$ solution since

$$\begin{bmatrix}\mathbf{A}_1\\ \vdots \\ \mathbf{A}_\ell \\ \mathbf{A}_{\ell+1}\end{bmatrix} \cdot \begin{bmatrix}\vec{c}_1^*\\ \vdots \\ \vec{c}_\ell^*\end{bmatrix} = \begin{bmatrix}\mathbf{I}^\bot_{\omega\ell}\\ \mathbf{A}_{\ell+1}\end{bmatrix} \cdot \vec{d} = \begin{bmatrix}\vec{d}\\ \mathbf{A}_{\ell+1}\cdot\vec{d}\end{bmatrix}$$

$$\overset{(1)}{=} \begin{bmatrix}\vec{d}\\ \vec{c}_{\ell+1}^*\end{bmatrix} = \begin{bmatrix}\vec{c}_1^*\\ \vdots \\ \vec{c}_\ell^* \\ \vec{c}_{\ell+1}^*\end{bmatrix}$$

Complexity of the attack. Denote by $\xi = |\mathcal{G}_c|$ the size of the challenge space. For ω repetitions, the attack requires ω concurrent sessions. Also, for some fixed $\vec{c}_{\ell+1}^*$, the probability that $\vec{c}_{i,1}^* = \vec{c}_{\ell+1,i}^*$ is $1/\xi$, where $\vec{c}_i^*$ is sampled as in the attack. Since we require that this holds for all ω coordinates (i.e., for all $i \in [\omega]$), the attack requires $\mathcal{O}(\omega\xi)$ hash evaluations in expectation. Our attack has success probability 1.

In particular, the attack is PPT if $\omega = \mathsf{poly}(\lambda)$ and $\xi = \mathsf{poly}(\lambda)$. We refer to Sections 4.1 and 4.2 for an application of our attack on schemes from the literature.

Remark 3.2 (Guessing more coordiantes at once). Our attack can be generalized in order to obtain a tradeoff between the number of required hash evaluations and the number of concurrent sessions. Let $N \in \mathbb{N}$. Recall that we sample $\vec{c}_i^*$ in such a way that $\vec{c}_{\ell+1,i}^* = \vec{c}_{i,1}^*$. We do this for all $i \in [\omega]$ in order to cover all coordinates of

$\vec{c}^*_{\ell+1}$. The idea is to target N coordinates of $\vec{c}^*_{\ell+1}$ at the same time. That is, we sample auxiliary values until the first N coordinates in $\vec{c}^*_i$ agree with N coordinates in $\vec{c}^*_{\ell+1}$ until all coordinates of $\vec{c}^*_{\ell+1}$ are covered. The probability that N coordinates agree is $\frac{1}{\xi^N}$ and we require $\lceil \omega/N \rceil$ concurrent sessions to cover all coordinates of $\vec{c}^*_{\ell+1}$. The modifications are straightforward and we omit details. In total, we require only $\ell = \lceil \omega/N \rceil$ concurrent sessions and $\mathcal{O}(\ell \cdot \xi^N)$ hash evaluations. The success probability remains 1.

While the idea in Remark 3.2 is simple, it is powerful when applied to concrete schemes. For example, for $\omega = 128$ repetitions and challenge space $\{\pm 1\}$, set $N = 4$. Then, we require only $\ell = 4$ concurrent sessions for an attack that runs in 2^{34} time in expectation.

From the above discussions, we conclude the following statement.

Theorem 3.3. *Let $N \in \mathbb{N}$ and $\ell = \lceil \omega/N \rceil$. When $|\mathcal{G}_c| = \xi = \mathsf{poly}(\lambda)$ and $\omega = \mathsf{poly}(\lambda)$, there exists an PPT adversary against* Group-pROS*, parameterized by $(\ell, \omega, \mathcal{G}, \mathcal{G}_c)$, that makes $\mathcal{O}(\ell \xi^N)$ queries to the hash function in expectation, achieving an advantage of* 1*.*

3.3 Breaking Parallel ROS for Large Challenge Space

We provide an attack on Ring-pROS under the condition that non-zero challenge differences are invertible, i.e.,

$$(\mathcal{R}_c - \mathcal{R}_c) \setminus \{0\} \subseteq \mathcal{R}^\times. \tag{2}$$

Also, we assume that the size of the challenge space $|\mathcal{R}_c| = \Omega(2^\lambda)$ is exponential. In this case, the attack in Section 3.2 is infeasible [3]. Let $(B_\mu)_{\mu \in [\xi]}$ be a *binary generating set* for $\mathcal{R}$ of size ξ. That is, we assume that we can represent all elements $y \in \mathcal{R}$ as $y = \sum_{\mu \in [\xi]} B_\mu \cdot b_\mu$, where $b_\mu \in \{0, 1\}$. For example for $\mathcal{R} = \mathbb{Z}_p$, a good choice is $B_\mu = 2^{\mu - 1}$ with $\xi = \lceil \log\, p \rceil$. We give an attack on the Ring-pROS problem with $\ell = \omega\xi$ concurrent sessions.

Overview. For a single parallel repetition (i.e., $\omega = 1$), the attack is identical to the ROS attack [BLL+21], albeit some minor changes and generalizations. If $\omega > 1$, we perform ω independent ROS attacks (using ξ concurrent sessions per attack) each targeting one coordinate of the target vector $\vec{c}^*_{\ell+1}$. We sketch the attack below.

First, we proceed as in Section 3.2. That is, we pick the the first ℓ matrices $\mathbf{A}_i$ such that the first $\omega\ell$ rows form an identity matrix. Also, we pick the challenges as $\vec{c}_i = \vec{c}^*_i$. This yields ℓ trivial solutions for any choice of aux_i.

Then, we conceptually split the $\ell = \omega\xi$ concurrent sessions into ω many sets of ξ concurrent sessions. For each such set of ξ sessions, we choose two different auxiliary values aux_i^b per session which define two challenge vectors $\vec{c}_i^{\,b}$, for $b \in \{0, 1\}$. Then, we compute a linear polynomial f that allows us to express any element in $\mathcal{R}$, depending on the choice of the auxiliary values aux_i^b (within the current set of ξ sessions) using the binary generating set $(B_\mu)_{\mu \in [\xi]}$. This step also requires polynomial interpolation (which in turn requires Equation (2)).

We do this for all ω sets of ξ sessions and obtain ω polynomials $(f_\kappa)_{\kappa \in [\omega]}$. Using the fact that each polynomial is linear, we embed it in into the last matrix $\mathbf{A}_{\ell+1}$ such that each polynomial f_κ allows us to express one coordinate of the challenge vector $\vec{c}^*_{\ell+1}$ (via the binary choice of either aux_i^0 or aux_i^1 per session). After choosing the right aux_i^0 or aux_i^1 according to a binary decomposition with respect to $(B_\mu)_{\mu \in [\xi]}$, this yields a solution to the Ring-pROS problem.

The attack. First, we define ℓ matrices $\mathbf{A}_j$ such that we can find trivial solutions for ℓ concurrent sessions. That is, we define for $j \in [\ell]$ the matrix

$$\mathbf{A}_j = \left[0 \mid \cdots \mid \mathbf{I}_\omega \mid \cdots \mid 0\right] \in \mathcal{R}^{\omega \times \omega\ell},$$

[3] An exponential challenge space is not required for the attack but it simplifies the description (cf. Remark 3.4).

where the identity matrix $\mathbf{I}_\omega$ of dimension ω is placed at columns $(j-1)\omega+1$ to $j\omega$ and the remaining values are zeros. Set $\mathsf{aux}_j^0 \neq \mathsf{aux}_j^1$ arbitrarily, say $\mathsf{aux}_j^b := b$ for $b \in \{0,1\}$, and define $\vec{c}_j^{\,b} = \mathsf{H}_{\mathsf{ros}}(\mathbf{A}_j, \mathsf{aux}_j^b)$. For some $\vec{b} \in \{0,1\}^\ell$, let us denote by

$$\vec{d}_{\vec{b}} = ((\vec{c}_1^{\,b_1})^\top, \cdots, (\vec{c}_\ell^{\,b_\ell})^\top)^\top \tag{3}$$

the vector in which we choose either $\vec{c}_j^{\,0}$ or $\vec{c}_j^{\,1}$ depending on b_j. Then, for any choice of $\mathbf{A}_{\ell+1}$ and $\vec{b}$, we have that

$$\begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_\ell \\ \mathbf{A}_{\ell+1} \end{bmatrix} \cdot \begin{bmatrix} \vec{c}_1^{\,b_1} \\ \vdots \\ \vec{c}_\ell^{\,b_\ell} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{\omega\ell} \\ \mathbf{A}_{\ell+1} \end{bmatrix} \cdot \vec{d}_{\vec{b}} = \begin{bmatrix} \vec{d}_{\vec{b}} \\ \mathbf{A}_{\ell+1} \cdot \vec{d}_{\vec{b}} \end{bmatrix} = \begin{bmatrix} \vec{c}_1^{\,b_1} \\ \vdots \\ \vec{c}_\ell^{\,b_\ell} \\ \mathbf{A}_{\ell+1} \cdot \vec{d}_{\vec{b}} \end{bmatrix}.$$

If we further have for some choice of $\vec{b}$ and $\mathsf{aux}_{\ell+1}$ that

$$\mathbf{A}_{\ell+1} \cdot \vec{d}_{\vec{b}} = \vec{c}_{\ell+1}^{\,*} = \mathsf{H}_{\mathsf{ros}}(\mathbf{A}_{\ell+1}, \mathsf{aux}_{\ell+1}), \tag{4}$$

then the values $(\mathbf{A}_j, \mathsf{aux}_j)_{j\in[\ell+1]}$ and $(\vec{c}_i)_{i\in[\ell]}$ form a valid Ring-pROS solution, where $\mathsf{aux}_i := \mathsf{aux}_i^{b_i}$ and $\vec{c}_i := \vec{c}_i^{\,b_i}$ for $i \in [\ell]$. Let $\mathsf{aux}_{\ell+1}$ be arbitrary. To conclude the attack, we define $\mathbf{A}_{\ell+1}$ and $\vec{b}$ such that Equation (4) is satisfied.

Recall that $\ell = \omega\xi$. Let $\kappa \in [\omega]$. We construct some polynomial f_κ which depends on the challenges from the κ^{th} set of ξ concurrent sessions, i.e., the challenges $\left\{\vec{c}^{\,b}_{(\kappa-1)\xi+1}, \cdots, \vec{c}^{\,b}_{\kappa\xi} \mid b \in \{0,1\}\right\}$. For this, denote by $\vec{\gamma}_\kappa^{\,b} := \left(c^b_{(\kappa-1)\xi+1,1}, \cdots, c^b_{\kappa\xi,1}\right)^\top \in \mathcal{R}_c^\xi$ the vector which consists of the first coordinate of each such challenge for $b \in \{0,1\}$. Let $f_{\kappa,\mu} := \frac{X_\mu - \gamma^0_{\kappa,\mu}}{\gamma^1_{\kappa,\mu} - \gamma^0_{\kappa,\mu}} \in \mathcal{R}[X_\mu]$ for $\mu \in [\xi]$. Note that here, we require that $\gamma^1_{\kappa,\mu} - \gamma^0_{\kappa,\mu} \in \mathcal{R}^\times$ which holds by assumption (cf. Equation (2)), unless $\gamma^1_{\kappa,\mu} = \gamma^0_{\kappa,\mu}$. The latter happens with negligible probability because we assume that the challenge space is exponential [4]. By construction, we have $f_{\kappa,\mu}(\gamma^b_{\kappa,\mu}) = b$. We now define the linear polynomial f_κ and parse its coefficients (without the constant term t_κ) in the vector $\vec{a}_\kappa$ as

$$f_\kappa := \sum_{\mu\in[\xi]} B_\mu f_{\kappa,\mu} = t_\kappa + \sum_{\mu\in[\xi]} a_{\kappa,\mu} X_\mu \in \mathcal{R}[X_1, \ldots, X_\xi].$$

Let $\vec{b}_\kappa \in \{0,1\}^\xi$ be arbitrary. (We choose concrete bit vectors later.) We denote by $\vec{\gamma}_{\vec{b}_\kappa} := \left(\gamma^{b_{\kappa,1}}_{\kappa,1}, \cdots, \gamma^{b_{\kappa,\xi}}_{\kappa,\xi}\right)$ the vector in which we choose either $\gamma^0_{\kappa,\mu}$ or $\gamma^1_{\kappa,\mu}$ dictated by $\vec{b}_\kappa$. Note that we have that

$$\begin{aligned} \left\langle \vec{a}_\kappa, \vec{\gamma}_{\vec{b}_\kappa} \right\rangle &= \sum_{\mu\in[\xi]} a_{\kappa,\mu} \cdot \gamma^{b_{\kappa,\mu}}_{\kappa,\mu} \\ &= f_\kappa\left(\gamma^{b_{\kappa,1}}_{\kappa,1}, \cdots, \gamma^{b_{\kappa,\xi}}_{\kappa,\xi}\right) - t_\kappa \\ &= \sum_{\mu\in[\xi]} B_\mu f_{\kappa,\mu}(\gamma^{b_{\kappa,\mu}}_{\kappa,\mu}) - t_\kappa \\ &= \sum_{\mu\in[\xi]} B_\mu b_{\kappa,\mu} - t_\kappa. \end{aligned} \tag{5}$$

Denote $\vec{b} = (\vec{b}_1, \cdots, \vec{b}_\omega)^\top$. Next, we embed the coefficient vector $\vec{a}_\kappa$ in the κ^{th} row of $\mathbf{A}_{\ell+1}$ in such a way that it lines up with $\vec{\gamma}_{\vec{b}_\kappa}$ (filling the other coordinates with zeros appropriately), i.e., such that

$$\mathbf{A}_{\ell+1}[\kappa] \cdot \vec{d}_{\vec{b}} = \left\langle \vec{a}_\kappa, \vec{\gamma}_{\vec{b}_\kappa} \right\rangle, \tag{6}$$

[4] Concretely, we have $\gamma^1_{\kappa,\mu} \neq \gamma^0_{\kappa,\mu}$ except with probability $1/|\mathcal{R}_c|$ since both values are distributed independently and uniformly at random in $\mathcal{R}_c$. Since we require this for all ℓ concurrent sessions, a union bound yields that we abort with probability at most $\ell/|\mathcal{R}_c|$. This probability is negligible for $\ell = \mathsf{poly}(\lambda)$ since we assume $|\mathcal{R}_c| = \Omega(2^\lambda)$. We refer to Remark 3.4 for a discussion on how to avoid aborts. For readability, we choose to abort here.

where $\vec{d}_{\vec{b}}$ is defined as in Equation (3). Recall that we choose the *first* coordinate of each challenge vector to define $\vec{\gamma}_{\vec{b}_\kappa}$, so we pad $\vec{a}_\kappa$ as follows. Let $\vec{a}^*_{\kappa,\mu} := (a_{\kappa,\mu}, 0, \cdots, 0) \in \mathcal{R}^\omega$ and $\vec{a}^*_\kappa := (\vec{a}^*_{\kappa,1}, \cdots, \vec{a}^*_{\kappa,\xi})^\top \in \mathcal{R}^{\omega\xi}$. Next, recall that each $\vec{\gamma}_{\vec{b}_\kappa}$ is constructed based on the challenges in the κ^{th} set of ξ concurrent sessions. Thus, to line up each coefficient, we set

$$\mathbf{A}_{\ell+1} := \begin{bmatrix} \vec{a}^*_1 & 0 & & \cdots & & 0 \\ 0 & \vec{a}^*_2 & 0 & \cdots & & 0 \\ \vdots & & & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \vec{a}^*_\omega \end{bmatrix} \in \mathcal{R}^{\omega \times \omega\ell}.$$

With this choice of $\mathbf{A}_{\ell+1}$, Equation (6) holds by construction.

We are now ready to choose $\vec{b}_\kappa$ concretely such that Equation (4) holds. Set $\vec{c}^*_{\ell+1} = \mathsf{H}_{\mathsf{ros}}(\mathbf{A}_{\ell+1}, \mathsf{aux}_{\ell+1})$. For $\kappa \in [\omega]$, set $y_\kappa := c^*_{\ell+1,\kappa} + t_\kappa$ and decompose $y_\kappa = \sum_{\mu\in[\xi]} B_\mu b_{\kappa,\mu}$ in binary. Set $\vec{b}_\kappa := (b_{\kappa,1}, \cdots, b_{\kappa,\xi})$. By construction, we have for $\kappa \in [\omega]$ that

$$\begin{aligned} \mathbf{A}_{\ell+1}[\kappa] \cdot \vec{d}_{\vec{b}} &\overset{(6)}{=} \left\langle \vec{a}_\kappa, \vec{\gamma}_{\vec{b}_\kappa} \right\rangle \\ &\overset{(5)}{=} \sum_{\mu\in[\xi]} B_\mu b_{\kappa,\mu} - t_\kappa \\ &= y_\kappa - t_\kappa = c^*_{\ell+1,\kappa}. \end{aligned}$$

Since this holds for all $\kappa \in [\omega]$, Equation (4) is satisfied and we found a solution to the Ring-pROS problem.

Complexity of the attack. The efficiency of the attack depends on the choice of the generating set. Given a binary generating set $(B_\mu)_{\mu\in[\xi]}$ of size ξ, the attack requires $\ell = \omega\xi$ concurrent sessions, where ω is the number of repetitions. Further, it requires $2\ell+1$ hash evaluations, $\mathcal{O}(\ell)$ ring operations and ω decompositions of some ring element in the basis $(B_\mu)_{\mu\in[\xi]}$. The attack succeeds except with probability $\mathcal{O}(\frac{\ell}{2^\lambda}) = \mathsf{negl}(\lambda)$.

Remark 3.4 (No aborts). We present the attack with negligible abort probability for readability, but this is not inherent. It is possible to adapt the attack to always succeed. Observe that we abort only if there is a collision in the first coordinates $\gamma^0_{\kappa,\mu}$ and $\gamma^1_{\kappa,\mu}$ of $\vec{c}^0_i$ and $\vec{c}^1_i$, respectively, for appropriate $i \in [\ell]$. A simple fix is to resample the challenges in this case (by choosing different aux^b_i for $b \in \{0,1\}$). A more direct approach is to instead setup the polynomials $f_{\kappa,\mu}$ using another coordinate of $\vec{c}^0_i$ and $\vec{c}^1_i$, where both challenge vectors have distinct values. Then, we have to change where to embed the coefficients in $\mathbf{A}_{\ell+1}$ but this is straightforward. In case both challenge vectors agree on all coordinates, we found a trivial collision (which yields a trivial solution).

We refer to Section 4.3 for an application of our attack with concrete parameters. The above considerations yield the following theorem.

Theorem 3.5. *Assume that* $(\mathcal{R}_c - \mathcal{R}_c) \setminus \{0\} \subseteq \mathcal{R}^\times$. *Let* $(B_\mu)_{\mu\in[\xi]}$ *be a binary generating set for* $\mathcal{R}$ *of size* ξ. *Assume that* $\omega = \mathsf{poly}(\lambda)$ *and* $\xi = \mathsf{poly}(\lambda)$. *Let* $\ell = \omega\xi$. *There exists an PPT adversary against* Ring-pROS, *parameterized by* $(\ell, \omega, \mathcal{R}, \mathcal{R}_c)$, *that makes at most* $2\ell+1$ *queries to the hash function, achieving an advantage of* 1. *In total, the adversary performs* $\mathcal{O}(\ell)$ *ring operations and* ω *decompositions of some ring element with respect to* $(B_\mu)_{\mu\in[\xi]}$.

4 Implications of Attack

In this section, we apply our attacks on parallel ROS from Section 3 to several concrete blind signatures from the literature. In particular, we use the attack in Section 3.2 to provide efficient attacks on CSI-Otter [KLLQ23], Blaze+ [AEB20a, AEB20b] and BlindOR [AHJ21]. Also, we give an attack on a conceptual variant of Schnorr with parallel repetitions to illustrate our attack in Section 3.3.

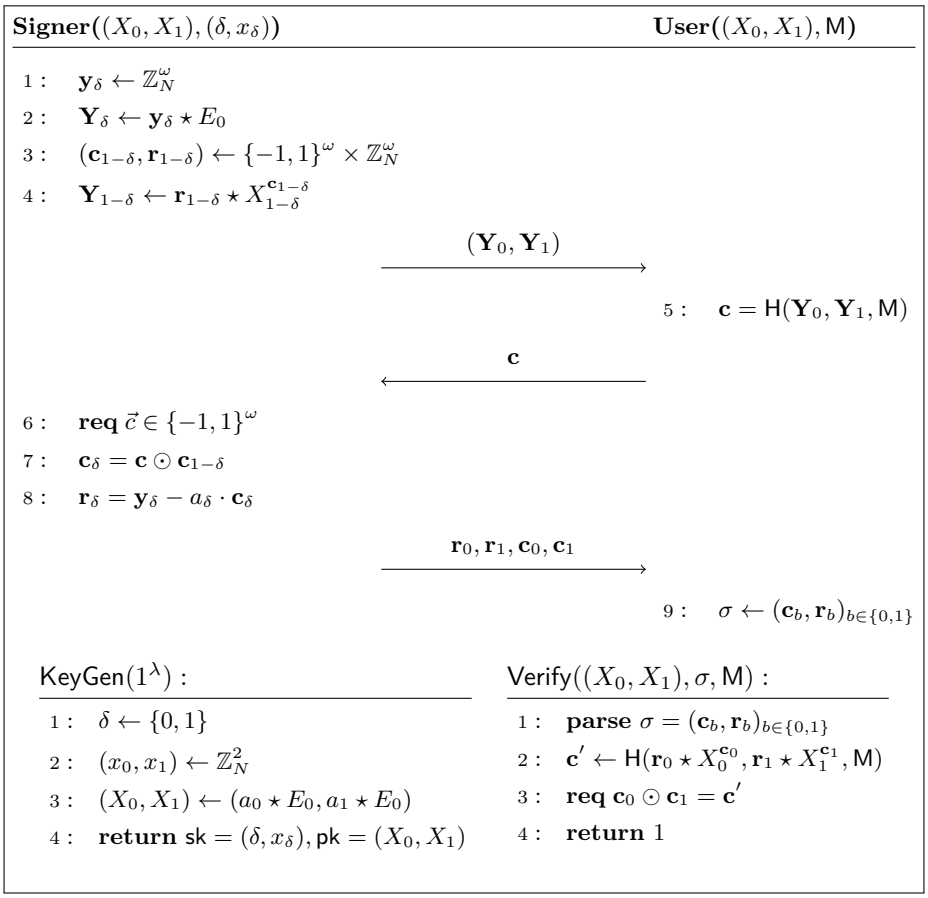

Figure 4: The unblinded signing protocol, key generation and verification of CSI-Otter, where $\mathsf{H} : \{0,1\}^* \to \{\pm 1\}^\omega$ is a collision-resistant hash function. Here, $\odot$ denotes the entry-wise multiplication. In the signing protocol (resp. verification), **req** aborts (resp. returns 0) if the requirement does not hold.

For each scheme, we first recall the (unblinded) interactive signing protocol. To improve readability, we only present the signer algorithms, key generation and verification, since these algorithms specify the OMUF game fully [5]. Then, we show that an adversary on Ring-pROS or Group-pROS (with appropriate parameters) allows to break OMUF of the scheme. Our attacks on Ring-pROS and Group-pROS from Section 3 yield concrete concurrent attacks on the scheme.

4.1 Isogeny-based Blind Signature: CSI-Otter

Let $(\mathbb{Z}_N, \mathcal{X}, \star, E_0)$ be a CEGAwT and $\mathsf{H} : \{0,1\}^* \to \{-1,1\}^\omega$ be a collision-resistant hash function. We recall the interactive signing protocol, key generation and verification of CSI-Otter [KLLQ23] in Fig. 4.

Our attack is summarized in the theorem below.

Theorem 4.1. *For any PPT adversary $\mathcal{A}$ on $\mathsf{Group\text{-}pROS}_{\ell,\omega,\{\pm1\},\{\pm1\}}$, there exists a PPT adversary $\mathcal{B}$ against ℓ-OMUF of CSI-Otter such that $\mathsf{Adv}^{\mathsf{OMUF}}_{\mathcal{B}}(\lambda) = \mathsf{Adv}^{\mathsf{Group\text{-}pROS}}_{\mathcal{A},\ell,\omega,\{\pm1\},\{\pm1\}}(\lambda)$ where $\{\pm1\}$ forms a group with the multiplication.*

[5] In particular, we omit the user's algorithms which specify how a signature can be blinded. Since we provide attacks, these are not relevant in our context.

Proof. Let $\mathcal{A}$ be an adversary on $\mathsf{Group\text{-}pROS}_{\ell,\omega,\{\pm1\},\{\pm1\}}$ for some $\ell \in \mathbb{N}$. The adversary $\mathcal{B}$ against ℓ-OMUF of $\mathsf{CSI\text{-}Otter}$ with access to $\mathcal{A}$ proceeds as follows.

Adversary $\mathcal{B}$. First, $\mathcal{B}$ invokes the ℓ-OMUF experiment and obtains a public key (X_0, X_1). Also, $\mathcal{B}$ obtains access to the hash function H and the signing oracles S_1 and S_2. Next, $\mathcal{B}$ opens ℓ concurrent (signing) sessions via the S_1 oracle and obtains $(\mathbf{Y}_{i,0}, \mathbf{Y}_{i,1})$ from the i^{th} session for each $i \in [\ell]$. Then, $\mathcal{B}$ sets up an empty table T and sets $\mathbf{Z}_b = (\mathbf{Y}_{1,b}^\top, \cdots, \mathbf{Y}_{\ell,b}^\top)^\top \in \mathbb{Z}_N^{\omega\ell}$ for $b \in \{0,1\}$.

Let Q be the number of $\mathsf{H}_{\mathsf{ros}}$ queries of $\mathcal{A}$ (including potential queries for verification of $\mathcal{A}$'s output). Assume without loss of generality that each $\mathsf{H}_{\mathsf{ros}}$ query is distinct. Adversary $\mathcal{B}$ answers the i^{th} query $(\mathbf{A}, \mathsf{aux})$ as follows. If $\mathbf{A} \notin (\{\pm1\} \cup \{\perp\})^{\omega\times\omega\ell}$ or $\|\mathbf{A}[k]\|_\infty \neq 1$ for some $k \in [\omega]$, returns $\vec{c}^* \leftarrow \{\pm1\}^\omega$. Otherwise, parses the k^{th} row of $\mathbf{A}$ as $(\perp, \cdots, \alpha_k, \cdots, \perp) = \mathbf{A}[k]$, where $\alpha_k \in \{\pm1\}$ is the non-$\perp$ entry in column ν_k. Computes

$$\mathbf{Y}_0^* = (\mathbf{Z}_{\nu_1}^{\alpha_1}, \cdots, \mathbf{Z}_{\nu_\omega}^{\alpha_\omega}) \qquad \text{and} \qquad \mathbf{Y}_1^* = (\mathbf{Z}_{\nu_1}, \cdots, \mathbf{Z}_{\nu_\omega}). \tag{7}$$

Note that $\mathbf{Y}_0^*, \mathbf{Y}_1^* \in \mathcal{X}^\omega$. Sets $\mathsf{M} = (\mathsf{aux}, i)$ and $\mathbf{c}^* = \mathsf{H}(\mathbf{Y}_0^*, \mathbf{Y}_1^*, \mathsf{M})$. Inserts $T[\mathbf{A}, \mathsf{aux}] = (\mathbf{Y}_0^*, \mathbf{Y}_1^*, \mathsf{M})$ into the table T and outputs $\mathbf{c}^*$. Note that by design, M is distinct for each distinct query.

After its query phase, $\mathcal{A}$ outputs $(\mathbf{A}_j, \mathsf{aux}_j)_{j\in[\ell+1]}$ and $(\mathbf{c}_i)_{i\in[\ell]}$. Adversary $\mathcal{B}$ checks if $\mathcal{A}$ succeeds; if not, outputs $\perp$ to the ℓ-OMUF game. Else, $\mathcal{B}$ closes the i^{th} session with $\mathbf{c}_i \in \{\pm1\}^\omega$ via the S_2 oracle and obtains $(\mathbf{r}_{i,0}, \mathbf{r}_{i,1}, \mathbf{c}_{i,0}, \mathbf{c}_{i,1})$ in response. Then, for $b \in \{0,1\}$, $\mathcal{B}$ sets $\mathbf{s}_b = (\mathbf{r}_{1,b}^\top, \cdots, \mathbf{r}_{\ell,b}^\top)$, $\mathbf{d}_b = (\mathbf{c}_{1,b}^\top, \cdots, \mathbf{c}_{\ell,b}^\top)^\top$. For each $j \in [\ell+1]$, parses $\mathbf{A}_j$ as above, i.e., for all rows $k \in [\omega]$, parses $(\perp, \cdots, \alpha_{j,k}, \cdots, \perp) = \mathbf{A}[k]$, where $\alpha_{j,k} \in \{\pm1\}$ is the non-$\perp$ entry in column $\nu_{j,k}$. To generate the forgeries, for $b \in \{0,1\}$, sets

$$\mathbf{r}_{j,b}^* = (\alpha_{j,1}^{1-b} \cdot s_{b,\nu_{j,1}}, \cdots, \alpha_{j,\omega}^{1-b} \cdot s_{b,\nu_{j,\omega}}), \mathbf{c}_{j,b}^* = (\alpha_{j,1}^{1-b} \cdot d_{b,\nu_{j,1}}, \cdots, \alpha_{j,\omega}^{1-b} \cdot d_{b,\nu_{j,\omega}}) \tag{8}$$

and parses $(\mathbf{Y}_{j,0}^*, \mathbf{Y}_{j,1}^*, \mathsf{M}_j) = T[\mathbf{A}_j, \mathsf{aux}_j]$. Finally, $\mathcal{B}$ sets $\sigma_j = (\mathbf{r}_{j,b}^*, \mathbf{c}_{j,b}^*)_{b\in\{0,1\}}$ and outputs the forgeries $(\sigma_j, \mathsf{M}_j)_{j\in[\ell+1]}$ to the ℓ-OMUF game.

Success probability. It is easy to see that $\mathcal{B}$ simulates $\mathsf{H}_{\mathsf{ros}}$ perfectly. Thus, it suffices to show that the signatures $(\sigma_j, \mathsf{M}_j)_{j\in[\ell+1]}$ are valid if $\mathcal{A}$'s output forms a valid $\mathsf{Group\text{-}pROS}$ solution. In that case, we have that $(\mathbf{A}_j, \mathsf{aux}_j)_{j\in[\ell+1]}$ are pairwise distinct, $\mathbf{c}_i \in \{\pm1\}^\omega$ and $\mathbf{A}_j \in (\{\pm1\} \cup \{\perp\})^{\omega\times\omega\ell}$ with $\|\mathbf{A}_j[k]\|_\infty = 1$ for all $k \in [\omega]$. By construction, we know that $(\mathsf{M}_j)_{j\in[\ell+1]}$ are pairwise distinct. Further, we know that for $j \in [\ell+1]$ it holds that $\mathbf{A}_j \cdot \mathbf{d} = \mathsf{H}_{\mathsf{ros}}(\mathbf{A}_j, \mathsf{aux}_j)$, where $\mathbf{d} = (\mathbf{c}_1^\top, \cdots, \mathbf{c}_\ell^\top)^\top$. With the above notation, this is equivalent to

$$(\alpha_{j,k} \cdot d_{\nu_{j,k}})_{k\in[\omega]} = \mathsf{H}(\mathbf{Y}_{j,0}^*, \mathbf{Y}_{j,1}^*, \mathsf{M}_j) := \mathbf{c}_j^* \tag{9}$$

by construction. For $i \in [\ell]$, we know that the signer's output satisfies $\mathbf{Y}_{i,0} = \mathbf{r}_{i,0} \star X_0^{\mathbf{c}_{i,0}}, \mathbf{Y}_{i,1} = \mathbf{r}_{i,1} \star X_1^{\mathbf{c}_{i,1}}$ and $\mathbf{c}_{i,0} \odot \mathbf{c}_{i,1} = \mathbf{c}_i$. With the above notation, this can be rewritten as

$$\mathbf{Z}_b = \mathbf{s}_b \star X_b^{\mathbf{d}_b} \qquad \text{and} \qquad \mathbf{d}_0 \odot \mathbf{d}_1 = \mathbf{d}. \tag{10}$$

We now show that $(\sigma_j, \mathsf{M}_j)_{j\in[\ell+1]}$ verify given the above. That is, $\mathbf{c}_{j,0}^* \odot \mathbf{c}_{j,1}^* = \mathbf{c}_j^*$ and that for $b \in \{0,1\}$, we have that $\mathbf{Y}_{j,b}^* = \mathbf{r}_{j,b}^* \star X_b^{\mathbf{c}_{j,b}^*}$. We verify that this holds for each coordinate. We have for $k \in [\omega]$ that

$$\begin{aligned} r_{j,b,k}^* \star X_b^{c_{j,b,k}^*} &\overset{(8)}{=} (\alpha_{j,k}^{1-b} \cdot s_{b,\nu_{j,k}}) \star X_b^{\alpha_{j,k}^{1-b} \cdot d_{b,\nu_{j,k}}} \\ &\overset{(10)}{=} Z_{b,\nu_{j,k}}^{\alpha_{j,k}^{1-b}} \\ &\overset{(7)}{=} Y_{j,b,k}^* \end{aligned}$$

and

$$c_{j,k}^* \overset{(9)}{=} \alpha_{j,k} \cdot d_{\nu_{j,k}}$$

$$\stackrel{(10)}{=} \alpha_{j,k} \cdot d_{0,\nu_{j,k}} \cdot d_{1,\nu_{j,k}}$$
$$\stackrel{(8)}{=} c^*_{j,0,k} \cdot c^*_{j,1,k}.$$

Thus, all signatures verify.

□

Remark 4.2. The above attack can be extended to encompass the general framework of CSI-Otter, wherein the groups $\mathcal{G}$ and $\mathcal{G}_c$, initially set as $\{\pm 1\}$, can be replaced by a cyclic group C_d of a higher order $d \in \mathbb{N}$. This substitution naturally corresponds the fact that the challenge space in CSI-Otter isomorphic to C_d as a group, where d corresponds to the d-th root of unity used within their generalized blind signature scheme.

Combining Theorem 4.1, Theorem 3.3, and the above remark, we obtain the following lemma.

Corollary 4.3. *Let $N \in \mathbb{N}$. Let the scheme* CSI-Otter *be parameterized by the challenge space $\{\pm 1\}$ with ω repetitions. Under this parameterization, there exists a PPT adversary against the ℓ-*OMUF *experiment of* CSI-Otter *for $\ell = \lceil \omega/N \rceil$ with expected $\mathcal{O}(\ell 2^N)$ queries to the hash function, achieving an advantage of* 1. *Furthermore, when ζ_d-*CSI-Otter *is parameterized by the challenge space $\langle \zeta_d \rangle$ of size d and ω repetitions, there exists a PPT adversary against the ℓ-*OMUF *experiment of ζ_d-*CSI-Otter *for $\ell = \lceil \omega/N \rceil$ with $\mathcal{O}(\ell d^N)$ queries to the hash functions, achieving an advantage of* 1.

Concretely, in CSI-Otter, ω is taken to be 128. With 128 concurrent sessions of CSI-Otter and expected 256 hash queries, we are able to break the 128-one-more unforgeability of it with an overwhelming probability. Also, with 4 concurrent sessions of CSI-Otter and expected 2^{34} hash queries, we are able to break 4-the one-more unforgeability of it with an overwhelming probability. In ζ_4-CSI-Otter, ω is taken to be 64. With 4 concurrent sessions of ζ_4-CSI-Otter, ω and expected 2^{34} hash queries, we are able to break the 4-one-more unforgeability of it with an overwhelming probability. Note that this does not contradict to the security proof of CSI-Otter where the reduction loss depends on the number of hash queries as given in [KLX22a].

4.2 Lattice-based Blind Signatures: Blaze+ and BlindOR

In this subsection, we consider the lattice-based blind signatures Blaze+ [AEB20a, AEB20b] and BlindOR [AHJ21]. Let $R_q = \mathbb{Z}_q[X]/(X^n + 1)$, where $n \in \mathbb{N}$ is a power of two. We first define the two subsets of R_q

$$\mathbb{T} = \{(-1)^b \cdot X^k \mid b \in \{0,1\}, k \in [0, n-1]\},$$
$$\mathcal{C} = \{c \in R_q \mid c = \sum_{i \in [\omega]} \hat{c}_i \text{ for } \hat{c}_i \in \mathbb{T}\}.$$

The challenge space of BlindOR is $\mathbb{T}^\omega$. The challenge space of Blaze+ is $\mathcal{C}$, but during signing, this space is decomposed into $\mathbb{T}^\omega$. For this, we define the mapping $\mathsf{Decomp} : \mathcal{C} \to \mathbb{T}^\omega$. Given $c \in \mathcal{C}$, Decomp outputs $(\hat{c}_1, \cdots, \hat{c}_\omega)$ such that $c = \sum_{i \in [\omega]} \hat{c}_i$.

Further, let $B_{\mathsf{kg}} < B_{\mathsf{ver}}$ be two real numbers. The value B_{kg} is a norm bound for the vectors in the secret key and B_{ver} is a norm bound for the vectors in a signature. Also, we denote the abort probability of rejection sampling by δ_{rej}. We also denote by δ_{ver} the probability that a signature from an honest *unblinded* signing interaction (of Blaze+ or BlindOR) passes verification.

4.2.1 Blaze+.

Here, we give a concrete attack on Blaze+. Let $\mathsf{H} : \{0,1\}^* \to \mathcal{C}$ be a collision-resistant hash function. Note that to generate the challenge c in Blaze+, the user evaluates the hash function $c \leftarrow \mathsf{H}(\mathsf{root}, \mathsf{M})$ on message M and a vector commitment root (i.e., a Merkle tree) to $w \in R_q$. Instead, we evaluate H on M and w to simplify presentation. The interactive signing protocol, key generation and verification is given in Figure 5. We stress that this simplification is purely for readability, and an attack on the scheme in Figure 5 immediately yields an attack on Blaze+ (by first committing to w in root).

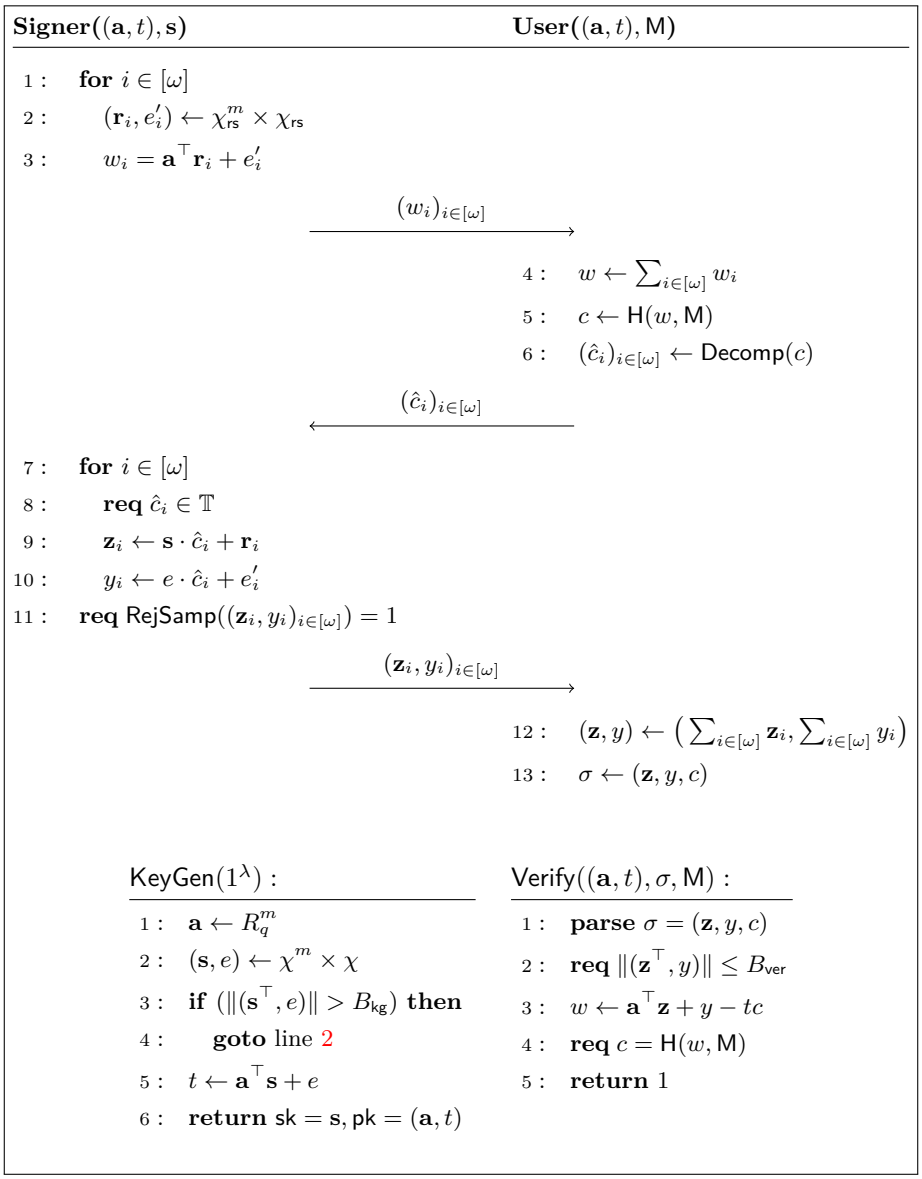

Figure 5: The unblinded signing protocol, key generation and verification Blaze+, where $\mathsf{H} : \{0,1\}^* \to \mathcal{C}$ is a collision-resistant hash function and $B_{\mathsf{kg}}, B_{\mathsf{ver}} \in R$ are fixed norm bounds. We also specify key generation and verification. In the signing protocol (resp. verification), **req** aborts (resp. returns 0) if the requirement does not hold.

Roughly, Blaze+ decomposes $c \in \mathcal{C}$ into ω monomials in $\mathbb{T}$ and runs ω parallel repetitions of a lattice-based identification protocol. For our attack, we observe that while the final signature in Blaze+ does not have explicit parallel repetitions, the signing protocol is parallelized. We can leverage this structure to attack the scheme. In particular, for $\ell \in \mathbb{N}$, we show that a successful adversary on $\mathsf{Group\text{-}pROS}_{\ell,\omega,\mathbb{T},\mathbb{T}}$ allows to break ℓ-OMUF with of Blaze+ with probability $\delta_{\mathsf{ver}}^{\omega}(1-\delta_{\mathsf{rej}})^{\omega}$. Note that this is the probability that the signer does not abort in ℓ signing sessions and that all obtained signatures verify (i.e., all obtained signatures satisfy $\|(\mathbf{z}_{\delta,i}^{\top}, y_{\delta,i})_{i\in[\omega]}\| \leq B_{\mathsf{ver}}$). Then, we can apply the attack from Section 3.2 to obtain a concrete attack.

Theorem 4.4. *For any PPT adversary $\mathcal{A}$ on $\mathsf{Group\text{-}pROS}_{\ell,\omega,\mathbb{T},\mathbb{T}}$, there is an adversary $\mathcal{B}$ against ℓ-OMUF*

of Blaze+ *such that* $\mathsf{Adv}^{\mathsf{OMUF}}_{\mathcal{B}}(\lambda) = \delta_{\mathsf{ver}}^{\ell}(1-\delta_{\mathsf{rej}})^{\ell} \cdot \mathsf{Adv}^{\mathsf{Group\text{-}pROS}}_{\mathcal{A},\ell,\omega,\mathbb{T},\mathbb{T}}(\lambda)$.

Proof. Let $\mathcal{A}$ be an adversary on $\mathsf{Group\text{-}pROS}_{\ell,\omega,\mathbb{T},\mathbb{T}}$ for some $\ell \in \mathbb{N}$. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ against ℓ-OMUF of Blaze+ as follows.

Adversary $\mathcal{B}$. Adversary $\mathcal{B}$ obtains a public key $(\mathbf{a}, t)$ and access to oracles H, S_1 and S_2. Then, $\mathcal{B}$ opens ℓ concurrent (signing) sessions via the S_1 oracle and obtains $(w_{i,k})_{k\in[\omega]}$ from the i^{th} session. Next, $\mathcal{B}$ generates a table T initialized with $\perp$ entries. Sets $\mathbf{v} = (w_{1,1}, w_{1,2}, \cdots, w_{\ell,\omega}) \in R_q^{\omega\ell}$.

Let Q be the number of $\mathsf{H}_{\mathsf{ros}}$ queries of $\mathcal{A}$ (including potential queries for verification of $\mathcal{A}$'s output). Assume without loss of generality that each $\mathsf{H}_{\mathsf{ros}}$ query is distinct. For the i^{th} such query $(\mathbf{A}, \mathsf{aux})$ to $\mathsf{H}_{\mathsf{ros}}$ of $\mathcal{A}$, adversary $\mathcal{B}$ proceeds as follows. If $\mathbf{A} \notin (\mathbb{T} \cup \{\perp\})^{\omega\times\omega\ell}$ or $\|\mathbf{A}[k]\|_\infty \neq 1$ for some $k \in [\omega]$, then returns $\vec{c}^{\,*} \leftarrow \mathbb{T}^\omega$. Else, sets $(w_1^*, \cdots, w_\omega^*)^\top = \mathbf{A} \cdot \mathbf{v}$ and $w^* \leftarrow \sum_{k\in[\omega]} w_k^*$. Also, sets $\mathsf{M} = (\mathsf{aux}, i)$ and $T[\mathbf{A}, \mathsf{aux}] = (w^*, \mathsf{M})$. Note that by construction, M is distinct for distinct queries. Sets $c^* = \mathsf{H}(w^*, \mathsf{M})$, decomposes $(\hat{c}_k)_{k\in[\omega]} = \mathsf{Decomp}(c^*)$ and outputs $\vec{c}^{\,*} = (\hat{c}_1, \cdots, \hat{c}_\omega)^\top$.

After $\mathcal{A}$'s query phase, it outputs $(\mathbf{A}_j, \mathsf{aux}_j)_{j\in[\ell+1]}$ and $(\vec{c}_i)_{i\in[\ell]}$. Adversary $\mathcal{B}$ checks if $\mathcal{A}$ is successful and outputs $\perp$ to the ℓ-OMUF game if not. If $\mathcal{A}$ was successful, then $\mathcal{B}$ parses $\vec{c}_i$ as $(\hat{c}_{i,k})_{k\in[\omega]} \in \mathbb{T}^\omega$. Then, $\mathcal{B}$ closes the i^{th} session with $(\hat{c}_{i,k})_{k\in[\omega]}$ via the S_2 oracle and obtains $(\mathbf{z}_{i,k}, y_{i,k})_{k\in[\omega]}$ if none of the sessions abort. For all $j \in [\ell+1]$ and $k \in [\omega]$, parses the k^{th} row of $\mathbf{A}_j$ as $(\perp, \cdots, \alpha_{j,k}, \cdots, \perp) = \mathbf{A}_j[k]$, where $\alpha_{j,k} \in \mathbb{T}$ is the non-$\perp$ entry at position $\nu_{j,k}$. Sets $\mu_{j,k} := (\nu'_{j,k}, \nu''_{j,k}) = (\lfloor \nu_{j,k}/\omega \rfloor, \nu_{j,k} \mod \omega) \in [\ell] \times [\omega]$. We write $\mathbf{z}_{\mu_{j,k}} = \mathbf{z}_{\nu'_{j,k},\nu''_{j,k}}$ for short, and extend this notation to $y_{\mu_{j,k}}, \hat{c}_{\mu_{j,k}}$ naturally. Using this notation, $\mathcal{B}$ constructs the values

$$\mathbf{z}_j^* = \sum_{k\in[\omega]} \alpha_{j,k} \cdot \mathbf{z}_{\mu_{j,k}} \qquad \text{and} \qquad y_j^* = \sum_{k\in[\omega]} \alpha_{j,k} \cdot y_{\mu_{j,k}}. \tag{11}$$

Also, $\mathcal{B}$ parses $(w_j^*, \mathsf{M}_j) \leftarrow T[\mathbf{A}_j, \mathsf{aux}_j]$. Note that by construction, we have that $w_j^* = \sum_{k\in[\omega]} \alpha_{j,k} \cdot w_{\mu_{j,k}}$. To generate the forgeries, $\mathcal{B}$ sets $c_j^* \leftarrow \mathsf{H}(w_j^*, \mathsf{M}_j)$ and $\sigma_j = (\mathbf{z}_j^*, y_j^*, \mathsf{M}_j)$. Finally, $\mathcal{B}$ outputs the forgeries $(\sigma_j, \mathsf{M}_j)_{j\in[\ell+1]}$ to the ℓ-OMUF game.

Success probability. Let us analyze the success probability. Recall that the random oracle H maps into $\mathcal{C}$, and for any $c \in \mathcal{C}$ we have that $\mathsf{Decomp}(c) \in \mathbb{T}^\omega$. Further, $\mathsf{Decomp}(c)$ is uniform over $\mathbb{T}^\omega$ for $c \leftarrow \mathcal{C}$ drawn at random. Thus, for each distinct $\mathsf{H}_{\mathsf{ros}}$ query, $\mathcal{B}$ outputs a random value in $\mathbb{T}^\omega$ as desired. Set $\mathbf{d} = (\vec{c}_1^{\,\top}, \cdots, \vec{c}_\ell^{\,\top})^\top$. If $\mathcal{A}$ is successful, we have that $(\mathbf{A}_j, \mathsf{aux}_j)_{j\in[\ell+1]}$ are pairwise distinct, $(\hat{c}_{i,k})_{k\in[\omega]} = \vec{c}_i \in \mathbb{T}^\omega$ and $\mathbf{A}_j \in (\mathbb{T} \cup \{\perp\})^{\omega\times\omega\ell}$ with $\|\mathbf{A}_j[k]\|_\infty = 1$ for all $k \in [\omega]$. By construction, we know that $(\mathsf{M}_j)_{j\in[\ell+1]}$ are pairwise distinct.

Also, for all $j \in [\ell+1]$ it holds that $\mathbf{A}_j \cdot \mathbf{d} = \mathsf{H}_{\mathsf{ros}}(\mathbf{A}_j, \mathsf{aux}_j)$. With the above notation, this is equivalent to $(\alpha_{j,k} \cdot \hat{c}_{\mu_{j,k}})_{k\in[\omega]} = \mathsf{H}_{\mathsf{ros}}(\mathbf{A}_j, \mathsf{aux}_j)$. Also, we have that $\mathsf{H}_{\mathsf{ros}}(\mathbf{A}_j, \mathsf{aux}_j) = \mathsf{Decomp}(\mathsf{H}(w_j^*, \mathsf{M}_j))$ by construction. Using both equalities and the definition of Decomp, we obtain for $j \in [\ell+1]$ that

$$\mathsf{H}(w_j^*, \mathsf{M}_j) = \sum_{k\in[\omega]} \alpha_{j,k} \cdot \hat{c}_{\mu_{j,k}} \tag{12}$$

Also, if none of the ℓ signing sessions abort, we have that for all $(i,k) \in [\ell] \times [\omega]$ that

$$w_{i,k} = \mathbf{a}^\top \mathbf{z}_{i,k} + y_{i,k} - t \cdot \hat{c}_{i,k} \tag{13}$$

Combining the above observations, we obtain that $c_j^* = \mathsf{H}(w_j^*, \mathsf{M}_j)$ and

$$\begin{aligned} w_j^* &= \sum_{k\in[\omega]} \alpha_{j,k} \cdot w_{\mu_{j,k}} \\ &\overset{(13)}{=} \sum_{k\in[\omega]} \alpha_{j,k}(\mathbf{a}^\top \mathbf{z}_{\mu_{j,k}} + y_{\mu_{j,k}} - t \cdot \hat{c}_{\mu_{j,k}}) \\ &= \mathbf{a}^\top (\sum_{k\in[\omega]} \alpha_{j,k} \mathbf{z}_{\mu_{j,k}}) + (\sum_{k\in[\omega]} \alpha_{j,k} y_{\mu_{j,k}}) - t \cdot (\sum_{k\in[\omega]} a_{\mu_{j,k}} \hat{c}_{\mu_{j,k}}) \end{aligned}$$

$$\overset{(11)}{=} \mathbf{a}^\top \mathbf{z}_j^* + y_j^* - t \cdot (\sum_{k\in[\omega]} a_{\mu_{j,k}} \hat{c}_{\mu_{j,k}})$$
$$\overset{(12)}{=} \mathbf{a}^\top \mathbf{z}_j^* + y_j^* - t \cdot \mathsf{H}(w_j^*, \mathsf{M}_j)$$
$$= \mathbf{a}^\top \mathbf{z}_j^* + y_j^* - t \cdot c_j^*$$

Also, since δ_{rej} is the probability that a signing session aborts and since $\mathcal{B}$ opens ℓ sessions in total, we have that no session aborts with probability at least $(1-\delta_{\mathsf{rej}})^\ell$. It remains to show that $\|((\mathbf{z}_j^*)^\top, y_j^*)\| \leq B_{\mathsf{ver}}$ with high probability. This follows as in the proof of correctness of Blaze+ [AEB20a, Theorem 1]. Roughly, observe that the multiplication with $\alpha_{j,k} \in \mathbb{T}$ does not increase the norm of $\mathbf{z}_{\mu_{j,k}}$ or $y_{\mu_{j,k}}$. Thus, the signatures output by $\mathcal{B}$ have the same norm distribution as honest (unblinded) signatures and consequently, the probability that all forgeries are valid is $\delta_{\mathsf{ver}}^\ell$. □

We can now combine Theorem 4.4 with our attack from Section 3.2 to break one-more unforgeability of Blaze+ in a concurrent setting.

Corollary 4.5. *Let $N \in \mathbb{N}$ and $\ell = \lceil \omega/N \rceil$. There is a PPT adversary on ℓ-OMUF of Blaze+ with success probability $\delta_{\mathsf{ver}}^\ell (1-\delta_{\mathsf{rej}})^\ell$ with expected $\mathcal{O}(\ell(2n)^N)$ queries to the hash functions.*

Concretely, in the instantiation of Blaze+, we have $n = 1024$ and $\omega = 16$ for $\lambda = 128$. Also, the abort probability δ_{rej} for rejection sampling is at most $\delta_{\mathsf{rej}} \leq 0.47$ in their instantiations, and the probability δ_{ver} that a honest signature verifies is at least $\delta_{\mathsf{ver}} \geq 1 - 2^{-\lambda}$. By taking $N = 1$, with 16 concurrent sessions of Blaze+ and an expected number of 2^{15} hash queries, we obtain an attack on 16-one-more unforgeability of Blaze+ with the probability greater than $2.8 \cdot 10^{-5}$. For $N = 4$, with 4 concurrent sessions of Blaze+ and an expected number of 2^{46} hash queries, we are able to break the 4-one-more unforgeability of it with the probability at least 0.073.

Remark 4.6 (Variant with 4 rounds). Note that a 4-move version of Blaze+ is also presented in the appendix of [AEB20b]. Here, the user has the option to abort in case the blinding procedure fails. In the event of an abort, the user is required to provide a proof that no valid signature was obtained. Our attack works without change on this variant.

4.2.2 BlindOR.

Here, we give an efficient attack on BlindOR [AHJ21]. At a high-level, BlindOR runs two instances of Blaze$^+$ in parallel and combines them with the OR-proof technique (similar to CSI-Otter). That is, one instance is run as usual, whereas the other instance is simulated. Each instance uses one share $\mathbf{c}_0$ or $\mathbf{c}_1$ as challenge, where $\mathbf{c} = \mathbf{c}_0 \odot \mathbf{c}_1$ is output by a hash function. Further, BlindOR makes other design choices listed below.

1. BlindOR uses challenges in $\mathbb{T}^\omega$ instead of $\mathcal{C}$ (as this allows to share the challenges multiplicatively over $\mathbb{T}$). On the other hand, Blaze$^+$ uses a single challenge $c \in \mathcal{C}$, but decomposes it into ω challenges in $\mathbb{T}$ during signing.

2. BlindOR performs η-many Blaze$^+$ protocols in parallel in S_1, but only a single protocol is completed in S_2. The completed protocol is the first for which rejection sampling succeeds (in random order).

3. Blaze$^+$ is described in the ring lattice setting, whereas BlindOR is described the module lattice setting.

For consistency, we present BlindOR in the ring setting (instead of in the module setting). We stress that this is purely to keep notation consistent and that it is straightforward to modify the attack to work in the module lattice setting. Also, since their instantiation uses $\eta = 1$, we omit the additional parallel executions in S_1 in our analysis. (We discuss in Remark 4.9 how to adapt the attack for $\eta > 1$.) Let $\mathsf{H} : \{0,1\}^* \to \mathbb{T}^\omega$ be a collision-resistant hash function. We describe the unblinded version of BlindOR in Figs. 6 and 7.

For our attack, we proceed as in our attack on Blaze+ (cf. Section 4.2) and handle the OR-proof as in our attack on CSI-Otter (cf. Section 4.1). We summarize our attack in the theorem below.

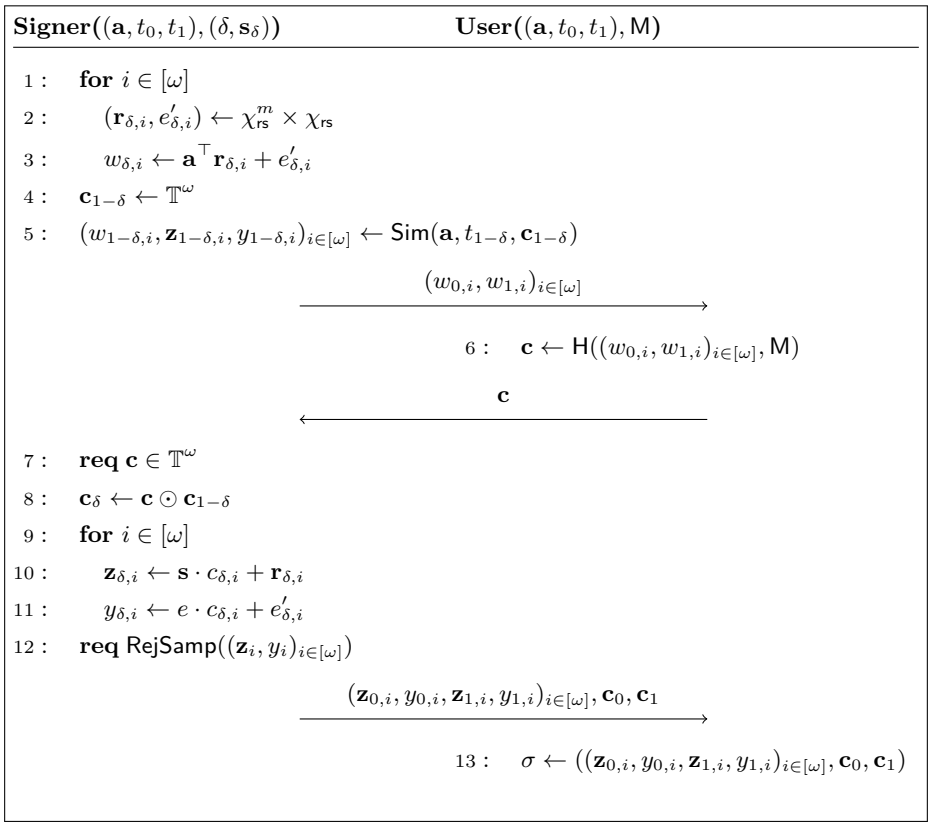

Figure 6: The unblinded signing protocol of BlindOR, where $\mathsf{H} : \{0,1\}^* \to \mathbb{T}^\omega$ is a collision-resistant hash function, $B_{\mathsf{ver}} \in R$ is some fixed norm bound, and $\odot$ is the entry-wise product. In the signing protocol, **req** aborts if the requirement does not hold. Key generation, verification and the simulator Sim are defined in Fig. 7

Theorem 4.7. *For any PPT adversary $\mathcal{A}$ on $\mathsf{Group\text{-}pROS}_{\ell,\omega,\mathbb{T},\mathbb{T}}$, there is an adversary $\mathcal{B}$ against ℓ-OMUF of BlindOR such that $\mathsf{Adv}^{\mathsf{OMUF}}_{\mathcal{B}}(\lambda) = \delta^{2\ell}_{\mathsf{ver}}(1-\delta_{\mathsf{rej}})^\ell \cdot \mathsf{Adv}^{\mathsf{Group\text{-}pROS}}_{\mathcal{A},\ell,\omega,\mathbb{T},\mathbb{T}}(\lambda)$.*

Proof. Let $\mathcal{A}$ be an adversary on $\mathsf{Group\text{-}pROS}_{\ell,\omega,\mathbb{T},\mathbb{T}}$ for some $\ell \in \mathbb{N}$. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ against ℓ-OMUF of BlindOR as follows.

Adversary $\mathcal{B}$. Adversary $\mathcal{B}$ obtains a public key $(\mathbf{a}, t_0, t_1)$ and access to oracles H, S_1 and S_2. Then, $\mathcal{B}$ opens ℓ concurrent (signing) sessions via the S_1 oracle and obtains $(w_{0,i,k}, w_{1,i,k})_{k\in[\omega]}$ from the i^{th} session. Next, $\mathcal{B}$ generates a table T initialized with $\perp$ entries. Sets $\mathbf{v}_\delta = (w_{\delta,1,1}, w_{\delta,1,2}, \cdots, w_{\delta,\ell,\omega}) \in R_q^{\omega\ell}$ for $\delta \in \{0,1\}$.

Let Q be the number of $\mathsf{H}_{\mathsf{ros}}$ queries of $\mathcal{A}$ (including potential queries for verification of $\mathcal{A}$'s output). Assume without loss of generality that each $\mathsf{H}_{\mathsf{ros}}$ query is distinct. For the i^{th} such query $(\mathbf{A}, \mathsf{aux})$ to $\mathsf{H}_{\mathsf{ros}}$ of $\mathcal{A}$, adversary $\mathcal{B}$ proceeds as follows. If $\mathbf{A} \notin (\mathbb{T} \cup \{\perp\})^{\omega\times\omega\ell}$ or $\|\mathbf{A}[k]\|_\infty \neq 1$ for some $k \in [\omega]$, then returns $\mathbf{c}^* \leftarrow \mathbb{T}^\omega$. Else, parses the k^{th} row of $\mathbf{A}$ as $(\perp, \cdots, \alpha_k, \cdots, \perp) = \mathbf{A}[k]$, where $\alpha_k \in \mathbb{T}$ is the non-$\perp$ entry in column ν_k. Sets $w^*_{0,k} = \alpha_k \mathbf{v}_{0,\nu_k}$ and $w^*_{1,k} = \mathbf{v}_{1,\nu_k}$ for $k \in [\omega]$. Note that $(w^*_{0,1}, \cdots, w^*_{0,\omega}) = \mathbf{A} \cdot \mathbf{v}_0$. Also, sets $\mathsf{M} = (\mathsf{aux}, i)$ and $T[\mathbf{A}, \mathsf{aux}] = ((w^*_{0,k}, w^*_{1,k})_{k\in[\omega]}, \mathsf{M})$. Note that by construction, M is distinct for distinct queries. Sets $\mathbf{c}^* = \mathsf{H}((w^*_{0,k}, w^*_{1,k})_{k\in[\omega]}, \mathsf{M})$ and outputs $\mathbf{c}^*$.

After $\mathcal{A}$'s query phase, it outputs $(\mathbf{A}_j, \mathsf{aux}_j)_{j\in[\ell+1]}$ and $(\mathbf{c}_i)_{i\in[\ell]}$. Adversary $\mathcal{B}$ checks if $\mathcal{A}$ is successful and outputs $\perp$ to the ℓ-OMUF game if not. If $\mathcal{A}$ was successful, $\mathcal{B}$ closes the i^{th} session with $\mathbf{c}_i$ via the S_2 oracle and obtains $((\mathbf{z}_{i,0,k}, y_{i,0,k}, \mathbf{z}_{i,1,k}, y_{i,1,k})_{k\in[\omega]}, \mathbf{c}_{i,0}, \mathbf{c}_{i,1})$ if none of the sessions abort. For all $j \in [\ell+1]$ and $k \in [\omega]$, parses the k^{th} row of $\mathbf{A}_j$ as above, i.e., $(\perp, \cdots, \alpha_{j,k}, \cdots, \perp) = \mathbf{A}_j[k]$, where $\alpha_{j,k} \in \mathbb{T}$ is the non-$\perp$

$\mathsf{Verify}((\mathbf{a}, \mathbf{b}_0, \mathbf{b}_1), \sigma, \mathsf{M})$:

1 : **parse** $\sigma = ((\mathbf{z}_{0,i}, y_{0,i}, \mathbf{z}_{1,i}, y_{1,i})_{i\in[\omega]}, \mathbf{c}_0, \mathbf{c}_1)$
2 : **for** $\delta \in \{0,1\}$
3 : **req** $\|(\mathbf{z}_{\delta,i}^\top, y_{\delta,i})_{i\in[\omega]}\| \leq B_{\mathsf{ver}}$
4 : **for** $i \in [\omega]$
5 : $w_{\delta,i} \leftarrow \mathbf{a}^\top \mathbf{z}_{\delta,i} + y_{\delta,i} - t_\delta \mathbf{c}_{\delta,i}$
6 : **req** $\mathbf{c}_0 \odot \mathbf{c}_1 = \mathsf{H}((w_{\delta,i}, w_{\delta,i})_{i\in[\omega]}, \mathsf{M})$
7 : **return** 1

$\mathsf{Sim}(\mathbf{a}, t, \mathbf{c})$:

1 : **for** $i \in [\omega]$
2 : $(\mathbf{z}_i, y_i) \leftarrow \chi_{\mathsf{rs}} \times \chi_{\mathsf{rs}}$
3 : $w_i = \mathbf{a}^\top \mathbf{z}_i + y_i - t c_i$
4 : **return** $(w_i, \mathbf{z}_i, y_i)_{i\in[\omega]}$

$\mathsf{KeyGen}(1^\lambda)$:

1 : $\mathbf{a} \leftarrow R_q^m$
2 : **for** $\delta \in \{0,1\}$
3 : $(\mathbf{s}_\delta, e_\delta) \leftarrow \chi^m \times \chi$
4 : **if** $(\|(\mathbf{s}_\delta^\top, e_\delta)\| > B_{\mathsf{kg}})$ **then**
5 : **goto** line 3
6 : $t_\delta \leftarrow \mathbf{a}^\top \mathbf{s}_\delta + e_\delta$
7 : $\delta \leftarrow \{0,1\}$
8 : $\mathsf{sk} = (\delta, \mathbf{s}_\delta, e_\delta)$
9 : $\mathsf{pk} = (\mathbf{a}, t_0, t_1)$
10 : **return** sk, pk

Figure 7: The key generation and verification protocol for BlindOR, where $B_{\mathsf{kg}} \in R$ is some fixed norm bound, and $\odot$ is the entry-wise product. We also provide a description of the simulator Sim. In the verification, **req** returns 0 if the requirement does not hold. Note that instead of running the simulator again if it aborts as in [AHJ21], we define Sim such that it does not abort directly. The unblinded signing protocol is defined in Fig. 6.

entry at position $\nu_{j,k}$. For $b \in \{0,1\}$, sets $\mu_{j,b,k} := (\nu'_{j,k}, b, \nu''_{j,k}) = (\lfloor \nu_{j,k}/\omega \rfloor, b, \nu_{j,k} \mod \omega) \in [\ell]\times\{0,1\}\times[\omega]$. For values $\mathbf{x}$ indexed over $[\ell] \times \{0,1\} \times [\omega]$, we write $\mathbf{x}_{\mu_{j,b,k}}$ short for $\mathbf{x}_{\nu'_{j,k},b,\nu''_{j,k}}$. Using this notation, for $j \in [\ell+1]$ and $k \in [\omega]$, adversary $\mathcal{B}$ sets

$$\begin{matrix} \mathbf{z}^*_{j,0,k} = \alpha_{j,k} \cdot \mathbf{z}_{\mu_{j,0,k}} \\ \mathbf{z}^*_{j,1,k} = \mathbf{z}_{\mu_{j,0,k}} \end{matrix}, \quad \begin{matrix} y^*_{j,0,k} = \alpha_{j,k} \cdot y_{\mu_{j,0,k}} \\ y^*_{j,1,k} = y_{\mu_{j,1,k}} \end{matrix} \quad \text{and} \quad \begin{matrix} \mathbf{c}^*_{j,0,k} = \alpha_{j,k} \cdot \mathbf{c}_{\mu_{j,0,k}} \\ \mathbf{c}^*_{j,1,k} = \mathbf{c}_{\mu_{j,1,k}} \end{matrix} \tag{14}$$

Sets $\mathbf{c}^*_{j,b} = (\mathbf{c}^*_{j,b,1}, \cdots, \mathbf{c}^*_{j,b,\omega})$. Also, $\mathcal{B}$ parses $((w^*_{j,0,k}, w^*_{j,1,k})_{k\in[\omega]}, \mathsf{M}_j) \leftarrow T[\mathbf{A}_j, \mathsf{aux}_j]$. Note that by construction, we have that

$$w^*_{j,b,k} = \alpha_{j,k}^{1-b} \cdot w_{\mu_{j,b,k}}. \tag{15}$$

Finally, $\mathcal{B}$ sets $\sigma_j = ((\mathbf{z}^*_{j,0,k}, \mathbf{z}^*_{j,1,k}, y^*_{j,0,k}, y^*_{j,0,k})_{k\in[\omega]}, \mathbf{c}^*_{j,0}, \mathbf{c}^*_{j,1})$ and outputs the forgeries $(\sigma_j, \mathsf{M}_j)_{j\in[\ell+1]}$ to the ℓ-OMUF game.

Success probability. Let us analyze the success probability. It is easy to check that for each distinct $\mathsf{H}_{\mathsf{ros}}$ query, $\mathcal{B}$ outputs a random value in $\mathbb{T}^\omega$ as desired. Set $\mathbf{d} = (\mathbf{c}_1^\top, \cdots, \mathbf{c}_\ell^\top)^\top$. If $\mathcal{A}$ is successful, we have that $(\mathbf{A}_j, \mathsf{aux}_j)_{j\in[\ell+1]}$ are pairwise distinct, $\mathbf{c}_i \in \mathbb{T}^\omega$ and $\mathbf{A}_j \in (\mathbb{T} \cup \{\perp\})^{\omega\times\omega\ell}$ with $\|\mathbf{A}_j[k]\|_\infty = 1$ for all $k \in [\omega]$. Also, we know that $(\mathsf{M}_j)_{j\in[\ell+1]}$ are pairwise distinct by construction. Further, for all $j \in [\ell+1]$ it holds that $\mathbf{A}_j \cdot \mathbf{d} = \mathsf{H}_{\mathsf{ros}}(\mathbf{A}_j, \mathsf{aux}_j) := \mathbf{c}^*_j$. Using the above notation, this is equivalent to $\alpha_{j,k}\mathbf{d}_{\nu_{j,k}} = \mathbf{c}^*_{j,k}$ for all $k \in [\omega]$. By construction, we have that $\mathsf{H}_{\mathsf{ros}}(\mathbf{A}_j, \mathsf{aux}_j) = \mathsf{H}((w^*_{j,0,k}, w^*_{j,1,k})_{k\in[\omega]}, \mathsf{M}_j)$. Using both equalities, we obtain for $j \in [\ell+1]$ that

$$\mathsf{H}((w^*_{j,0,k}, w^*_{j,1,k})_{k\in[\omega]}, \mathsf{M}_j) = (\alpha_{j,k} \cdot \mathbf{d}_{\nu_{j,k}})_{k\in[\omega]} \tag{16}$$

Also, if none of the ℓ signing sessions abort, we have that for all $(i,b,k) \in [\ell] \times \{0,1\} \times [\omega]$ that

$$\begin{matrix} w_{i,b,k} = \mathbf{a}^\top \mathbf{z}_{i,b,k} + y_{i,b,k} - t_b \cdot \mathbf{c}_{i,b,k} \\ \mathbf{c}_i = \mathbf{c}_{i,0} \odot \mathbf{c}_{i,1} \end{matrix} \tag{17}$$

Combining the above observations, we obtain for all $j \in [\ell], b \in \{0,1\}, k \in [\omega]$ that

$$w^*_{j,b,k} \overset{(15)}{=} \alpha_{j,k}^{1-b} \cdot w_{\mu_{j,b,k}}$$

$$\overset{(17)}{=} \alpha_{j,k}^{1-b}(\mathbf{a}^\top \mathbf{z}_{\mu_{j,b,k}} + y_{\mu_{j,b,k}} - t_b \cdot \mathbf{c}_{\mu_{j,b,k}})$$
$$= \mathbf{a}^\top(\alpha_{j,k}^{1-b} \cdot \mathbf{z}_{\mu_{j,b,k}}) + (\alpha_{j,k}^{1-b} \cdot y_{\mu_{j,b,k}}) - t_b \cdot (\alpha_{j,k}^{1-b} \cdot \mathbf{c}_{\mu_{j,b,k}})$$
$$\overset{(14)}{=} \mathbf{a}^\top \mathbf{z}^*_{j,b,k} + y^*_{j,b,k} - t_b \cdot \mathbf{c}^*_{j,b,k}$$

and

$$\mathsf{H}((w^*_{j,0,k}, w^*_{j,1,k})_{k\in[\omega]}, \mathsf{M}_j) \overset{(16)}{=} \alpha_{j,k} \cdot \mathbf{d}_{\nu_{j,k}}$$
$$\overset{(17)}{=} \alpha_{j,k} \cdot \mathbf{c}_{\mu j,0,k} \cdot \mathbf{c}_{\mu j,1,k}$$
$$\overset{(14)}{=} \mathbf{c}^*_{j,0,k} \cdot \mathbf{c}^*_{j,1,k}$$

Also, since δ_{rej} is the probability that rejection sampling fails and since $\mathcal{B}$ opens ℓ sessions in total, we have that no session aborts with probability at least $(1-\delta_{\mathsf{rej}})^\ell$. We can show that $\|((\mathbf{z}^*_{j,b,k})^\top, y^*_{j,b,k})\| \leq B_{\mathsf{ver}}$ as in the proof of Corollary 4.5 (i.e., our attack on Blaze^+). Since we obtain two Blaze^+ signatures per signature, the norm bound holds for all signatures with probability $\delta_{\mathsf{ver}}^{2\ell}$.

□

Corollary 4.8. *Let $N \in \mathbb{N}$ and $\ell = \lceil \omega/N \rceil$. There exists a PPT adversary on ℓ-OMUF of* BlindOR *with success probability $\delta_{\mathsf{ver}}^{2\ell}(1-\delta_{\mathsf{rej}})^\ell$ and with expected $\mathcal{O}(\ell(2n)^N)$ queries to the hash functions.*

For concrete numbers, we refer to the analysis of Blaze^+ (cf. Section 4.2)[6]. Note that the success probability is to be multiplied with $(1-2^\lambda)^\ell \approx 1$ due to the additional factor 2 in the exponent of δ_{ver}.

Remark 4.9. To reduce the abort probability δ_{rej} due to the signer's rejection sampling in S_2, BlindOR gives the option to initiate $\eta > 1$ signing sessions of S_1 concurrently, but only a single session is finished (if any). This session is the first session (chosen at random) where rejection sampling does not abort in S_2. Our attack can be adapted to this setting by guessing which session is finished. Then, we can proceed with our attack as before. The probability that we guess correctly in our attack is at least $(1/\eta)^\ell$.

4.3 Blind Signature based on Parallel Schnorr

Let $(\mathbb{G}, +)$ be a group with prime order p. We use additive notation. Let $g \in \mathbb{G}$ be a generator. We define a natural variant of blind Schnorr with ω parallel repetitions. The interactive signing protocol, key generation and verification is defined in Figure 8. We also provide a full specification in Figure 9 (without security proofs). We analyze it exclusively for illustrative purposes and note that the benefit compared to the original blind Schnorr is marginal.

Conceptually, the use of both parallel repetitions *and* exponential challenge space should make the scheme more secure. For example for the standard Schnorr Σ-protocol, the soundness error $\varepsilon := 1/p$ can be exponentially reduced with ω parallel repetitions to ε^ω [Dam02]. Since blind Schnorr remains secure for $o(\log p)$-many concurrent sessions, perhaps its security can be amplified in a similar manner due to the exponential challenge space. We show that this intuition is wrong and there is an efficient attack on the scheme.

In more detail, we show that a successful adversary on $\mathsf{Ring\text{-}pROS}_{\ell,\omega,\mathbb{Z}_p,\mathbb{Z}_p}$ allows to break the ℓ-OMUF of parallel blind Schnorr with ω repetitions. Then, we can apply the attack from Section 3.3 since $\mathbb{Z}_p$ is a field and binary decomposition is efficient over $\mathbb{Z}_p$.

Theorem 4.10. *For any PPT adversary $\mathcal{A}$ on $\mathsf{Ring\text{-}pROS}_{\ell,\omega,\mathbb{Z}_p,\mathbb{Z}_p}$, there is an adversary $\mathcal{B}$ against ℓ-OMUF of parallel blind Schnorr such that $\mathsf{Adv}^{\mathsf{OMUF}}_{\mathcal{B}}(\lambda) = \mathsf{Adv}^{\mathsf{Ring\text{-}pROS}}_{\mathcal{A},\ell,\omega,\mathcal{R},\mathcal{R}_c}(\lambda)$.*

[6] Since BlindOR runs two Blaze^+ instances in parallel, the efficiency of the attack is identical for shared concrete parameters. Note that we presented our attack in the ring lattice setting, but as mentioned above, it is straightforward to adapt it to the module lattice setting of the BlindOR instantiation.

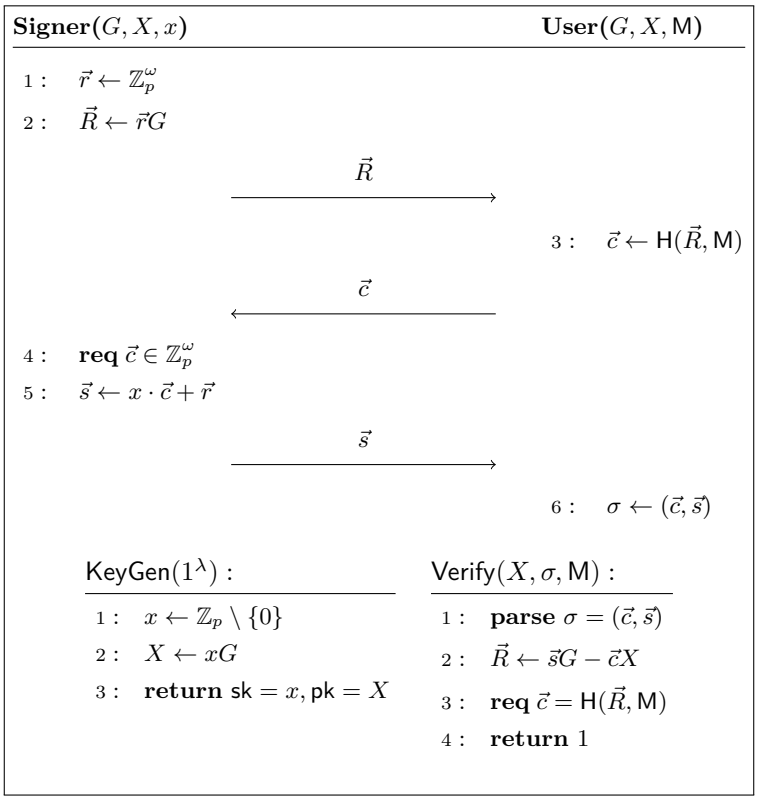

Figure 8: An interactive signing protocol for Schnorr with ω parallel repetitions. We also specify key generation and verification. In the signing protocol (resp. verification), **req** aborts (resp. returns 0) if the requirement does not hold.

Proof. Let $\mathcal{A}$ be an adversary on $\mathsf{Ring\text{-}pROS}_{\ell,\omega,\mathbb{Z}_p,\mathbb{Z}_p}$ for some $\ell \in \mathbb{N}$. We construct an adversary $\mathcal{B}$ against ℓ-OMUF of parallel blind Schnorr with ω parallel repetitions using $\mathcal{A}$.

Adversary $\mathcal{B}$. Initially, $\mathcal{B}$ obtains a public key X and access to oracles H, S_1 and S_2. Then, $\mathcal{B}$ opens ℓ concurrent (signing) sessions via the S_1 oracle. It obtains $\vec{R}_i$ from the i^{th} session. Then, $\mathcal{B}$ generates a table T initialized with $\perp$ entries. Sets $\vec{S} \leftarrow (\vec{R}_1^\top, \cdots, \vec{R}_\ell^\top)^\top \in \mathbb{G}^{\omega \cdot \ell}$.

Let Q be the number of $\mathsf{H}_{\mathsf{ros}}$ queries of $\mathcal{A}$ (including potential queries for verification of $\mathcal{A}$'s output). Assume without loss of generality that each $\mathsf{H}_{\mathsf{ros}}$ query is distinct. For the i^{th} such query $(\mathbf{A}, \mathsf{aux})$ to $\mathsf{H}_{\mathsf{ros}}$ of $\mathcal{A}$, adversary $\mathcal{B}$ proceeds as follows. If $\mathbf{A} \notin \mathbb{Z}_p^{\omega \times \omega\ell}$, then returns $\vec{c}^* \leftarrow \mathbb{Z}_p^\omega$. Otherwise, sets $\vec{R}^* \leftarrow \mathbf{A} \cdot \vec{S}$ and $\mathsf{M} = (\mathsf{aux}, i)$. Sets $T[\mathbf{A}, \mathsf{aux}] = (\vec{R}^*, \mathsf{M})$. Note that by construction, M is distinct for distinct queries. Outputs $\mathsf{H}(\vec{R}^*, \mathsf{M})$.

After $\mathcal{A}$'s query phase, it outputs $(\mathbf{A}_j, \mathsf{aux}_j)_{j \in [\ell+1]}$ and $(\vec{c}_i)_{i \in [\ell]}$. Adversary $\mathcal{B}$ checks if $\mathcal{A}$ is successful and outputs $\perp$ to the ℓ-OMUF game if not. Next, closes the i^{th} session with $\vec{c}_i$ via the S_2 oracle and obtains $\vec{s}_i$. Sets $\vec{t} \leftarrow (\vec{s}_1^\top, \cdots, \vec{s}_\ell^\top)^\top$. Now, $\mathcal{B}$ parses $(\vec{R}_j^*, \mathsf{M}_j) \leftarrow T[\mathbf{A}_j, \mathsf{aux}_j]$. Sets $\vec{s}_j^* \leftarrow \mathbf{A}_j \cdot \vec{t}$, $\vec{c}_j^* \leftarrow \mathsf{H}(\vec{R}_j^*, \mathsf{M}_j)$, and $\sigma_j = (\vec{c}_j^*, \vec{s}_j^*)$. Finally, outputs the forgeries $(\sigma_j, \mathsf{M}_j)_{j \in [\ell+1]}$ to the ℓ-OMUF game.

Success probability. Let us analyze the success probability. Recall that the random oracle H maps into $\mathbb{Z}_p^\omega$. Thus, for each distinct $\mathsf{H}_{\mathsf{ros}}$ query, $\mathcal{B}$ outputs a random value in $\mathbb{Z}_p^\omega$ as desired. Set $\vec{d} := (\vec{c}_1^\top, \cdots, \vec{c}_\ell^\top)^\top \in \mathbb{Z}_p^{\omega\ell}$. If $\mathcal{A}$ is successful, we have that $(\mathbf{A}_j, \mathsf{aux}_j)_{j \in [\ell+1]}$ are pairwise distinct, $\vec{c}_i \in \mathbb{Z}_p^\omega$ and $\mathbf{A}_j \in \mathbb{Z}_p^{\omega \times \omega\ell}$. Also, for all $j \in [\ell+1]$ it holds that $\mathbf{A}_j \cdot \vec{d} = \mathsf{H}_{\mathsf{ros}}(\mathbf{A}_j, \mathsf{aux}_j)$. Thus, we know that $(\mathsf{M}_j)_{j \in [\ell+1]}$ are pairwise distinct by

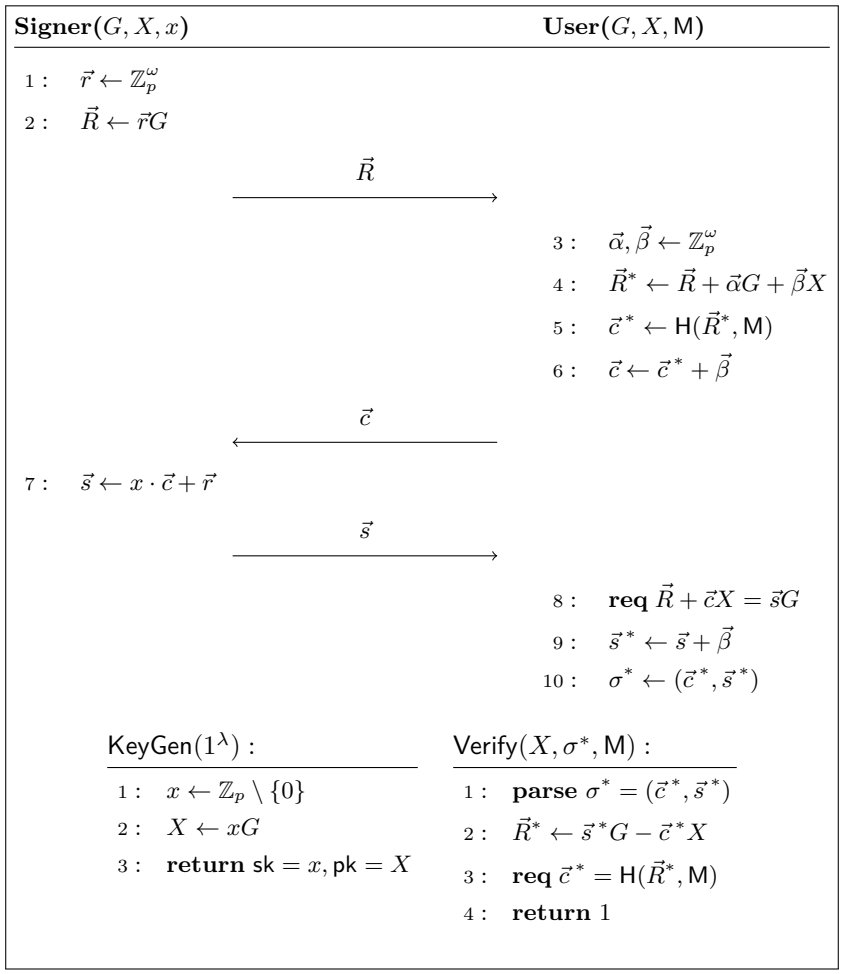

Figure 9: The blinded protocol for parallel Schnorr with ω parallel repetitions.

construction. Also, all forgeries are valid since $\vec{c}_j^* = \mathsf{H}(\vec{R}_j^*, \mathsf{M}_j)$ and

$$\begin{aligned}
\vec{R}_j^* &= \mathbf{A}_j \cdot \vec{S} = \mathbf{A}_j \cdot (\vec{R}_1^\top, \cdots, \vec{R}_\ell^\top)^\top \\
&= \mathbf{A}_j \cdot ((\vec{s}_1 G - \vec{c}_1 X)^\top, \cdots, (\vec{s}_\ell G - \vec{c}_\ell X)^\top)^\top \\
&= \mathbf{A}_j \cdot (\vec{t}G - \vec{d}X) = (\mathbf{A}_j \cdot \vec{t})G - (\mathbf{A}_j \cdot \vec{d})X \\
&= \vec{s}_j^* G - \mathsf{H}_{\mathsf{ros}}(\mathbf{A}_j, \mathsf{aux}_j)X \\
&= \vec{s}_j^* G - \mathsf{H}(\vec{R}_j^*, \mathsf{M}_j)X = \vec{s}_j^* G - \vec{c}_j^* X.
\end{aligned}$$

□

Set $B_\mu = 2^{\mu-1}$. If we combine Theorem 4.10 with Theorem 3.5 (i.e., the attack from Section 3.3 with binary generating set $(B_\mu)_{\mu \in [\lceil \log\ p \rceil]}$), we obtain an attack on parallel blind Schnorr. Since any value in $\mathbb{Z}_p$ can be decomposed in binary efficiently, the attack runs in $\mathcal{O}(\ell)$ for $\ell \geq \omega \lceil \log\ p \rceil$. Note that in practice, we often have $\lceil \log\ p \rceil = 256$ and the attack remains concretely efficient for a large number of parallel repetitions ω.

Corollary 4.11. *Let $n \in \mathbb{N}$ and $\ell \geq \omega \lceil \log\ p \rceil$. There exists an PPT adversary on ℓ-OMUF of parallel Schnorr with success probability 1 that runs in time $\mathcal{O}(\ell)$, counting hash queries, binary decompositions and operations in $\mathbb{Z}_p$ and $\mathbb{G}$.*

Acknowledgements

Shuichi Katsumata was supported by JST CREST Grant Number JPMJCR22M1, JST AIP Acceleration Research JPMJCR22U5, and JSPS KAKENHI Grant Number JP19H01109, Japan. Yi-Fu Lai was supported in part by the Ministry for Business, Innovation and Employment of New Zealand and was also supported by the European Union (ERC AdG REWORC - 101054911).

References

[Abe01] Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 136–151. Springer, Heidelberg, May 2001.

[AEB20a] Nabil Alkeilani Alkadri, Rachid El Bansarkhani, and Johannes Buchmann. BLAZE: Practical lattice-based blind signatures for privacy-preserving applications. In Joseph Bonneau and Nadia Heninger, editors, *FC 2020*, volume 12059 of *LNCS*, pages 484–502. Springer, Heidelberg, February 2020.

[AEB20b] Nabil Alkeilani Alkadri, Rachid El Bansarkhani, and Johannes Buchmann. On lattice-based interactive protocols: An approach with less or no aborts. In Joseph K. Liu and Hui Cui, editors, *ACISP 20*, volume 12248 of *LNCS*, pages 41–61. Springer, Heidelberg, November / December 2020.

[AGHO11] Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 649–666. Springer, Heidelberg, August 2011.

[AHJ21] Nabil Alkeilani Alkadri, Patrick Harasser, and Christian Janson. BlindOR: an efficient lattice-based blind signature scheme from OR-proofs. In Mauro Conti, Marc Stevens, and Stephan Krenn, editors, *CANS 21*, volume 13099 of *LNCS*, pages 95–115. Springer, Heidelberg, December 2021.

[AO00] Masayuki Abe and Tatsuaki Okamoto. Provably secure partially blind signatures. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 271–286. Springer, Heidelberg, August 2000.

[BCC04] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 2004*, pages 132–145. ACM Press, October 2004.

[BDE+22] Maxime Buser, Rafael Dowsley, Muhammed F. Esgin, Clémentine Gritti, Shabnam Kasra Kermanshahi, Veronika Kuchta, Jason T. LeGrow, Joseph K. Liu, Raphaël C.-W. Phan, Amin Sakzad, Ron Steinfeld, and Jiangshan Yu. A survey on exotic signatures for post-quantum blockchain: Challenges & research directions. *ACM Comput. Surv.*, 2022. Just accepted.

[BFPV13] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Short blind signatures. *Journal of computer security*, 21(5):627–661, 2013.

[BLL+21] Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. In Anne Canteaut and Franccois-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 33–53. Springer, Heidelberg, October 2021.

[BLNS23] Ward Beullens, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Lattice-based blind signatures: Short, efficient, and round-optimal. Cryptology ePrint Archive, Paper 2023/077, 2023. https://eprint.iacr.org/2023/077.

[Cha82] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982.

[CKLR18] Mauro Conti, E Sandeep Kumar, Chhagan Lal, and Sushmita Ruj. A survey on security and privacy issues of bitcoin. *IEEE communications surveys & tutorials*, 20(4):3416–3452, 2018.

[CP93] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, Heidelberg, August 1993.

[Dam02] Ivan Damgård. On σ-protocols. *Lecture Notes, University of Aarhus, Department for Computer Science*, page 84, 2002.

[DHK+23] Julien Duman, Dominik Hartmann, Eike Kiltz, Sabrina Kunzweiler, Jonas Lehmann, and Doreen Riepel. Generic models for group actions. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 406–435. Springer, Heidelberg, May 2023.

[dK22] Rafaël del Pino and Shuichi Katsumata. A new framework for more efficient round-optimal lattice-based (partially) blind signature via trapdoor sampling. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 306–336. Springer, Heidelberg, August 2022.

[Fis06] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 60–77. Springer, Heidelberg, August 2006.

[FW22] Georg Fuchsbauer and Mathias Wolf. (concurrently secure) blind schnorr from schnorr. *IACR Cryptol. ePrint Arch.*, page 1676, 2022.

[HIP+22] Scott Hendrickson, Jana Iyengar, Tommy Pauly, Steven Valdez, and Christopher A. Wood. Private access tokens. internet-draft draft-private-access-tokens-01, April 2022. Work in Progress.

[HKL19] Eduard Hauck, Eike Kiltz, and Julian Loss. A modular treatment of blind signatures from identification schemes. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 345–375. Springer, Heidelberg, May 2019.

[HKLN20] Eduard Hauck, Eike Kiltz, Julian Loss, and Ngoc Khanh Nguyen. Lattice-based blind signatures, revisited. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 500–529. Springer, Heidelberg, August 2020.

[KLLQ23] Shuichi Katsumata, Yi-Fu Lai, Jason T. LeGrow, and Ling Qin. CSI -otter: Isogeny-based (partially) blind signatures from the class group action with a twist. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 729–761. Springer, 2023.

[KLX22a] Julia Kastner, Julian Loss, and Jiayu Xu. The abe-okamoto partially blind signature scheme revisited. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 279–309. Springer, Heidelberg, December 2022.

[KLX22b] Julia Kastner, Julian Loss, and Jiayu Xu. On pairing-free blind signature schemes in the algebraic group model. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 468–497. Springer, Heidelberg, March 2022.

[KRS23] Shuichi Katsumata, Michael Reichle, and Yusuke Sakai. Practical round-optimal blind signatures in the rom from standard assumptions. *Cryptology ePrint Archive*, 2023. To Appear in ASIACRYPT.

[KSD19] Mojtaba Khalili, Daniel Slamanig, and Mohammad Dakhilalian. Structure-preserving signatures on equivalence classes from standard assumptions. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 63–93. Springer, Heidelberg, December 2019.

[MSF10] Sarah Meiklejohn, Hovav Shacham, and David Mandell Freeman. Limitations on transformations from composite-order to prime-order groups: The case of round-optimal blind signatures. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 519–538. Springer, Heidelberg, December 2010.

[SC12] Jae Hong Seo and Jung Hee Cheon. Beyond the limitation of prime-order bilinear groups, and round optimal blind signatures. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 133–150. Springer, Heidelberg, March 2012.

[Sch01] Claus-Peter Schnorr. Security of blind discrete log signatures against interactive attacks. In Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *ICICS 01*, volume 2229 of *LNCS*, pages 1–12. Springer, Heidelberg, November 2001.

[TZ22] Stefano Tessaro and Chenzhi Zhu. Short pairing-free blind signatures with exponential security. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 782–811. Springer, Heidelberg, May / June 2022.

[VPN22] Vpn by Google one, explained. https://one.google.com/about/vpn/howitworks, 2022. Accessed: 2023-02-02.

[Wag02] David Wagner. A generalized birthday problem. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, Heidelberg, August 2002.

[YL19] Xun Yi and Kwok-Yan Lam. A new blind ECDSA scheme for bitcoin transaction anonymity. In Steven D. Galbraith, Giovanni Russello, Willy Susilo, Dieter Gollmann, Engin Kirda, and Zhenkai Liang, editors, *ASIACCS 19*, pages 613–620. ACM Press, July 2019.