On Pairing-Free Blind Signature Schemes in the Algebraic Group Model

Julia Kastner
ETH Zürich
julia.kastner@inf.ethz.ch

Julian Loss
University of Maryland
lossjulian@gmail.com

Michael Rosenberg
University of Maryland
micro@cs.umd.edu

Jiayu Xu
George Mason University
jxu27@gmu.edu

***Abstract*—Studying the security and efficiency of blind signatures is an important goal for privacy sensitive applications. In particular, for large-scale settings (e.g. cryptocurrency tumblers), it is important for schemes to scale well with the number of users in the system. Unfortunately, all practical, group-based schemes either 1) rely on (very strong) number theoretic hardness assumptions and computationally expensive pairing operations over bilinear groups or 2) support only a polylogarithmic number of *concurrent* (i.e., arbitrarily interleaved) signing sessions per public key. Following the recent work of Fuchsbauer et al. (EUROCRYPT '20), we revisit the security of two *pairing-free* blind signature schemes in the algebraic group model (AGM) + Random Oracle Model (ROM). First, we prove that the popular blind Schnorr scheme is secure under the one-more discrete logarithm assumption if (polynomially many) signatures are issued *sequentially*. This stands in stark contrast to the results of Fuchsbauer et al. and Benhamouda et al. (EPRINT '20). Under the same assumptions, their (combined) results imply security against a polynomial time attacker iff the signer opens at most polylogarithmically many *concurrent* signing sessions. We then reconsider the security of Abe's scheme (EUROCRYPT '01), which is known to have a flawed proof in the plain ROM. We give a proof under the discrete logarithm assumption in the AGM+ROM, even for (polynomially many) *concurrent* signing sessions. Finally, we demonstrate that these pairing-free signature schemes are immediately usable in a real-world setting. Using a cryptocurrency tumbling service as a model, we benchmark the Schnorr and Abe schemes under different workloads and degrees of parallelism and conclude that they can both handle large workloads at reasonable security levels, and have distinct optimal use cases.**

I. Introduction

Blind signatures, first introduced by Chaum [Cha82], are a fundamental cryptographic building block. They find applications in many privacy sensitive applications such as anonymous credentials, eCash, and eVoting. Informally, a blind signature scheme is a interactive protocol between a *user* and a *signer*. Here, the signer holds a secret key sk and the user holds the corresponding public key pk. The goal of the interaction is for the user to learn a signature σ on a message m of its choice such that σ can efficiently be verified under pk. The protocol should ensure two properties: (1) *Blindness:* the signer can not link the transcripts of protocol runs to the signatures that they created. In particular, it does not learn the messages that it signs. (2) *One-More-Unforgeability:* if the protocol is run ℓ times, the user should not be able to create $\ell+1$ or more valid signatures. In spite of decades of study, the security guarantees of practical blind signature schemes are not satisfactory. Most constructions rely on strong number-theoretic hardness assumptions and/or computationally expensive pairing operations over bilinear groups groups [Bol03; Bel+03; Oka06; GG14; FHS15]. Other constructions rely on weaker assumptions (and no pairings) but allow only for a very small (polylogarithmic) number of signatures to be issued per public key concurrently, i.e., when sessions may overlap in an arbitrary fashion [PS96; PS97; PS00; HKL19; Hau+20]. The reason for this is that the homomorphic structure of these schemes gives rise to the so-called ROS attack (Random inhomogenities in Overdetermined Sytems of equations) when sufficiently many sessions of the scheme are executed concurrently. Shortly after its discovery by Schnorr [Sch01], Wagner [Wag02] showed how to carry out the ROS attack in sub-exponential time against the Schnorr and Okamoto-Schnorr signature schemes.[1] A recent work of Benhamouda et al. [Ben+20] improved the attack to polynomial time, given a polylogarithmic number of concurrent signing sessions. In this work, we revisit the security properties of existing blind signature schemes that *without pairings.* Concretely, we make the following contributions.

A. Sequential Security of Blind Schnorr.

We first study the security of blind Schnorr signatures in the algebraic group model (AGM) [FKL18] + random oracle model (ROM) [BR93] with *sequential signing sessions*, i.e., where the i-th session must be completed before the $(i+1)$-st session is opened. We show that under these model assumptions, the blind Schnorr signature scheme remains secure under the one-more discrete logarithm (OMDL) assumption even when polynomially many signatures are issued for the same public key pk. Our result complements the recent work of Fuchsbauer et al. [FPS20] who showed that the scheme remains secure in the AGM+ROM for *polylogarithmically* many concurrent signing sessions (also under OMDL). More precisely, their result shows that the scheme is secure under the OMDL+ROS assumption, where the ROS problem remains information theoretically hard as long as the number of signing sessions remains polylogarithmic. Hence when combined with the results of Benhamouda et al. [Ben+20], this shows that the blind Schnorr scheme is concurrently secure (in the

[1] Although the attack can be formulated for all the aforementioned blind signature schemes, the algebraic structure in the latter two schemes gives rise to an *efficient attack.*

AGM+ROM) if and only if the signer issues at most polylogarithmically many signatures. We remark that security under sequential signing sessions is still a very meaningful security guarantee and has been explored in prior works (see below). Namely, sequentiality of sessions is easy to ensure (from the signer's perspective) at the expense of some efficiency.

B. Concurrent Security of Abe's Scheme.

In the second part of our work, we revisit the concurrent security properties of Abe's blind signature scheme [Abe01]. This scheme was initially proven secure under the DL assumption in the ROM (with blindness holding computationally under the DDH assumption). However, a later work by Abe and Okhubo [OA03] pointed out that the original proof contained a flaw and gave a security proof in the generic group model (GGM)+ROM instead. To give a more meaningful security guarantee, we show that the scheme can be proven secure *with concurrent signing sessions* under the DL assumption in the far more realistic AGM+ROM. Our reduction follows the original proof strategy of Abe, but uses the AGM to avoid the rewinding step which causes the problem in his proof. This also has the upside of leading to a *tightly secure scheme* with relatively practical parameter sizes.

On Security in the AGM. Fuchsbauer, Kiltz and Loss [FKL18] introduced the *Algebraic Group Model (AGM)* as a means to analyze group based cryptosystems. In the AGM, an adversary must output an explanation of how it computed its output group elements from the group elements in its input. Since its introduction, the AGM has been readily adopted and has served as useful tool to prove the security of schemes that would be too difficult to analyze in the plain model. From a qualitative point of view, proofs in the AGM provide a weaker form of security than proofs in the plain model, but a much stronger one than proofs in the GGM. The recent work of Agrikola et al. [AHK20] shows that results from the AGM can be transferred to the standard model using strong, but falsifiable assumptions. This suggests that proofs in the AGM indeed hold some meaning for the plain model.

C. Experimental Results

Having established theoretical security bounds on these pairing-free schemes in the AGM+ROM, we demonstrate practicality. Concretely, we present AWS EC2 benchmarks of webservers running the blind Schnorr and Abe signature schemes, modeled after the cryptocurrency tumbler described in [HBG16]. We show that the parallel Abe scheme can process anonymity sets of 100 clients in under a second, given enough cores to work with. We also show the cutoff point in workload whereby the sequential Schnorr scheme becomes the more efficient option due to its lower complexity and smaller signature size.

D. Related Work and Discussion

We have already mentioned several works that study the security of blind signatures in the concurrent signer model. In the sequential model, the work of Baldimtsi and Lysyanskaya [BL13a] proves that an enhanced version of Abe's scheme is secure under DL. Pointcheval [Poi98] gave a transformation of the blind Okamoto-Schnorr scheme that remains secure even in a (slightly) relaxed version of the sequential model. Either of these schemes are significantly less efficient than the Schnorr scheme (both in signature size and required computational effort). Fuchsbauer et al. [FPS20] gave a (concurrently secure) scheme under the OMDL and *modified ROS assumption* in the AGM+ROM. The latter assumption asserts the conjectured hardness of an (apparently harder) version of the ROS problem, even given unbounded computing power and polynomially many concurrent signing sessions. Various constructions in the standard model exist (e.g. [Fis06; Gar+11]), but are usually not considered practical.

Open Questions. Our work leaves open the question of what can be proven about both the Abe and Schnorr blind signature schemes in the plain (random oracle) model. Interestingly, Baldimtsi and Lysyanskaya [BL13b] rule out a security proof for the blind Schnorr scheme using standard reduction techniques even in the sequential signing model. Namely, their result excludes such a reduction from a computational hardness assumption even if the signer just issues *a single signature* (which trivially restricts the sessions to being sequential). Another interesting direction for future work could be a more fine grained security analysis (in the AGM+ROM) of the Schnorr scheme in a less restrictive signing model that allows for a low degree of concurrency. Namely, the ROS attack requires a polylogarithmic number of signing sessions to be open *at the same time*. Thus, it might be possible to prove the security of the scheme if, say, up to a constant number of signing sessions may be interleaved at any given point in time.

Regarding Abe's scheme, there might yet be a glimmer of hope that the original proof can be salvaged (currently, it works only if the adversary's success probability is overwhelming).

II. Preliminaries

A. Notation and Security Games

Notation. For positive integer n, we write $[n]$ for $\{1, \ldots, n\}$. We write x_j for the j-th entry of $\overrightarrow{x}$ and write $x \xleftarrow{\$} \mathcal{X}$ to denote that x is drawn uniformly at random from set $\mathcal{X}$. We denote the security parameter with λ.

Security Games. We use the standard notion of (prose-based) *security games* [BR04; Sho04] to present our proofs. We denote the binary output of a game $\mathbf{G}$ with an adversary A as $\mathbf{G}^{\mathsf{A}}$ and say that A *wins* $\mathbf{G}$ if $\mathbf{G}^{\mathsf{A}} = 1$.

B. The Algebraic Group Model

In the following, let pp be public parameters that describe a group $\mathbb{G}$ of prime order q with generator $\mathbf{g}$. We denote the neutral element by ϵ and write all other group elements in bold face. We further write $\mathbb{Z}_q$ for $\mathbb{Z}/q\mathbb{Z}$.

Definition 1 (Algebraic Algorithm)**.** We say that an algorithm A is *algebraic* if, for any group element $\mathbf{y} \in \mathbb{G}$ that it outputs, it also outputs a list of *algebraic coefficients* $\overrightarrow{z} \in \mathbb{Z}_q^t$, i.e.,

$$(\mathbf{y}, \overrightarrow{z}) \xleftarrow{\$} \mathsf{A}(\overrightarrow{\mathbf{x}})$$

such that

$$\mathbf{y} = \prod \mathbf{x}_i^{z_i}$$

We denote this representation by $[\mathbf{y}]_{\overrightarrow{\mathbf{x}}}$. For an adversary A that has access to oracles during its runtime, we impose the above restriction to all group elements that it outputs to an oracle. Similarly, all group elements that A receives through oracle interactions are treated as inputs to A; hence, such group elements become part of $\overrightarrow{\mathbf{x}}$ when A outputs group elements (and hence algebraic coefficients) at a later point.

In the algebraic group model (AGM), all algorithms are treated as algebraic algorithms. Moreover, we define the *running time* of an algorithm A in the AGM as the *number of group operations* that A performs.

C. Hardness Assumption

Definition 2 (Discrete Logarithm Problem (DLP))**.** We define the *discrete logarithm problem (with adversary* A*)* via the following game:
Game $\mathbf{DLP}$:

- **Setup.** $\mathbf{DLP}$ samples $x \xleftarrow{\$} \mathbb{Z}_q$ and runs A on input $\mathbf{g}, \mathbf{U} := \mathbf{g}^x$.
- **Output Determination.** When A outputs x', $\mathbf{DLP}$ returns 1 if $\mathbf{g}^{x'} = \mathbf{U}$. Otherwise, it returns 0.

We define the advantage of A in $\mathbf{DLP}$ as

$$\mathsf{Adv}_{\mathsf{A}}^{\mathbf{DLP}} := \Pr\left[\mathbf{DLP}^{\mathsf{A}} = 1\right]$$

Definition 3 (One-More-Discrete Logarithm Problem (OMDL))**.** We define the *ℓ-one-more-discrete-logarithm problem (with adversary* A*)* via the following game:
Game ℓ-$\mathbf{OMDL}$:

- **Setup.** ℓ-$\mathbf{OMDL}$ initializes $i = 0$ and $C = \emptyset$. It runs A on input $\mathbf{g}$.
- **Online Phase.** A is given access to the following oracles:
  - Oracle $\mathbf{chal}$ takes no input and samples a group element $\mathbf{y} \xleftarrow{\$} \mathbb{G}$. It sets $C := C \cup \{\mathbf{y}\}$ and returns $\mathbf{y}$.
  - Oracle $\mathbf{dlog}$ takes as input a group element $\mathbf{y}$. It returns $\mathrm{dlog}_{\mathbf{g}} \mathbf{y}$. We assume that $\mathbf{dlog}$ can be queried at most ℓ many times.
- **Output Determination.** When A outputs $(\mathbf{y}_i, x_i)_{i=1}^{\ell+1}$, ℓ-$\mathbf{OMDL}$ returns 1 if for all $i \in [\ell + 1]$: $\mathbf{y}_i \in C$, $\mathbf{g}^{x_i} = \mathbf{y}_i$, and $y_i \neq y_j$ for all $j \neq i$. Otherwise, it returns 0.

We define the advantage of A in ℓ-$\mathbf{OMDL}$ as

$$\mathsf{Adv}_{\mathsf{A},\ell}^{\mathbf{OMDL}} := \Pr\left[\ell\text{-}\mathbf{OMDL}^{\mathsf{A}} = 1\right]$$

D. Blind Signature Schemes

In this section, we introduce the syntax and security definitions of blind (three-move) signature schemes [HKL19]. We also give a definition of blindness (in the honest signer model) in the appendix. However, we only focus on one-more-unforgeability in this work; blindness proofs of either schemes can be found in previous works.

Definition 4 (Three-Move Blind Signature Scheme)**.** A three-move blind signature scheme is a tuple of algorithms $\mathsf{BS} = (\mathsf{KeyGen}, \mathsf{Sign} := (\mathsf{Sign}_1, \mathsf{Sign}_2), \mathsf{User} := (\mathsf{User}_1, \mathsf{User}_2), \mathsf{Verify})$ with the following behaviour.

- The randomized *key generation algorithm* KeyGen takes as input the security parameter λ and group description pp, and outputs a public key pk and a secret key sk. We assume for convenience that pk contains pp and sk contains pk.
- The signing algorithm $\mathsf{Sign} := (\mathsf{Sign}_1, \mathsf{Sign}_2)$ is split into two algorithms:
  - The randomized algorithm Sign_1 takes as input a secret key sk and outputs a commitment C as well as a state St_S.
  - The deterministic algorithm Sign_2 takes as input a secret key sk, a state St_S, and a challenge e. It outputs a response R.
- The user algorithm $\mathsf{User} := (\mathsf{User}_1, \mathsf{User}_2)$ is split into two algorithms:
  - The randomized algorithm User_1 takes as input a public key pk, a message m, and a commitment C. It outputs a challenge e and a state St_U.
  - The deterministic algorithm User_2 takes as input a public key pk, a state St_U, and a response R. It outputs a signature σ or $\perp$.
- The deterministic verifier algorithm Verify takes as input a public key pk, a signature σ, and a message m. It outputs either 1 (accept) or 0 (reject).

Definition 5 (One-More-Unforgeability (OMUF))**.** We define the (concurrent) *one-more unforgeability* of a three-move blind signature scheme BS (against an adversary M) via the following game:
Game $\mathbf{OMUF}_{\mathsf{BS}}$:

- **Setup**. $\mathbf{OMUF}_{\mathsf{BS}}$ samples pk, sk via $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{BS.KeyGen}(\mathsf{pp})$. It initializes $\ell_{\mathtt{closed}} := 0$ and runs M on input pk, pp.
- **Online Phase.** M is given access to the oracles $\mathbf{Sign}_1$ and $\mathbf{Sign}_2$ that behave as follows.
  - Oracle $\mathbf{Sign}_1$ takes no input and samples a fresh session identifier $\mathtt{id}$. It sets $\mathtt{session}_{\mathtt{id}} := \mathtt{open}$ and generates $(C_{\mathtt{id}}, St_{\mathtt{id}}) \xleftarrow{\$} \mathsf{BS.Sign}_1(\mathsf{sk})$. Then it returns $C_{\mathtt{id}}$ and $\mathtt{id}$.
  - Oracle $\mathbf{Sign}_2$ takes as input a challenge e and a session identifier $\mathtt{id}$. If $\mathtt{session}_{\mathtt{id}} \neq \mathtt{open}$, it returns $\perp$. Otherwise, it sets $\ell_{\mathtt{closed}} := \ell_{\mathtt{closed}} + 1$ and $\mathtt{session}_{\mathtt{id}} := \mathtt{closed}$. Then it generates the response R via $R \xleftarrow{\$} \mathsf{BS}(\mathsf{sk}, St_{\mathtt{id}}, e)$ and returns R.

- **Output Determination.** When M outputs tuples $(m_1, \sigma_1), ..., (m_k, \sigma_k)$, $\mathbf{OMUF}_{\mathsf{BS}}$ returns 1 if $k \geq \ell_{\mathtt{closed}} + 1$ and for all $i \in [k] : \mathsf{BS.Verify}(\mathsf{pk}, \sigma_i, m) = 1$ and $(m_i, \sigma_i) \neq (m_j, \sigma_j)$ for all $j \neq i$.[2] Otherwise, it returns 0.

We define the advantage of M in $\mathbf{OMUF}_{\mathsf{BS}}$ as

$$\mathsf{Adv}^{\mathbf{OMUF}}_{\mathsf{M,BS}} := \Pr\left[\mathbf{OMUF}^{\mathsf{M}}_{\mathsf{BS}} = 1\right].$$

Definition 6 (Sequential One-More-Unforgeability (SEQ-OMUF))**.** We define the *sequential one-more unforgeability* of a three-move blind signature scheme BS (against an adversary M) via the following game:

Game $\mathbf{SEQ\text{-}OMUF}_{\mathsf{BS}}$:

- **Setup**. $\mathbf{SEQ\text{-}OMUF}_{\mathsf{BS}}$ generates a key pair pk, sk via $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{BS.KeyGen}(\mathsf{pp})$ and sets a flag $\mathtt{open} := 0$ to indicate whether the adversary currently has an open signing session. It initializes $\ell_{\mathtt{closed}} := 0$ and runs M on input pk, pp.
- **Online Phase.** M is given access to the following oracles.
  - Oracle $\mathbf{Sign}_1$: if no session is currently open ($\mathtt{open} = 0$), $\mathbf{Sign}_1$ generates $(C, St) \xleftarrow{\$} \mathsf{BS.Sign}_1$. It sets the flag $\mathtt{open} := 1$ and returns C to the adversary. If there is already an open session ($\mathtt{open} = 1$) it outputs $\perp$.
  - Oracle $\mathbf{Sign}_2$ takes as input a challenge e. If the session is open ($\mathtt{open} = 1$), $\mathbf{Sign}_2$ generates the response R via $\mathsf{BS.Sign}_2(\mathsf{sk}, St, e)$. It sets $\mathtt{open} := 0$ and $\ell_{\mathtt{closed}} := \ell_{\mathtt{closed}} + 1$, and outputs R to the adversary. If the session is not open ($\mathtt{open} = 0$) it outputs $\perp$.
- **Output Determination.** Same as for $\mathbf{OMUF}_{\mathsf{BS}}$.

We define the advantage of M in $\mathbf{SEQ\text{-}OMUF}_{\mathsf{BS}}$ as

$$\mathsf{Adv}^{\mathbf{SEQ\text{-}OMUF}}_{\mathsf{M,BS}} := \Pr\left[\mathbf{SEQ\text{-}OMUF}^{\mathsf{A}}_{\mathsf{BS}} = 1\right].$$

III. Sequential Unforgeability of Schnorr's Blind Signature Scheme

In this section we show that Schnorr's blind signature scheme satisfies sequential one-more unforgeability under the one-more DL assumption in the AGM. We first recall Schnorr's blind signature scheme BSS below.[3] Let $H\colon \{0,1\}^* \to \mathbb{G} \setminus \{\epsilon\}$ be a hash function.

- KeyGen: On input pp, KeyGen samples $x \xleftarrow{\$} \mathbb{Z}_q$ and sets $\mathbf{x} := \mathbf{g}^x$. It sets $\mathsf{sk} := x, \mathsf{pk} := \mathbf{x}$ and returns $(\mathsf{sk}, \mathsf{pk})$.
- Sign_1: On input sk, Sign_1 samples $r \xleftarrow{\$} \mathbb{Z}_q$ and returns the commitment $\mathbf{r} := \mathbf{g}^r$ and the state $St_S := r$.
- Sign_2: On input a secret key sk, a state $St_S = r$ and a challenge c, Sign_2 computes $s := c \cdot \mathsf{sk} + r \mod q$ and returns the response s.
- User_1: On input a public key pk, a commitment $\mathbf{r}$, and a message m, User_1 does the following. It samples first samples $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_q$. Then, it computes $\mathbf{r}' := \mathbf{r} \cdot \mathbf{g}^\alpha \cdot \mathsf{pk}^\beta$ and $c' := H(\mathbf{r}', m), c := c' + \beta \mod q$. It returns the challenge c and the state $St_U := (\mathbf{r}, c, \alpha, \beta, m)$.
- User_2: On input a public key pk, a state $St_U = (\mathbf{r}, c, \alpha, \beta, m)$, and a response s, User_2 first checks if $\mathbf{g}^s = \mathbf{r} \cdot \mathbf{x}^c$ and returns $\perp$ if not. Otherwise, it computes $\mathbf{r}' := \mathbf{r} \cdot \mathbf{g}^\alpha \cdot \mathsf{pk}^\beta$ and $s' := s + \alpha$ and returns the signature $\sigma := (\mathbf{r}', s')$.
- Verify: On input a public key pk, a signature $\sigma = (\mathbf{r}', s')$ and a message m, Verify computes $c' := H(\mathbf{r}', m)$ and checks whether $g^{s'} = \mathbf{r}' \cdot \mathsf{pk}^{c'}$. If so, it returns 1; otherwise, it returns 0.

Theorem 1. *Let M be an algebraic adversary that runs in time t_{M}, makes at most ℓ queries to $\mathbf{Sign}_2$ in $\mathbf{SEQ\text{-}OMUF}_{\mathsf{BSS}}$, and at most q_h random oracle queries to H. Then there exists an adversary B such that*

$$\mathsf{Adv}^{\mathbf{OMDL}}_{\mathsf{B},\ell} \geq \mathsf{Adv}^{\mathbf{SEQ\text{-}OMUF}}_{\mathsf{M,BSS}} - \frac{q_h^2 + q_h + 2}{2q},$$

and B runs in time $t_{\mathsf{B}} \approx t_{\mathsf{M}}$.

We explain the proof idea first. Many of the ideas and notations are reused from [FPS20]; we include them for completeness. Since M can query $\mathbf{Sign}_2$ ℓ times, it is allowed to open $\ell + 1$ sessions and close the first ℓ of them (the last session is never closed). Let $\mathbf{x}$ be the public key, and $\mathbf{r}_1, \ldots, \mathbf{r}_{\ell+1}$ be the group elements returned by $\mathbf{Sign}_1$. Let $(m_1^*, (\mathbf{r}_1^*, s_1^*)), \ldots, (m_{\ell+1}^*, (\mathbf{r}_{\ell+1}^*, s_{\ell+1}^*))$ be M's final outputs, i.e., $(\mathbf{r}_i^*, s_i^*)(i \in [\ell+1])$ is M's forgery on message m_i^*. Since M is algebraic, it also outputs $\mathbf{r}_i^*$'s algebraic representation $(\gamma_i^*, \xi_i^*, \rho_{i,1}^*, \ldots, \rho_{i,\ell+1}^*)$ based on $\mathbf{g}, \mathbf{x}, \mathbf{r}_1, \ldots, \mathbf{r}_{\ell+1}$, i.e.,

$$\begin{aligned}\mathbf{r}_i^* &= \mathbf{g}^{\gamma_i^*} \cdot \mathbf{x}^{\xi_i^*} \cdot \prod_{j=1}^{\ell} \mathbf{r}_j^{\rho_{i,j}^*} \cdot \mathbf{r}_{\ell+1}^{\rho_{i,\ell+1}^*} \\ &= \mathbf{g}^{\gamma_i^* + \rho_{i,\ell+1}^* r_{\ell+1}} \cdot \mathbf{x}^{\xi_i^*} \cdot \prod_{j=1}^{\ell} \mathbf{r}_j^{\rho_{i,j}^*}\end{aligned}$$

(where $r_{\ell+1} = \mathrm{dlog}\, \mathbf{r}_{\ell+1}$). Suppose M wins $\mathbf{SEQ\text{-}OMUF}_{\mathsf{BSS}}$, i.e., $(\mathbf{r}_i^*, s_i^*)$ is a valid forgery on message m_i^* and we have that

$$\mathbf{g}^{s_i^*} = \mathbf{r}_i^* \cdot \mathbf{x}^{c_i^*}, \tag{1}$$

where $c_i^* = H(\mathbf{r}_i^*, m_i^*)$, for all $i \in [\ell + 1]$. The two equations above combined yield

$$\mathbf{x}^{c_i^* + \xi_i^*} \cdot \prod_{j=1}^{\ell} \mathbf{r}_j^{\rho_{i,j}^*} = \mathbf{g}^{s_i^* - \gamma_i^* - \rho_{i,\ell+1}^* r_{\ell+1}}. \tag{2}$$

The reduction to ℓ-$\mathbf{OMDL}$, B, works as follows. B queries its challenge oracle $\ell+1$ times to obtain $\mathbf{x}, \mathbf{r}_1, \ldots, \mathbf{r}_\ell$, samples $r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_q$ and sets $\mathbf{r}_{\ell+1} \leftarrow \mathbf{g}^{r_{\ell+1}}$, and simulates $\mathbf{Sign}_1()$ by returning $\mathbf{r}_i$. To simulate $\mathbf{Sign}_2(c_j)$ queries, B queries its dlog oracle and returns $s_j := \mathsf{dlog}(\mathbf{g}, \mathbf{r}_j \cdot \mathbf{x}^{c_j})$. Substituting the definition of s_j into Eq. (2), we get

$$\mathbf{x}^{c_i^* + \xi_i^* - \sum_{j=1}^{\ell} \rho_{i,j}^* c_j} = g^{s_i^* - \gamma_i^* - \sum_{j=1}^{\ell} \rho_{i,j}^* s_j - \rho_{i,\ell+1}^* r_{\ell+1}},$$

[2] This is the "strong" variant of OMUF. We use this variant throughout this work. Since a strong-OMUF adversary has more winning conditions than a weak-OMUF adversary, our results here are strengthened.

[3] We use different letters to denote the variables in the scheme than what we used in the previous section. Our choices are in line with the standard notation for this scheme.

which can be used to compute $x = \mathrm{dlog}\,\mathbf{x}$ as long as $\chi_i = c_i^* + \xi_i^* - \sum_{j=1}^{\ell} \rho_{i,j}^* c_j \neq 0$ for some i.

Now we need to upper bound the probability that $\chi_i = 0$ for *all* $i = 1, \ldots, \ell+1$. Recall that in [FPS20] this is reduced to the ℓ-ROS problem (which can be solved in polynomial time when $\ell \geq \lambda$, as shown in the recent work of [Ben+20]). Here, since we are in the sequential setting where the adversary must close one session before opening another, we can make a statistical argument instead.

For each message/forgery pair $(m_i^*, (\mathbf{r}_i^*, s_i^*))$, there is a corresponding random oracle query $H(\mathbf{r}_i^*, m_i^*)$. (If M does not make such a query, then $\Pr[\chi_i = 0] = 1/q$.) Call such query the *i-th special query*. Any special query is made during a session which is eventually closed (i.e., between M's j-th $\mathbf{Sign}_1$ query and j-th $\mathbf{Sign}_2$ query for some $j \in [\ell]$), or between two sessions (including before the first session), or during the last session which is never closed. If there is *any* special query (say the i-th) made between two sessions or during the last session, then all coefficients in χ_i's expression, except c_i^*, are fixed when M makes its i-th special query. On the other hand, c_i^* is a uniformly random element of $\mathbb{Z}_q$. Therefore, $\Pr[\chi_i = 0] = 1/q$ for a single $H(\mathbf{r}_i^*, m_i^*)$ query. Otherwise, i.e., if *all* special queries are made during some session which is eventually closed, since there are ℓ such sessions and $\ell+1$ special queries, there is at least one session (say the j_0-th) with at least two special queries (say the i-th and $(i+1)$-th) during it. At the time when M makes its $(i+1)$-th special query, i.e., when c_{i+1}^* is chosen at random from $\mathbb{Z}_q$, all coefficients in both χ_i and χ_{i+1}'s expression, except c_{i+1}^* and c_{j_0}, are fixed. Therefore, at this time whether M can come up with a c_{j_0} s.t. $\chi_i = \chi_{i+1} = 0$ is already determined, and it depends on the random choice of c_{i+1}^*. It can be shown that there is at most one c_{i+1}^* s.t. the linear system $\chi_i = \chi_{i+1} = 0$ (with unknown c_{j_0}) has a solution; therefore, $\Pr[\chi_i = \chi_{i+1} = 0] \leq 1/q$ for a single pair of $H(\mathbf{r}_i^*, m_i^*)$ and $H(\mathbf{r}_{i+1}^*, m_{i+1}^*)$ queries.[4]

Proof. Let M be as in the theorem statement. Without loss of generality, we assume that M makes exactly $\ell+1$ $\mathbf{Sign}_1()$ and exactly ℓ $\mathbf{Sign}_2$ queries, and returns exactly $\ell+1$ valid signatures $(\mathbf{r}_1^*, s_1^*), \ldots, (\mathbf{r}_{\ell+1}^*, s_{\ell+1}^*)$ of messages $m_1^*, \ldots, m_{\ell+1}^*$.[5] We further assume that pairs $(m_1^*, \mathbf{r}_1^*), \ldots, (m_{\ell+1}^*, \mathbf{r}_{\ell+1}^*)$ are all distinct; otherwise M could not win $\mathbf{SEQ\text{-}OMUF}_{\mathsf{BSS}}$.[6] Let $\mathbf{x}$ be the public key, $\mathbf{r}_1, \ldots, \mathbf{r}_{\ell+1}$ be the group elements returned by $\mathbf{Sign}_1$, and M's $\mathbf{Sign}_2$ queries be $\mathbf{Sign}_2(c_1), \ldots, \mathbf{Sign}_2(c_\ell)$. The proof goes by a sequence of games, which we describe below. For convenience, we set $\mathsf{Adv}_{\mathsf{M}}^{\mathbf{G_i}} := \Pr[\mathbf{G}_{\mathbf{i}}^{\mathsf{M}} = 1]$.

Game $\mathbf{G_0}$. This is the SEQ-OMUF game. We have that

$$\mathsf{Adv}_{\mathsf{M}}^{\mathbf{G_0}} = \mathsf{Adv}_{\mathsf{M},\mathsf{BSS}}^{\mathbf{SEQ\text{-}OMUF}}.$$

Game $\mathbf{G_1}$. In $\mathbf{G_1}$ we make the following change. When M returns its final outputs $(m_1^*, (\mathbf{r}_1^*, s_1^*)), \ldots, (m_{\ell+1}^*, (\mathbf{r}_{\ell+1}^*, s_{\ell+1}^*))$, together with $\mathbf{r}_i^*$'s algebraic representation $(\gamma_i^*, \xi_i^*, \rho_{i,1}^*, \ldots, \rho_{i,\ell+1}^*)$ based on $\mathbf{g}, \mathbf{x}, \mathbf{r}_1, \ldots, \mathbf{r}_{\ell+1}$, for each $i \in [\ell+1]$ for which $H(\mathbf{r}_i^*, m_i^*)$ is undefined, we emulate a query $c_i^* := H(\mathbf{r}_i^*, m_i^*)$ via lazy sampling. After that, we define $\chi_i := c_i^* + \xi_i^* - \sum_{j=1}^{\ell} \rho_{i,j}^* c_j$, and abort if $\chi_i = 0$ for all i. (Note that $\rho_{i,\ell+1}^*$ does not appear in the definition of χ_i.)

$\mathbf{G_1}$ and $\mathbf{G_0}$ are identical unless $\chi_i = 0$ for all $i \in [\ell+1]$. Call this event E.

Claim 1. $\Pr[E] \leq \frac{q_h^2 + q_h + 2}{2q}$

Proof. If M does not query $H(\mathbf{r}_i^*, m_i^*)$ for some i, then c_i^* is a uniformly random element of $\mathbb{Z}_q$ in M's view, so $\Pr[\chi_i = 0] = 1/q$.

Next we assume that M queries $H(\mathbf{r}_i^*, m_i^*)$ for all i; call such query the *i-th special query*. Since $(m_i^*, \mathbf{r}_i^*)$ pairs are all distinct, $c_i^* = H(\mathbf{r}_i^*, m_i^*)$ is a uniformly random element of $\mathbb{Z}_q$ (independent of everything else) when M makes the i-th special query. Also, $\mathbf{r}_i^*$'s algebraic representation $(\gamma_i^*, \xi_i^*, \rho_{i,1}^*, \ldots, \rho_{i,\ell+1}^*)$ is already determined when M makes its i-th special query. Any special query is made either during a session which is eventually closed (i.e., between M's j-th $\mathbf{Sign}_1$ query and j-th $\mathbf{Sign}_2$ query for some $j \in [\ell]$), or between two sessions (including before the first session), or during the last session which is never closed (i.e., after M's $(\ell+1)$-th $\mathbf{Sign}_1$ query). We consider these cases separately:

Case C_1. Suppose that there is *any* special query (say the i-th) made (a) between two sessions (including before the first session); say the i-th special query is made after the j_0-th $\mathbf{Sign}_2$ query and before the (j_0+1)-th $\mathbf{Sign}_1$ query, or (b) after the $(\ell+1)$-th $\mathbf{Sign}_1$ query. Consider the time when M makes its i-th special query $H(\mathbf{r}_i^*, m_i^*)$. In case (a), at this point all group elements M has seen are $\mathbf{g}, \mathbf{x}, \mathbf{r}_1, \ldots, \mathbf{r}_{j_0}$, so $\rho_{i,j_0+1}^* = \ldots = \rho_{i,\ell}^* = 0$; furthermore, the algebraic coefficients (for $\mathbf{r}_i^*$) $\xi_i^*, \rho_{i,1}^*, \ldots, \rho_{i,j_0}^*$ are all fixed. Finally, $c_j (j \in [j_0])$ is fixed when M makes its j-th $\mathbf{Sign}_2$ query, which happens *before* M's i-th special query. Similarly, in case (b), at this point the algebraic coefficients (for $\mathbf{r}_i^*$) $\xi_i^*, \rho_{i,1}^*, \ldots, \rho_{i,\ell+1}^*$ are all fixed, and $c_1, \ldots, c_\ell$ are fixed when M makes its ℓ-th $\mathbf{Sign}_2$ query, which happens *before* M's i-th special query. This means that in both cases (a) and (b), all coefficients in χ_i's expression, except c_i^*, are fixed when M makes its i-th special query. On the other hand, c_i^* is a uniformly random element in $\mathbb{Z}_q$. Therefore,

$$\Pr[\chi_i = c_i^* + \xi_i^* - \sum_{j=1}^{j_0} \rho_{i,j}^* c_j = 0] = \frac{1}{q}$$

[4] We remark that this is essentially the 1-ROS problem, which is statistically hard.

[5] Since the security game is *sequential* OMUF, and M can make at most ℓ $\mathbf{Sign}_2$ queries, this implies that M can make at most $\ell+1$ $\mathbf{Sign}_1$ queries. Obviously, any adversary who makes less than $\ell+1$ $\mathbf{Sign}_1$ queries, or less than ℓ $\mathbf{Sign}_2$ queries, or returns more than $\ell+1$ valid signatures, can be turned into an adversary who makes exactly $\ell+1$ $\mathbf{Sign}_1$ and exactly ℓ $\mathbf{Sign}_2$ queries, and returns exactly $\ell+1$ valid signatures, with the same advantage and roughly the same running time.

[6] Suppose $(m_i^*, r_i^*) = (m_j^*, r_j^*)$ for $i \neq j \in [\ell+1]$. If $s_i^* = s_j^*$ then M outputs two identical message/signature pairs, violating the winning condition. Otherwise it cannot be the case that both (r_i^*, s_i^*) and (r_i^*, s_j^*) are both valid signatures of m_i^*, since given m_i^* and r_i^*, s_i^* as in the valid signature is uniquely defined (as in Eq. (1)).

for a single $H(\mathbf{r}_i^*, m_i^*)$ query. Since M makes q_h random oracle queries in total, we have that $\Pr[\chi_i = 0 \wedge C_1] \leq \frac{q_h}{q}$, and hence $\Pr[E \wedge C_1] \leq \frac{q_h}{q}$.

Case C_2. Suppose that *all* special queries are made during some session which is eventually closed. Since there are ℓ such sessions and $\ell + 1$ special queries, there is at least one session with at least two special queries during it; say the i-th and $(i+1)$-th special queries are made during the j_0-th session. Consider the time when M makes its $(i+1)$-th special query. At this point all group elements M has seen are $\mathbf{g}, \mathbf{x}, \mathbf{r}_1, \ldots, \mathbf{r}_{j_0}$, so $\rho^*_{i,j_0+1} = \ldots = \rho^*_{i,\ell} = 0$; furthermore, the algebraic coefficients (for $\mathbf{r}_i^*$ and $\mathbf{r}_{i+1}^*$) $\xi_i^*, \rho^*_{i,1}, \ldots, \rho^*_{i,j_0}, \xi^*_{i+1}, \rho^*_{i+1,1}, \ldots, \rho^*_{i+1,j_0}$ are all fixed. The output of M's i-th special query c_i^* is also fixed right after M makes its i-th special query, which happens *before* M's $(i+1)$-th special query. Finally, $c_j (j \in [j_0 - 1])$ is fixed when M makes its j-th $\mathbf{Sign}_2$ query, which again happens *before* M's $(i+1)$-th special query. (This is because M's $(i+1)$-th special query is made during the j_0-th session, which is started after the j-th session is closed.) This means that all coefficients in χ_i and χ_{i+1}'s expressions, except c_{j_0} and c^*_{i+1}, are fixed when M makes its $(i+1)$-th special query.

Next consider the time when M makes its j_0-th $\mathbf{Sign}_2$ query (i.e., when the j_0-th session is closed). At this point c^*_{i+1} is also fixed, so the only coefficient in χ_i and χ_{i+1}'s expressions which is not fixed is c_{j_0} (to be chosen by M). In sum, the last coefficient fixed is c_{j_0} (chosen by M), and the second last coefficient fixed is c^*_{i+1} (uniformly random in $\mathbb{Z}_q$).

Consider the linear system of unknown c_{j_0}

$$\begin{cases} \chi_i = c_i^* + \xi_i^* - \sum_{j=1}^{j_0} \rho^*_{i,j} c_j = 0, \\ \chi_{i+1} = c^*_{i+1} + \xi^*_{i+1} - \sum_{j=1}^{j_0} \rho^*_{i+1,j} c_j = 0, \end{cases} \tag{3}$$

we have that

$$\begin{aligned}
&\Pr[\chi_i = \chi_{i+1} = 0] \\
=&\Pr[c_{j_0} \text{ is the solution of (3)}] \\
\leq&\Pr[\text{(3) has a solution}] \\
=&\Pr\left[\begin{array}{l} \underbrace{\operatorname{rank}\begin{pmatrix} \rho^*_{i,j_0} & c_i^* + \xi_i^* - \sum_{j=1}^{j_0-1} \rho^*_{i,j} c_j \\ \rho^*_{i+1,j_0} & c^*_{i+1} + \xi^*_{i+1} - \sum_{j=1}^{j_0-1} \rho^*_{i+1,j} c_j \end{pmatrix}}_{\text{augmented matrix of (3)}} \\ = \operatorname{rank} \underbrace{\begin{pmatrix} \rho^*_{i,j_0} \\ \rho^*_{i+1,j_0} \end{pmatrix}}_{\text{coefficient matrix of (3)}} \end{array}\right] \\
\leq&\Pr\left[\operatorname{rank}\begin{pmatrix} \rho^*_{i,j_0} & c_i^* + \xi_i^* - \sum_{j=1}^{j_0-1} \rho^*_{i,j} c_j \\ \rho^*_{i+1,j_0} & c^*_{i+1} + \xi^*_{i+1} - \sum_{j=1}^{j_0-1} \rho^*_{i+1,j} c_j \end{pmatrix} \leq 1\right] \\
=&\Pr\left[\begin{vmatrix} \rho^*_{i,j_0} & c_i^* + \xi_i^* - \sum_{j=1}^{j_0-1} \rho^*_{i,j} c_j \\ \rho^*_{i+1,j_0} & c^*_{i+1} + \xi^*_{i+1} - \sum_{j=1}^{j_0-1} \rho^*_{i+1,j} c_j \end{vmatrix} = 0\right] \\
=&\Pr\left[\begin{array}{l} \rho^*_{i,j_0} c^*_{i+1} + \rho^*_{i,j_0}(\xi^*_{i+1} - \sum_{j=1}^{j_0-1} \rho^*_{i+1,j} c_j) \\ -\rho^*_{i+1,j_0}(c_i^* + \xi_i^* - \sum_{j=1}^{j_0-1} \rho^*_{i,j} c_j) = 0 \end{array}\right] \\
=&\frac{1}{q}
\end{aligned}$$

for a single pair of $H(\mathbf{r}_i^*, m_i^*)$ and $H(\mathbf{r}^*_{i+1}, m^*_{i+1})$ queries. (The last equation is because when M makes its $(i+1)$-th special query, c^*_{i+1} is a uniformly random element of $\mathbb{Z}_q$, and all other coefficients are fixed.) Since M makes q_h random oracle queries in total, we have that $\Pr[\chi_i = \chi_{i+1} = 0 \wedge C_2] \leq \frac{\binom{q_h}{2}}{q}$, and hence $\Pr[E \wedge C_2] \leq \frac{\binom{q_h}{2}}{q}$.

In sum, we have that (let case C_0 be "M does not make the i-th special query for some $i \in [\ell + 1]$")

$$\begin{aligned}
\Pr[E] &= \Pr[E \wedge C_0] + \Pr[E \wedge C_1] + \Pr[E \wedge C_2] \\
&\leq \frac{1}{q} + \frac{q_h}{q} + \frac{\binom{q_h}{2}}{q} = \frac{q_h^2 + q_h + 2}{2q}.
\end{aligned}$$

□

By the claim, $\mathsf{Adv}^{\mathbf{G_1}}_{\mathsf{M}} \leq \mathsf{Adv}^{\mathbf{G_0}}_{\mathsf{M}} - \frac{q_h^2+q_h+2}{2q}$.

Reduction to OMDL. We now upper bound $\mathsf{Adv}^{\mathbf{G_1}}_{\mathsf{M}}$ via a reduction B solving the OMDL problem. B runs on input $(\mathbb{G}, \mathbf{g}, q)$, and is given oracle access to $\mathbf{chal}$ and $\mathbf{dlog}$. B first queries $\mathbf{x} \coloneqq \mathbf{chal}()$ and runs $\mathsf{M}(\mathbb{G}, \mathbf{g}, q, \mathbf{x})$. B runs the code of $\mathbf{G_1}$ except that (1) on M's j-th $\mathbf{Sign}_1$ query ($j \in [\ell]$), B returns $\mathbf{r}_j \coloneqq \mathbf{chal}()$; (2) on M's j-th $\mathbf{Sign}_2$ query, B returns $s_j \coloneqq \mathbf{dlog}(\mathbf{g}, \mathbf{r}_j \cdot \mathbf{x}^{c_j})$. ($\mathsf{B}$ answers M's $(\ell+1)$-th $\mathbf{Sign}_1$ query just as in $\mathbf{G_1}$, i.e., by sampling $r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_q$ and returning $\mathbf{r}_{\ell+1} \coloneqq \mathbf{g}^{r_{\ell+1}}$.) Finally, when M returns its final outputs, if there exists an $i \in [\ell + 1]$ s.t. $\chi_i \neq 0$, B computes

$$x \coloneqq \frac{s_i^* - \gamma_i^* - \sum_{j=1}^{\ell} \rho^*_{i,j} s_j - \rho^*_{i,\ell+1} r_{\ell+1}}{\chi_i}$$

and

$$r_j \coloneqq s_j - c_j x,$$

and outputs $(x, r_1, \ldots, r_\ell)$. (If $\chi_i = 0$ for all i, B aborts.)

Clearly, B runs in time $t_{\mathsf{M}} + O(\ell + q_h)$. We claim that B wins the OMDL game if M wins $\mathbf{G_1}$. Since M is algebraic, we have that

$$\begin{aligned}
\mathbf{r}_i^* &= \mathbf{g}^{\gamma_i^*} \cdot \mathbf{x}^{\xi_i^*} \cdot \prod_{j=1}^{\ell} \mathbf{r}_j^{\rho^*_{i,j}} \cdot \mathbf{r}_{\ell+1}^{\rho^*_{i,\ell+1}} \\
&= \mathbf{g}^{\gamma_i^* + \rho^*_{i,\ell+1} r_{\ell+1}} \cdot \mathbf{x}^{\xi_i^*} \cdot \prod_{j=1}^{\ell} \mathbf{r}_j^{\rho^*_{i,j}}.
\end{aligned}$$

On the other hand, since M wins $\mathbf{G_1}$, i.e., $(\mathbf{r}_i^*, s_i^*)$ is a valid forgery on message m_i^*, we have that

$$\mathbf{g}^{s_i^*} = \mathbf{r}_i^* \cdot \mathbf{x}^{c_i^*}.$$

The two equations above combined yield

$$\mathbf{x}^{c_i^* + \xi_i^*} \cdot \prod_{j=1}^{\ell} \mathbf{r}_j^{\rho^*_{i,j}} = \mathbf{g}^{s_i^* - \gamma_i^* - \rho^*_{i,\ell+1} r_{\ell+1}}. \tag{4}$$

By definition of s_j, we have that

$$\mathbf{r}_j = \frac{\mathbf{g}^{s_j}}{\mathbf{x}^{c_j}}, \tag{5}$$

substituting (5) into (4), we get

$$\mathbf{x}^{\chi_i} = \mathbf{x}^{c_i^* + \xi_i^* - \sum_{j=1}^{\ell} \rho_{i,j}^* c_j} = \mathbf{g}^{s_i^* - \gamma_i^* - \sum_{j=1}^{\ell} \rho_{i,j}^* s_j - \rho_{i,\ell+1}^* r_{\ell+1}},$$

so $x = \operatorname{dlog} \mathbf{x}$. By (5) again, $r_j = \operatorname{dlog} \mathbf{r}_j$. This means that B wins the OMDL game. We have that

$$\mathsf{Adv}^{\mathbf{OMDL}}_{\mathsf{B},\ell} = \mathsf{Adv}^{\mathbf{G_1}}_{\mathsf{M}}.$$

We conclude that

$$\mathsf{Adv}^{\mathbf{OMDL}}_{\mathsf{B},\ell} \geq \mathsf{Adv}^{\mathbf{SEQ\text{-}OMUF}}_{\mathsf{M},\mathsf{BSS}} - \frac{q_h^2 + q_h + 2}{2q},$$

completing the proof. □

IV. Abe's Blind Signature Scheme

We begin by describing Abe's blind signature scheme BSA [Abe01]. Let again $\mathbb{G}$ be a group of order q with generator $\mathbf{g}$ described by public parameters pp. Let $H_1 \colon \{0,1\}^* \to \mathbb{G} \setminus \{\epsilon\}$, $H_2 \colon \{0,1\}^* \to \mathbb{G} \setminus \{\epsilon\}$, $H_3 \colon \{0,1\}^* \to \mathbb{Z}_q$ be hash functions.

- KeyGen: On input pp, KeyGen samples $\mathbf{h} \xleftarrow{\$} \mathbb{G}$, $x \xleftarrow{\$} \mathbb{Z}_q$ and sets $\mathbf{y} := \mathbf{g}^x$, $\mathbf{z} := H_1(\mathsf{pp}, \mathbf{g}, \mathbf{h}, \mathbf{y})$. It sets $\mathsf{sk} := x$, $\mathsf{pk} := (\mathbf{g}, \mathbf{h}, \mathbf{y}, \mathbf{z})$ and returns $(\mathsf{sk}, \mathsf{pk})$.
- Sign_1: On input sk, Sign_1 samples $\mathrm{rnd} \xleftarrow{\$} \{0,1\}^\lambda$ and $u, d, s_1, s_2 \xleftarrow{\$} \mathbb{Z}_q$. It computes $\mathbf{z}_1 := H_2(\mathrm{rnd})$, $\mathbf{z}_2 := \mathbf{z}/\mathbf{z}_1$, $\mathbf{a} := \mathbf{g}^u$, $\mathbf{b_1} := \mathbf{g}^{s_1} \cdot \mathbf{z}_1^d$, $\mathbf{b_2} := \mathbf{h}^{s_2} \cdot \mathbf{z}_2^d$. It returns a commitment $(\mathrm{rnd}, \mathbf{a}, \mathbf{b_1}, \mathbf{b_2})$ and a state (u, d, s_1, s_2).
- Sign_2: On input a secret key sk, a challenge e, and state $St_S = (s_1, s_2, u, d)$, Sign_2 computes $c := e - d \mod q$, $r := u - c \cdot \mathsf{sk} \mod q$ and returns the response (c, d, r, s_1, s_2).
- User_1: On input a public key pk and a commitment $(\mathrm{rnd}, \mathbf{a}, \mathbf{b_1}, \mathbf{b_2})$, and message m, User_1 does the following. It samples $\gamma \xleftarrow{\$} \mathbb{Z}_q^*$ and $\tau, t_1, t_2, t_3, t_4, t_5 \xleftarrow{\$} \mathbb{Z}_q$. Then, it computes $\mathbf{z}_1 := H_2(\mathrm{rnd})$, $\alpha := \mathbf{a} \cdot \mathbf{g}^{t_1} \cdot \mathbf{y}^{t_2}$, $\zeta := \mathbf{z}^\gamma$, $\zeta_1 := \mathbf{z}_1^\gamma$, $\zeta_2 := \zeta/\zeta_1$. Next, it sets $\beta_1 := \mathbf{b}_1^\gamma \cdot \mathbf{g}^{t_3} \cdot \zeta_1^{t_4}$, $\beta_2 := \mathbf{b}_2^\gamma \cdot \mathbf{h}^{t_5} \cdot \zeta_2^{t_4}$, $\eta := \mathbf{z}^\tau$, and $h := H_3(\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta, m)$. Finally, it computes a challenge $e := h - t_2 - t_4 \mod q$, the state $St_U := (\gamma, \tau, t_1, t_2, t_3, t_4, t_5, m)$ and returns e, St_U.
- User_2: On input a public key pk, a response (c, d, r, s_1, s_2) and a state $(\gamma, \tau, t_1, t_2, t_3, t_4, t_5, m)$, User_2 first computes $\rho := r + t_1$, $\omega := c + t_2$, $\sigma_1 := \gamma \cdot s_1 + t_3$, $\sigma_2 := \gamma \cdot s_2 + t_5$, and $\delta := d + t_4$. Then, it computes $\mu := \tau - \delta \cdot \gamma$ and $\varepsilon := H_3(\zeta, \zeta_1, \mathbf{g}^\rho \mathbf{y}^\omega, \mathbf{g}^{\sigma_1} \zeta_1^\delta, \mathbf{h}^{\sigma_2} \zeta_2^\delta, \mathbf{z}^\mu \zeta^\delta, m)$. It returns the signature $\sigma := (\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu)$ if $\delta + \omega = \varepsilon$; otherwise, it returns $\perp$.
- Verify: On input a public key pk, a signature $(\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu)$ and a message m, Verify computes $\varepsilon := H_3(\zeta, \zeta_1, \mathbf{g}^\rho \mathbf{y}^\omega, \mathbf{g}^{\sigma_1} \zeta_1^\delta, \mathbf{h}^{\sigma_2} \zeta_2^\delta, \mathbf{z}^\mu \zeta^\delta, m)$. It returns 1 if $\delta + \omega = \varepsilon$; otherwise, it returns 0.

By [Abe01], the scheme is computationally blind under the Decisional Diffie-Hellman Assumption.

In the following, we provide a proof for the one-more-unforgeability. Similar to [Abe01] we do this in two steps. First, we show that it is infeasible for an adversary to generate a signature that does not use a tag that corresponds to a closed signing session (as the scheme is only computationally blind, signatures can be linked to signing sessions through so-called *tags*). This corresponds to Abe's restrictive blinding lemma. Then, as the main theorem, we show that it is also infeasible for an adversary to win the OMUF game by providing two signatures corresponding to the same closed signing session.

Our techniques. The main idea for both the lemma and the theorem is to use the algebraic representations of the group elements submitted to the random oracle H_3 in combination with the corresponding signature to compute the discrete logarithm of either $\mathbf{y}$ or $\mathbf{h}$ or $\mathbf{z}$. This fails either when the adversary has not made a hash query for the signature in question, or when the representation of the hash query does not contain more information than the signature, i.e., the exponents in the representation already match the signature. We show that both of these cases only occur with a negligible probability. We simulate the protocol in two different ways. One way is to use the secret key x like an honest signer and try to extract the discrete logarithm of $\mathbf{h}$ or $\mathbf{z}$. The other way is to program the random oracles H_1 and H_2 so that the reduction can use the discrete logarithms of $\mathbf{z}, \mathbf{z}_1, \mathbf{z}_2$ to simulate the other side of the OR-proof for extraction of the secret key. We also use the programming of the random oracles to efficiently compute which signature is the "forgery". This, in combination with not having to run the protocol twice for forking, renders a tight proof.

A. The Restrictive Blinding Lemma

We first provide a reduction for the restrictive blinding lemma in the AGM + ROM. We therefore define the game RB-OMUF below.

Game $\mathbf{RB\text{-}OMUF}^{\mathsf{M}}_{\mathsf{BSA}}$:

- **Setup:** $\mathbf{RB\text{-}OMUF}$ generates a key pair via $(\mathsf{sk}, \mathsf{pk}) := (x, (\mathbf{g}, \mathbf{h}, \mathbf{y}, \mathbf{z}))$ via $\mathsf{BSA.KeyGen}(1^\lambda)$.
- **Online Phase:** M is given access to oracles $\mathbf{Sign}_1, \mathbf{Sign}_2$ that emulate the behavior of the honest signer in BS. In addition, it is given access to random oracles H_1, H_2, H_3. Let ℓ denote the number of interactions that M completes with oracle $\mathbf{Sign}_2$ in this phase.
- **Output Determination:** When M outputs a list L of tuples $(m_1, \mathsf{sig}_1), \ldots, (m_L, \mathsf{sig}_L)$, $\mathbf{RB\text{-}OMUF}_{\mathsf{BSA}}$ acts as follows:
  - If the list contains a tuple (m, sig) s.t. $\mathsf{Verify}(\mathsf{pk}, m, \mathsf{sig}) = 0$, or does not contain $\ell + 1$ pairwise-distinct tuples, it returns 0.
  - Let $\mathbf{z}_{1,j}$ denote the value of $\mathbf{z}_1$ used in the j-th invocation of $\mathbf{Sign}_1$. If there exists $(m, \mathsf{sig}) \in L$ with signature components $\zeta \neq \zeta_1$ (equivalently, $\zeta_2 \neq \epsilon$), s.t. for all j whose sessions were closed with an invocation of $\mathbf{Sign}_2$, $\zeta^{\operatorname{dlog}_{\mathbf{z}} \mathbf{z}_{1,j}} \neq \zeta_1$, then $\mathbf{RB\text{-}OMUF}_{\mathsf{BSA}}$ returns 1. Otherwise, it returns 0. We call the first signature in L with these mismatched tags the *special signature*

We define $\mathsf{Adv}^{\mathbf{RB\text{-}OMUF}}_{\mathsf{M},\mathsf{BSA}} := \Pr[\mathbf{RB\text{-}OMUF}^{\mathsf{M}}_{\mathsf{BSA}} = 1]$. We show that an algebraic forger M that wins $\mathbf{RB\text{-}OMUF}_{\mathsf{BSA}}$

can be used to break the discrete logarithm assumption. This reduction is tight and does not require rewinding of the adversary.

Lemma 1 (Restrictive Blinding, see Lemma 3 in [Abe01])**.** *Let* M *be an algebraic algorithm that runs in time* t_{M}*, makes at most* ℓ *queries to oracle* Sign_2 *in* $\mathbf{RB\text{-}OMUF}_{\mathsf{BSA}}$ *and at most (total)* q_h *queries to* H_1, H_2, H_3*. Then, in the random oracle model, there exists an algorithm* B *s.t.*

$$\mathsf{Adv}^{\mathbf{DLP}}_{\mathsf{B}} \geq \frac{1}{4}\mathsf{Adv}^{\mathbf{RB\text{-}OMUF}}_{\mathsf{M},\mathsf{BSA}} - \frac{q_h}{q}$$

and B *runs in time*

$$t_{\mathsf{B}} \approx t_{\mathsf{M}}.$$

Proof. Let M be as in the lemma statement. As before, we assume w.l.o.g. that M makes exactly ℓ queries to Sign_2 and returns a list of $\ell + 1$ tuples. The proof goes by a series of games, which we describe below.

Game $\mathbf{G_0}$. This is the original $\mathbf{RB\text{-}OMUF}_{\mathsf{BSA}}$ game.

Game $\mathbf{G_1}$. To define Game $\mathbf{G_1}$, we first define the following event E_1. E_1 happens if M returns a list L of $\ell + 1$ valid signatures on distinct messages $m_1, ..., m_\ell$ and there exists $(m, \mathsf{sig}) = (m, (\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu)) \in L$ s.t. for all j whose sessions were closed with an invocation of $\mathbf{Sign}_2$, $\zeta^{\mathrm{dlog}_{\mathbf{z}}\mathbf{z}_{1,j}} \neq \zeta_1 \wedge \zeta_2 \neq \epsilon$ and M did not make a query of the form $H_3(\zeta, \zeta_1, \mathbf{g}^\rho\mathbf{y}^\omega, \mathbf{g}^{\sigma_1}\zeta_1^\delta, \mathbf{h}^{\sigma_2}\zeta_2^\delta, \mathbf{z}^\mu\zeta^\delta, m)$. In the following, we refer to the first tuple $(m, \mathsf{sig}) \in L$ as *the special tuple* for convenience. $\mathbf{G_1}$ is identical to game $\mathbf{G_0}$, except that it aborts when E_1 happens.

Claim 2. $\Pr[E_1] = \frac{\ell+1}{q}$

Proof. The only way for an adversary to succeed without querying H_3 for the signature is by guessing the hash value $\varepsilon = \omega + \delta$. Since there are $\ell + 1$ valid signatures in L, the probability of guessing ε correctly for one of them is $\frac{\ell+1}{q}$. □

By the claim, we have that $\mathsf{Adv}^{\mathbf{G_1}}_{\mathsf{M}} \geq \mathsf{Adv}^{\mathbf{G_0}}_{\mathsf{M}} - \frac{\ell+1}{q}$.

Game $\mathbf{G_2}$. Game $\mathbf{G_2}$ is identical to $\mathbf{G_1}$, except that it keeps track of the algebraic representations of group elements submitted to H_3 by M and aborts if a certain event E_2 happens. In the following, we define the event E_2 which depends on these representations.

Simplifying Notations. For each query to H_3, the adversary M submits a set of group elements $\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta$ along with a message m. As M is algebraic, is also provides a representation of these group elements to the basis of elements $\mathbf{g}, \mathbf{h}, \mathbf{y}, \mathbf{z}, \overrightarrow{\mathbf{a}}, \overrightarrow{\mathbf{b}_1}, \overrightarrow{\mathbf{b}_2}, \overrightarrow{\mathbf{z}_1}$ that is has previously obtained via calls to $H_1, H_2, \mathbf{Sign}_1$, or $\mathbf{Sign}_2$. Any element $\mathbf{a}, \mathbf{b}_1, \mathbf{b}_2$ that was returned as reply to a query to $\mathbf{Sign}_1$ can be represented as $\mathbf{a} = \mathbf{y}^c \cdot \mathbf{g}^r, \mathbf{b}_1 = \mathbf{z}_1^d \cdot \mathbf{g}^{s_1}, \mathbf{b}_2 = \mathbf{z}_2^d \cdot \mathbf{h}^{s_2}$. Here, $\mathbf{z}_1, \mathbf{z}_2 = \mathbf{z}/\mathbf{z}_1$ correspond to the call $H_2(\mathrm{rnd})$ made as part of answering this query to $\mathbf{Sign}_1$. This allows us to convert any representation provided by M into a *reduced representation* in the (simpler) basis $\mathbf{g}, \mathbf{h}, \mathbf{y}$. For a group element $\mathbf{o}$, we denote this reduced representation by $[\mathbf{o}]_{\mathbf{g},\mathbf{h},\mathbf{y}}$ and its components as $g_{[\mathbf{o}]_{\overrightarrow{I}}}, h_{[\mathbf{o}]_{\overrightarrow{I}}}, y_{[\mathbf{o}]_{\overrightarrow{I}}}$, respectively. To make this notation more compact, we will define $\overrightarrow{I} := (\mathbf{g}, \mathbf{h}, \mathbf{y})$. If M wins, we denote the special message/signature pair in its winning output as $(m, (\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu))$. The algebraic coefficients of this tuple define the following values:

$$\delta' := (g_{[\beta_2]_{\overrightarrow{I}}} + x \cdot y_{[\beta_2]_{\overrightarrow{I}}})/(x \cdot y_{[\zeta_2]_{\overrightarrow{I}}} + g_{[\zeta_2]_{\overrightarrow{I}}})$$
$$\delta'' := \frac{h_{[\beta_1]_{\overrightarrow{I}}}}{h_{[\zeta_1]_{\overrightarrow{I}}}}, \delta''' := \frac{h_{[\eta]_{\overrightarrow{I}}}}{h_{[\zeta]_{\overrightarrow{I}}}}.$$

We further define the following non-exclusive boolean variables that describe when which of the above values is actually well-defined:

- $C_0 := (\omega \neq y_{[\alpha]_{\overrightarrow{I}}})$
- $C_1 := (\omega = y_{[\alpha]_{\overrightarrow{I}}}) \wedge (x \cdot y_{[\zeta_2]_{\overrightarrow{I}}} + g_{[\zeta_2]_{\overrightarrow{I}}} \neq 0)$
- $C_2 := (\omega = y_{[\alpha]_{\overrightarrow{I}}}) \wedge (h_{[\zeta_1]_{\overrightarrow{I}}} \neq 0)$
- $C_3 := (\omega = y_{[\alpha]_{\overrightarrow{I}}}) \wedge (h_{[\zeta]_{\overrightarrow{I}}} \neq 0)$[7]

Claim 3. $\bigvee_i C_i = 1$.

Proof. Since $\mathbf{G_2}^{\mathsf{M}} = 1 \Rightarrow \zeta_2 \neq \epsilon$, it follows $y_{[\zeta_2]_{\overrightarrow{I}}}, g_{[\zeta_2]_{\overrightarrow{I}}}$, and $h_{[\zeta_2]_{\overrightarrow{I}}}$ can not simultaneously be 0 when $G_2^{\mathsf{M}} = 1$. Therefore, either $x \cdot y_{[\zeta_2]_{\overrightarrow{I}}} + g_{[\zeta_2]_{\overrightarrow{I}}} \neq 0$ or $h_{[\zeta_2]_{\overrightarrow{I}}} \neq 0$. Moreover, since $[\zeta_2]_{\mathbf{g},\mathbf{h},\mathbf{y}} = [\zeta]_{\mathbf{g},\mathbf{h},\mathbf{y}} - [\zeta_1]_{\mathbf{g},\mathbf{h},\mathbf{y}}$, either $h_{[\zeta_1]_{\overrightarrow{I}}} \neq 0$ or $h_{[\zeta]_{\overrightarrow{I}}} \neq 0$, whenever $h_{[\zeta_2]_{\overrightarrow{I}}} \neq 0$. Therefore, $(h_{[\zeta_1]_{\overrightarrow{I}}} \neq 0 \vee h_{[\zeta]_{\overrightarrow{I}}} \neq 0 \vee x \cdot y_{[\zeta_2]_{\overrightarrow{I}}} + g_{[\zeta_2]_{\overrightarrow{I}}} \neq 0) = 1$. Since $(\omega \neq y_{[\alpha]_{\overrightarrow{I}}} \vee \omega = y_{[\alpha]_{\overrightarrow{I}}}) = 1$, it follows that $\bigvee_i C_i = 1$. □

We now define the event E_2 as follows:

$$\begin{aligned} E_2 := & \neg(C_0 = 1) \\ & \wedge (\neg(C_1 = 1)) \vee (C_1 = 1 \wedge (\delta' = \delta))) \\ & \wedge (\neg(C_2 = 1) \vee (C_2 = 1 \wedge (\delta'' = \delta))) \\ & \wedge (\neg(C_3 = 1) \vee (C_3 = 1 \wedge (\delta''' = \delta))). \end{aligned}$$

Claim 4. $\Pr[E_2] \leq \frac{3q_h}{q}$.

Proof. Let $\varepsilon = H_3(\zeta, \zeta_1, \mathbf{g}^\rho\mathbf{y}^\omega, \mathbf{g}^{\sigma_1}\zeta_1^\delta, \mathbf{h}^{\sigma_2}\zeta_2^\delta, \mathbf{z}^\mu\zeta^\delta, m)$. Recall that in $\mathbf{G_2}$, M always makes this query if $((\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu), m)$ denotes the tuple with the special signature in its output. Thus, by Claim 3, we consider the probability that M makes a query $H_3(\zeta, \zeta_1, \mathbf{g}^\rho\mathbf{y}^\omega, \mathbf{g}^{\sigma_1}\zeta_1^\delta, \mathbf{h}^{\sigma_2}\zeta_2^\delta, \mathbf{z}^\mu\zeta^\delta, m)$ such that $\bigvee_i C_i = 1$ and E_2 happens. For any particular such query, since $C_0 = 0$, then $\omega = y_{[\alpha]_{\overrightarrow{I}}}$ is fixed at the time this query is made. Hence, $\delta := \varepsilon - \omega$ is a uniformly distributed value from M's perspective at the time it submits this query. In particular, δ is uniform and independent of the algebraic coefficients that M submits together with its query to H_3 and thus of the values $\delta', \delta'', \delta'''$, which are completely determined via the algebraic coefficients (when they are defined). On the other hand, since E_2 happens and $C_0 = 0$, $C_i = 1$ for some some $i \in [3]$. Hence $\delta' = \delta \vee \delta'' = \delta \vee \delta''' = \delta$ holds. Since $\Pr[\delta' = \delta \vee \delta'' = \delta \vee \delta''' = \delta] \leq \frac{3}{q}$ (where equality only holds if the left hand side is defined) and there are at most q_h

[7] We implicitly include the condition that $\mathbf{G_2}^{\mathsf{M}} = 1$ in each of the conditions; otherwise, they would not be well-defined.

queries to H_3, we can upper bound M's success probability of causing event E_2 as $\frac{3q_h}{q}$ (using a union bound argument). $\square$

By the claim, $\mathsf{Adv}^{\mathbf{G_2}}_{\mathsf{M}} \geq \mathsf{Adv}^{\mathbf{G_1}}_{\mathsf{M}} - \frac{3q_h}{q}$.

In the following, we explain how the reduction can simulate game $\mathbf{G_2}$ to the adversary M and win the discrete logarithm game.

Simulation of H_1, H_2, H_3. We begin by describing how $S_0, \ldots, S_3$ simulate the random oracles H_1, H_2, H_3. These simulations are common to all S_i and are performed in the straightforward way using lazy sampling. We assume that the oracles keep respective lists L_i for bookkeeping, where L_i stores input/output pairs. More specifically.

- H_1 and H_2: on each fresh input ξ, H_i samples $v \xleftarrow{\$} \mathbb{Z}_q$ and returns $\mathbf{g}^v$. It stores $(\xi, \mathbf{g}^v, v)$ in L_i.
- H_3 : on each fresh input $(\xi, \cdot)$, H_3 samples $h \xleftarrow{\$} \mathbb{Z}_q$ and returns h. It stores $(\xi, \overrightarrow{\text{rep}}, h)$ in L_i.
- On repeated inputs H_i returns whatever it returned the first time that ξ was queried.

Scheduling of Signing Sessions. We assume that each S_i internally schedules sessions with the oracles $\mathbf{Sign}_1$ and $\mathbf{Sign}_2$ as required by the **OMUF** experiment. This can be easily implemented by using a fresh session identifier for each new session.

Extracting Equations from Forgery. Suppose that M wins game $\mathbf{G_2}$, i.e., $\mathbf{G}^{\mathsf{M}}_{\mathbf{2}} = 1$. Recall that in this case, M produces a one-more forgery of at least $\ell+1$ valid signatures, after having completed at most ℓ sessions with oracle $\mathbf{Sign}_2$. In addition, we have required that one of the returned tuples (m, sig) be special, i.e., that $\zeta^{\mathrm{dlog}_{\mathbf{z}}\, \mathbf{z}_{1,j}} \neq \zeta_1$ for all $\mathbf{z}_{1,j}$ (where again $\mathbf{z}_{1,j}$ corresponds to the value of $\mathbf{z}_j$ derived during the j-th interaction with oracle $\mathbf{Sign}_1$).

From the verification equation of the special signature (m, sig), one obtains the equations $\alpha = \mathbf{g}^{\rho} \cdot \mathbf{y}^{\omega}$, $\beta_1 = \zeta_1^{\delta} \cdot \mathbf{g}^{\sigma_1}$, $\beta_2 = \zeta_2^{\delta} \cdot \mathbf{h}^{\sigma_2}$, $\eta = \mathbf{z}^{\mu} \cdot \zeta^{\delta}$. Denoting $w_0 := \mathrm{dlog}\, \mathbf{z}$, $w := \mathrm{dlog}\, \mathbf{h}$, we obtain the reduced equations

$$g_{[\alpha]_{\overrightarrow{T}}} + x \cdot y_{[\alpha]_{\overrightarrow{T}}} + w \cdot h_{[\alpha]_{\overrightarrow{T}}} = \rho + x \cdot \omega \tag{6}$$

$$g_{[\beta_1]_{\overrightarrow{T}}} + x \cdot y_{[\beta_1]_{\overrightarrow{T}}} + w \cdot h_{[\beta_1]_{\overrightarrow{T}}} = (g_{[\zeta_1]_{\overrightarrow{T}}} + w \cdot h_{[\zeta_1]_{\overrightarrow{T}}} + x \cdot y_{[\zeta_1]_{\overrightarrow{T}}}) \cdot \delta + \sigma_1 \tag{7}$$

$$g_{[\beta_2]_{\overrightarrow{T}}} + x \cdot y_{[\beta_2]_{\overrightarrow{T}}} + w \cdot h_{[\beta_2]_{\overrightarrow{T}}} = (g_{[\zeta_2]_{\overrightarrow{T}}} + w \cdot h_{[\zeta_2]_{\overrightarrow{T}}} + x \cdot y_{[\zeta_2]_{\overrightarrow{T}}}) \cdot \delta + \sigma_2 \cdot w \tag{8}$$

$$g_{[\eta]_{\overrightarrow{T}}} + w \cdot h_{[\eta]_{\overrightarrow{T}}} + x \cdot y_{[\eta]_{\overrightarrow{T}}} = w_0 \cdot \mu + (g_{[\zeta]_{\overrightarrow{T}}} + w \cdot h_{[\zeta]_{\overrightarrow{T}}} + x \cdot y_{[\zeta]_{\overrightarrow{T}}}) \cdot \delta. \tag{9}$$

We continue by describing simulators $S_0, ..., S_3$ that cover the cases defined by $C_0, \ldots, C_3$. As we will see, the values c, r, d, s_1, s_2 inside a signature issued as part of a signing query are all known to S_i. Together with the above observations, it is easy for each simulator to convert a query to H_3 into reduced representation. Moreover, the winning tuple in M's output can be identified through knowledge of the logarithms of $\mathbf{z}$ and all $\mathbf{z}_{1,i}$ efficiently.

Case $C_0 = 1$. We describe simulator S_0, which simulates $\mathbf{G_2}$ using w. On input a discrete logarithm instance $\mathbf{U} := \mathbf{g}^x$, it behaves as follows:

Setup: S_0 samples $w, w_0 \xleftarrow{\$} \mathbb{Z}_q$. If $w_0 = 0$, it aborts. Otherwise it computes the public key pk as $\mathsf{pk} := (\mathbf{g}, \mathbf{h} := \mathbf{g}^w, \mathbf{y} := \mathbf{U}, \mathbf{z} := \mathbf{g}^{w_0})$, which implicitly sets $\mathsf{sk} := x$. It adds the tuple $((\mathsf{pp}, \mathbf{g}, \mathbf{h}, \mathbf{y}), \mathbf{z}, w_0)$ to the list L_1.

Online Phase. S_0 runs M on input pp, pk and simulates the oracles $\mathbf{Sign}_1, \mathbf{Sign}_2$ as described below. In addition, it simulates the oracles H_1, H_2, H_3 as outlined above.

Queries to $\mathbf{Sign}_1$. When M queries $\mathbf{Sign}_1()$ to open session sid, S_0 samples $\mathrm{rnd}_{\mathrm{sid}} \xleftarrow{\$} \{0,1\}^{\lambda}$ and sets $\mathbf{z}_{1,\mathrm{sid}} := \mathbf{g}^{w_{1,\mathrm{sid}}} = H_2(\mathrm{rnd}_{\mathrm{sid}})$, which places the tuple $(\mathrm{rnd}_{\mathrm{sid}}, \mathbf{z}_{1,\mathrm{sid}}, w_{1,\mathrm{sid}})$ into L_2. It then sets $\mathbf{z}_{2,\mathrm{sid}} := \mathbf{z}/\mathbf{z}_{1,\mathrm{sid}}$, $w_{2,\mathrm{sid}} := \frac{w_0 - w_{1,\mathrm{sid}}}{w}$, $c_{\mathrm{sid}}, r_{\mathrm{sid}}, u_{1,\mathrm{sid}}, u_{2,\mathrm{sid}} \xleftarrow{\$} \mathbb{Z}_q$, $\mathbf{a}_{\mathrm{sid}} := \mathbf{y}^{c_{\mathrm{sid}}} \cdot \mathbf{g}^{r_{\mathrm{sid}}}$, $\mathbf{b_1}_{,\mathrm{sid}} := \mathbf{g}^{u_{1,\mathrm{sid}}}$, $\mathbf{b_2}_{,\mathrm{sid}} := \mathbf{h}^{u_{2,\mathrm{sid}}}$ and returns $\mathbf{a}_{\mathrm{sid}}, \mathbf{b_1}_{,\mathrm{sid}}, \mathbf{b_2}_{,\mathrm{sid}}$.

Queries to $\mathbf{Sign}_2$. When M queries $\mathbf{Sign}_2(\mathrm{sid}, e_{\mathrm{sid}})$, S_0 sets $d_{\mathrm{sid}} := e_{\mathrm{sid}} - c_{\mathrm{sid}}$, $s_{1,\mathrm{sid}} := u_{1,\mathrm{sid}} - d_{\mathrm{sid}} \cdot w_{1,\mathrm{sid}}$, $s_{2,\mathrm{sid}} := u_{2,\mathrm{sid}} - d_{\mathrm{sid}} \cdot w_{2,\mathrm{sid}}$ and returns $c_{\mathrm{sid}}, d_{\mathrm{sid}}, r_{\mathrm{sid}}, s_{1,\mathrm{sid}}, s_{2,\mathrm{sid}}$.

It is straightforward to verify that the above simulation of $\mathbf{G_2}$ is perfect.

Solving the DLP instance. When M returns $\ell + 1$ message signature pairs, S_0 identifies the special signature using the exponents stored in L_2. It retrieves the corresponding hash query to H_3 from L_3 together with representations of $\alpha, \beta_1, \beta_2, \eta$. S_0 uses Eq. (6) and the fact that $C_0 = 1 \Leftrightarrow \omega \neq y_{[\alpha]_{\overrightarrow{T}}}$, to (efficiently) compute and output the value x as $x = (\rho - g_{[\alpha]_{\overrightarrow{T}}} - w \cdot h_{[\alpha]_{\overrightarrow{T}}})/(y_{[\alpha]_{\overrightarrow{T}}} - \omega)$. (In case $C_0 = 0$, or there is no hash query corresponding to the special signature, it aborts.)

If $C_0 = 1$, then S_0's simulation of $\mathbf{G_2}$ is perfect.

Case $C_0 = C_2 = C_3 = 0 \ \wedge \ C_1 = 1$. We describe simulator S_1, which simulates $\mathbf{G_2}$ using x. On input a discrete logarithm instance $\mathbf{U} := \mathbf{g}^w$, it behaves as follows:

Setup. S_1 samples $x, w_0 \xleftarrow{\$} \mathbb{Z}_q$. It sets $\mathsf{pk} := (\mathbf{g}, \mathbf{h} := \mathbf{U}, \mathbf{y} := \mathbf{g}^x, \mathbf{z} := \mathbf{g}^{w_0})$, $\mathsf{sk} := x$ and adds $((\mathsf{pp}, \mathbf{g}, \mathbf{h}, \mathbf{y}), \mathbf{z}, w_0)$ to L_1.

Online Phase. S_1 runs M on input pp, pk and simulates the oracles $\mathbf{Sign}_1, \mathbf{Sign}_2$ as described below. In addition, it simulates the oracles H_1, H_2, H_3 as outlined above.

Queries to $\mathbf{Sign}_1$. When M queries $\mathbf{Sign}_1$ to open session sid, S_1 samples $\mathrm{rnd}_{\mathrm{sid}} \xleftarrow{\$} \{0,1\}^{\lambda}$ and sets $\mathbf{z}_{1,\mathrm{sid}} := \mathbf{g}^{w_{1,\mathrm{sid}}} = H_2(\mathrm{rnd}_{\mathrm{sid}})$ (hence $w_{1,\mathrm{sid}}$ is known to S_1 from programming H_2). It then samples $u_{\mathrm{sid}}, d_{\mathrm{sid}}, s_{1,\mathrm{sid}}, s_{2,\mathrm{sid}} \xleftarrow{\$} \mathbb{Z}_q$ and sets $\mathbf{a}_{\mathrm{sid}} := \mathbf{g}^{u_{\mathrm{sid}}}$, $\mathbf{b}_{1,\mathrm{sid}} := \mathbf{g}^{s_{1,\mathrm{sid}}} \cdot \mathbf{z}_{1,\mathrm{sid}}^{d_{\mathrm{sid}}}$, $\mathbf{b}_{2,\mathrm{sid}} := \mathbf{h}^{s_{2,\mathrm{sid}}} \cdot \mathbf{z}_{2,\mathrm{sid}}^{d_{\mathrm{sid}}}$ and returns $\mathbf{a}_{\mathrm{sid}}, \mathbf{b}_{1,\mathrm{sid}}, \mathbf{b}_{2,\mathrm{sid}}$.

Queries to $\mathbf{Sign}_2$. When M queries $\mathbf{Sign}_2$ on input $(\mathrm{sid}, e_{\mathrm{sid}})$, S_1 sets $c_{\mathrm{sid}} := e_{\mathrm{sid}} - d_{\mathrm{sid}}$, $r_{\mathrm{sid}} := u_{\mathrm{sid}} - c_{\mathrm{sid}} \cdot x$ and returns $c_{\mathrm{sid}}, d_{\mathrm{sid}}, r_{\mathrm{sid}}, s_{1,\mathrm{sid}}, s_{2,\mathrm{sid}}$

Solving the DLP instance. When M returns $\ell + 1$ message signature pairs, S_1 identifies the special signature using

the exponents stored in L_2. It retrieves the corresponding hash query to H_3 from L_3 together with representations of $\alpha, \beta_1, \beta_2, \eta$. If there is no hash query to H_3 corresponding to the special signature, it aborts. Otherwise S_1 uses Eq. (8) and the fact that $C_1 = 1 \Leftrightarrow (\omega = y_{[\alpha]_{\overrightarrow{1}}}) \wedge (x \cdot y_{[\zeta_2]_{\overrightarrow{1}}} + g_{[\zeta_2]_{\overrightarrow{1}}} \neq 0)$, to (efficiently) compute and output the value w as follows. (In case $C_1 = 0$, it aborts.) S_1 first computes δ' as $\delta' \coloneqq (g_{[\beta_2]_{\overrightarrow{1}}} + x \cdot y_{[\beta_2]_{\overrightarrow{1}}})/(x \cdot y_{[\zeta_2]_{\overrightarrow{1}}} + g_{[\zeta_2]_{\overrightarrow{1}}})$, which gives the equality

$$\begin{aligned} &\delta' \cdot (g_{[\zeta_2]_{\overrightarrow{1}}} + x \cdot y_{[\zeta_2]_{\overrightarrow{1}}}) + w \cdot h_{[\beta_2]_{\overrightarrow{1}}} \\ &= g_{[\beta_2]_{\overrightarrow{1}}} + x \cdot y_{[\beta_2]_{\overrightarrow{1}}} + w \cdot h_{[\beta_2]_{\overrightarrow{1}}}. \end{aligned} \tag{10}$$

Eqs. (10) and (8) yield

$$\begin{aligned} &\delta' \cdot (g_{[\zeta_2]_{\overrightarrow{1}}} + x \cdot y_{[\zeta_2]_{\overrightarrow{1}}}) + w \cdot h_{[\beta_2]_{\overrightarrow{1}}} \\ &= g_{[\beta_2]_{\overrightarrow{1}}} + x \cdot y_{[\beta_2]_{\overrightarrow{1}}} + w \cdot h_{[\beta_2]_{\overrightarrow{1}}} \\ &= \delta \cdot (g_{[\zeta_2]_{\overrightarrow{1}}} + x \cdot y_{[\zeta_2]_{\overrightarrow{1}}} + w \cdot h_{[\zeta_2]_{\overrightarrow{1}}}) + \sigma_2 \cdot w. \end{aligned}$$

If $h_{[\beta_2]_{\overrightarrow{1}}} - \delta \cdot h_{[\zeta_2]_{\overrightarrow{1}}} - \sigma_2 \neq 0$, S_1 outputs $w = ((\delta - \delta') \cdot (g_{[\zeta_2]_{\overrightarrow{1}}} + x \cdot h_{[\zeta_2]_{\overrightarrow{1}}}))/(h_{[\beta_2]_{\overrightarrow{1}}} - \delta \cdot h_{[\zeta_2]_{\overrightarrow{1}}} - \sigma_2)$. (In case $C_1 = 0$ or $C_i = 1$ for some $i \in \{0, 2, 3\}$, or there is no hash query for the special signature, it aborts).

We prove the following claim.

Claim 5. $h_{[\beta_2]_{\overrightarrow{1}}} - \delta \cdot h_{[\zeta_2]_{\overrightarrow{1}}} - \sigma_2 \neq 0$.

Proof. Since $C_0 = C_2 = C_3 = 0 \ \wedge \ C_1 = 1$ and event E_2 does not happen (since otherwise $\mathbf{G_2^M} = 0$), we know that $\delta \neq \delta'$. Hence, it suffices to show that if $\delta \neq \delta'$, then $h_{[\beta_2]_{\overrightarrow{1}}} - \delta \cdot h_{[\zeta_2]_{\overrightarrow{1}}} - \sigma_2 \neq 0$. Due to Eq. (8) we get

$$\begin{aligned} &\delta' \cdot (g_{[\zeta_2]_{\overrightarrow{1}}} + x \cdot y_{[\zeta_2]_{\overrightarrow{1}}}) + w \cdot h_{[\beta_2]_{\overrightarrow{1}}} \\ &= \delta \cdot (g_{[\zeta_2]_{\overrightarrow{1}}} + x \cdot y_{[\zeta_2]_{\overrightarrow{1}}} + w \cdot h_{[\zeta_2]_{\overrightarrow{1}}}) + \sigma_2 \cdot w \\ &= \delta \cdot (g_{[\zeta_2]_{\overrightarrow{1}}} + x \cdot y_{[\zeta_2]_{\overrightarrow{1}}}) + w \cdot h_{[\beta_2]_{\overrightarrow{1}}}, \end{aligned}$$

which yields $(\delta' - \delta) \cdot (g_{[\zeta_2]_{\overrightarrow{1}}} + x \cdot y_{[\zeta_2]_{\overrightarrow{1}}}) = 0$. Since $C_1 = 1$, we have $g_{[\zeta_2]_{\overrightarrow{1}}} + x \cdot y_{[\zeta_2]_{\overrightarrow{1}}} \neq 0$, which contradicts the assumption that $\delta' \neq \delta$. □

It is easily verified that whenever $C_1 = 1$, S_1's simulation of $\mathbf{G_2}$ is perfect.

Case $C_0 = C_1 = C_3 = 0 \ \wedge \ C_2 = 1$**.** We describe simulator S_2. S_2 uses the same simulation strategy as S_1. When M returns $\ell+1$ message signature pairs, S_2 identifies the special signature using the exponents stored in L_2. It retrieves the corresponding hash query to H_3 from L_3 together with representations of $\alpha, \beta_1, \beta_2, \eta$. S_2 uses Eq. (7) and the fact that $C_2 = 1 \Leftrightarrow (\omega = y_{[\alpha]_{\overrightarrow{1}}}) \wedge (h_{[\zeta_1]_{\overrightarrow{1}}} \neq 0)$, to compute and output the discrete logarithm w of the instance $\mathbf{U}$ as follows. (In case $C_2 = 0$, or there was no hash query corresponding to the special signature to H_3, it aborts). S_2 first computes $\delta'' \coloneqq \frac{h_{[\beta_1]_{\overrightarrow{1}}}}{h_{[\zeta_1]_{\overrightarrow{1}}}}$ which leads to the equality

$$\begin{aligned} &\delta'' \cdot w \cdot h_{[\zeta_1]_{\overrightarrow{1}}} + g_{[\beta_1]_{\overrightarrow{1}}} + x \cdot y_{[\beta_1]_{\overrightarrow{1}}} \\ &= g_{[\beta_1]_{\overrightarrow{1}}} + x \cdot y_{[\beta_1]_{\overrightarrow{1}}} + w \cdot h_{[\beta_1]_{\overrightarrow{1}}}. \end{aligned} \tag{11}$$

Eqs. (11) and (7) yield

$$\begin{aligned} &\delta'' \cdot w \cdot h_{[\zeta_1]_{\overrightarrow{1}}} + g_{[\beta_1]_{\overrightarrow{1}}} + x \cdot y_{[\beta_1]_{\overrightarrow{1}}} \\ &= g_{[\beta_1]_{\overrightarrow{1}}} + x \cdot y_{[\beta_1]_{\overrightarrow{1}}} + w \cdot h_{[\beta_1]_{\overrightarrow{1}}} \\ &= (g_{[\zeta_1]_{\overrightarrow{1}}} + w \cdot h_{[\zeta_1]_{\overrightarrow{1}}} + x \cdot y_{[\zeta_1]_{\overrightarrow{1}}}) \cdot \delta + \sigma_1. \end{aligned}$$

By the same argument as in the previous case, $\delta \neq \delta''$, and S_2 can compute and output w as $w = (\delta \cdot (g_{[\zeta_1]_{\overrightarrow{1}}} + x \cdot y_{[\zeta_1]_{\overrightarrow{1}}}) + \sigma_1 - g_{[\beta_1]_{\overrightarrow{1}}} + x \cdot y_{[\beta_1]_{\overrightarrow{1}}})/((\delta - \delta'') \cdot h_{[\zeta_1]_{\overrightarrow{1}}})$, as $C_2 = 1$ implies that $h_{[\zeta_1]_{\overrightarrow{1}}} \neq 0$. Moreover, S_2's simulation of $\mathbf{G_2}$ is perfect if $C_2 = 1$ holds.

Case $C_0 = C_1 = C_2 = 0 \ \wedge \ C_3 = 1$**.** We describe simulator S_3. S_3 uses the same simulation strategy as simulators S_1 and S_2. In this case, S_3 uses Eq. (9) and the fact that $C_3 = 1 \Leftrightarrow (\omega = y_{[\alpha]_{\overrightarrow{1}}}) \wedge (h_{[\zeta]_{\overrightarrow{1}}} \neq 0)$, to compute and output the discrete logarithm w of the instance $\mathbf{U}$ as we described below. (In case $C_3 = 0$ there was no query corresponding to the special signature to H_3, it aborts). S_3 computes $\delta''' \coloneqq h_{[\eta]_{\overrightarrow{1}}}/h_{[\zeta]_{\overrightarrow{1}}}$, leading to

$$\begin{aligned} &\delta''' \cdot w \cdot h_{[\zeta]_{\overrightarrow{1}}} + g_{[\eta]_{\overrightarrow{1}}} + x \cdot y_{[\eta]_{\overrightarrow{1}}} \\ &= g_{[\eta]_{\overrightarrow{1}}} + x \cdot y_{[\eta]_{\overrightarrow{1}}} + w \cdot h_{[\eta]_{\overrightarrow{1}}}. \end{aligned} \tag{12}$$

Eqs. (12) and (9) imply that

$$\begin{aligned} &\delta''' \cdot w \cdot h_{[\zeta]_{\overrightarrow{1}}} + g_{[\eta]_{\overrightarrow{1}}} + x \cdot y_{[\eta]_{\overrightarrow{1}}} \\ &= g_{[\eta]_{\overrightarrow{1}}} + x \cdot y_{[\eta]_{\overrightarrow{1}}} + w \cdot h_{[\eta]_{\overrightarrow{1}}} \\ &= w_0 \cdot \mu + (g_{[\zeta]_{\overrightarrow{1}}} + w \cdot h_{[\zeta]_{\overrightarrow{1}}} + x \cdot y_{[\zeta]_{\overrightarrow{1}}}) \cdot \delta. \end{aligned}$$

As in the previous cases, $\delta \neq \delta'''$, so S_3 can output w by computing $w = (\delta \cdot (g_{[\zeta]_{\overrightarrow{1}}} + x \cdot y_{[\zeta]_{\overrightarrow{1}}}) + \mu \cdot w_0 - (g_{[\eta]_{\overrightarrow{1}}} + x \cdot y_{[\eta]_{\overrightarrow{1}}}))/((\delta''' - \delta) \cdot h_{[\zeta]_{\overrightarrow{1}}})$, since $h_{[\zeta]_{\overrightarrow{1}}} \neq 0$ due to $C_3 = 1$. Moreover, S_3's simulation of $\mathbf{G_2}$ is perfect if $C_3 = 1$ holds.

Since all simulators provide a perfect simulation (in their respective cases) and cover all cases that can happen whenever $\mathbf{G_2^M} = 1$, B can run the correct simulator to extract the discrete logarithm with advantage $\mathsf{Adv}_{\mathsf{B}}^{\mathbf{DLP}} \geq \mathsf{Adv}_{\mathsf{M}}^{\mathbf{G_2}}/4$. Moreover, we have $\mathsf{Adv}_{\mathsf{M}}^{\mathbf{G_2}} \geq \mathsf{Adv}_{\mathsf{M}}^{\mathbf{G_1}} - \frac{3q_h}{q} \geq \mathsf{Adv}_{\mathsf{M}}^{\mathbf{G_0}} - \frac{3q_h}{q} - \frac{\ell+1}{q}$. Hence, $t_{\mathsf{B}} \approx t_{\mathsf{M}}$ and

$$\begin{aligned} \mathsf{Adv}_{\mathsf{B}}^{\mathbf{DLP}} &\geq \frac{1}{4}\mathsf{Adv}_{\mathsf{M}}^{\mathbf{RB\text{-}OMUF}} - \frac{3q_h}{4q} - \frac{\ell+1}{4q} \\ &\geq \frac{1}{4}\mathsf{Adv}_{\mathsf{M}}^{\mathbf{RB\text{-}OMUF}} - \frac{q_h}{q}. \end{aligned}$$

□

B. The Main Theorem

In the following, we show that Abe's blind signature scheme has full one-more-unforgeability. We make use of the restrictive blinding lemma to identify the forged signature.

Theorem 2. *Let* M *be an algebraic algorithm that runs in time* t_{M}*, makes at most* ℓ *queries to oracle* $\mathbf{Sign}_2$ *in* $\mathbf{OMUF}_{\mathsf{BSA}}$ *and at most (total)* q_h *queries to* H_1, H_2, H_3*. Then, in the random oracle model, there exists an algorithm* B *such that*

$$\mathsf{Adv}_{\mathsf{B}}^{\mathbf{DLP}} \geq \frac{1}{10}\mathsf{Adv}_{\mathsf{M}}^{\mathbf{OMUF}_{\mathsf{BSA}}} - \frac{7q_h}{10q}$$

and B *runs in time*

$$t_{\mathsf{B}} \approx t_{\mathsf{M}}.$$

Proof. Let M be as in the lemma statement. As before, we assume w.l.o.g. that M makes exactly ℓ queries to $\mathbf{Sign}_2$ and returns a list of $\ell+1$ tuples. The proof goes by a series of games, which we describe below.

Game $\mathbf{G_0}$. This is the original $\mathbf{OMUF}_{\mathsf{BSA}}$ game.

Game $\mathbf{G_1}$. This game is identical to $\mathbf{G_0}$, except that it aborts if M returns a pair $(m, \mathsf{sig}) = (m, (\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu)) \in L$ s.t. for all j corresponding to a session that was previously closed via an interaction with $\mathbf{Sign}_2$ (i.e., a session from which M learned a signature), we have: $\zeta^{\mathrm{dlog}_{\mathbf{z}}\, \mathbf{z}_{1,j}} \neq \zeta_1 \wedge \zeta_2 \neq \epsilon$. We denote this abort event by E_1. Due to Lemma 1, an adversary that outputs such a signature can be used to solve the discrete logarithm problem. We state this in the following claim:

Claim 6. There exists an algorithm B_{rb} such that $\Pr[E_1] \leq \mathsf{Adv}^{\mathbf{DLP}}_{\mathsf{B}_{\mathrm{rb}}}$ and B_{rb} runs in time $\mathsf{B}_{\mathrm{rb}} \approx t_{\mathsf{M}}$.

Proof. Here, B_{rb} is the reduction algorithm from Lemma 1. The claim follows from the lemma. □

Game $\mathbf{G_2}$. This game is identical to $\mathbf{G_1}$ except that it aborts if M did not make *both of the hash queries* to H_3 that correspond to the two special signatures. We denote this event as E_2.

Claim 7. $\Pr[E_2] \leq \frac{\ell+1}{q}$

Proof. Analogous to Claim 2. □

Thus, if M wins $\mathbf{G_2}$, there must exist two signatures $\mathsf{sig}, \mathsf{sig}'$ s.t. $\mathrm{dlog}_{\zeta}\, \zeta_1 = \mathrm{dlog}_{\zeta'}\, \zeta_1'$, i.e., the two pairs ζ, ζ_1 and ζ', ζ_1' were derived from the same pair $\mathbf{z}, \mathbf{z}_1$. Analogous to Lemma 1, we call these two signatures the *special signatures*.

Game $\mathbf{G_3}$. This game is the same as $\mathbf{G_2}$ but analogously to $\mathbf{G_2}$ in Lemma 1, it has an additional abort event E_3 depending on the representation of the group elements submitted to the hash oracle H_3. In the following we explain some preliminaries for this condition.

Simplifying notations and assumptions. As in the proof of Lemma 1, the reduction obtains representations of all group elements output by the adversary because the adversary is algebraic. Through the lists L_1, L_2 and its own computation of the elements output by $\mathbf{Sign}_1$, it knows a representation of all input group elements to the basis $\mathbf{g}, \mathbf{y}, \mathbf{z}$ and can thus compute a reduced representation to basis $\mathbf{g}, \mathbf{y}, \mathbf{z}$ for all output group elements of the adversary. We denote its components for a group element $\mathbf{o}$ as $g_{[\mathbf{o}]_{\vec{K}}}, y_{[\mathbf{o}]_{\vec{K}}}, z_{[\mathbf{o}]_{\vec{K}}}$. To make this notation compact we define $\vec{K} := (\mathbf{g}, \mathbf{y}, \mathbf{z})$.

We further define the following values:

$$\delta' := \frac{g_{[\eta]_{\vec{K}}} + y_{[\eta]_{\vec{K}}} \cdot x}{g_{[\zeta]_{\vec{K}}} + y_{[\zeta]_{\vec{K}}} \cdot x}, \delta'' := \frac{z_{[\beta_1]_{\vec{K}}}}{z_{[\zeta_1]_{\vec{K}}}}, \delta''' := \frac{z_{[\beta_2]_{\vec{K}}}}{z_{[\zeta_2]_{\vec{K}}}},$$

and describe the following cases in which at least one of the above variables is defined:

Equations to extract the discrete logarithm. When M queries H_3 it also submits a representation of all group elements contained in the query. Given that M wins $\mathbf{G_3}$, the two special signatures exist and for both of them, these queries to H_3 were indeed made (otherwise the abort events E_1 or E_2 happen and M does not win $\mathbf{G_3}$). We denote the group elements submitted to the random oracle for special signature $i \in \{1,2\}$ in these queries by $\alpha_i, \beta_{1,i}, \beta_{2,i}, \eta_i$ and consider below their algebraic coefficients. More concretely, we consider the following four (non-exclusive) cases for each of two special signatures:

- $C_{0,i} := (\omega_i \neq y_{[\alpha_i]_{\vec{K}}})$
- $C_{1,i} := (\omega_i = y_{[\alpha_i]_{\vec{K}}} \wedge g_{[\zeta_i]_{\vec{K}}} + x \cdot y_{[\zeta_i]_{\vec{K}}} \neq 0)$
- $C_{2,i} := (\omega_i = y_{[\alpha_i]_{\vec{K}}} \wedge z_{[\zeta_{1,i}]_{\vec{K}}} \neq 0)$
- $C_{3,i} := (\omega_i = y_{[\alpha_i]_{\vec{K}}} \wedge z_{[\zeta_{2,i}]_{\vec{K}}} \neq 0)$

Claim 8. $\bigvee_j C_{j,i} = 1$ for $i \in \{1,2\}$

Proof. For simplicity, we omit the index i. Since for a valid signature, none of the components ζ, ζ_1, ζ_2 can be the neutral element ϵ, it must be the case that $g_{[\zeta]_{\vec{K}}} + y_{[\zeta]_{\vec{K}}} \cdot x \neq 0$ or $z_{[\zeta]_{\vec{K}}} \neq 0$. In other words, if $g_{[\zeta]_{\vec{K}}} + x \cdot y_{[\zeta]_{\vec{K}}} = 0$, then $z_{[\zeta]_{\vec{K}}} \neq 0$. As $[\zeta_2]_{\vec{K}} = [\zeta]_{\vec{K}} - [\zeta_1]_K$, this means that $z_{[\zeta_1]_{\vec{K}}} \neq 0$ or $z_{[\zeta_2]_{\vec{K}}} \neq 0$. Hence if $C_0 = 0$, then $C_1 = 1 \vee C_2 = 1 \vee C_3 = 1$ must be true. □

Let h_1, h_2 be the random oracle outputs of the two special signatures and let $\alpha_i, \beta_{1,i}, \beta_{2,i}, \eta_i$ denote the group elements submitted to H_3 to obtain h_1, h_2 (recall that in $\mathbf{G_3}$, these queries are always made if $\mathbf{G_3}^{\mathsf{M}} = 1$). Moreover, let $\delta_i', \delta_i'', \delta_i''', y_{[\alpha_i]_{\vec{K}}}$ denote the values computed from the corresponding algebraic representations. We define the abort events $E_3 := E_{3,1} \wedge E_{3,2}$ where

$$\begin{aligned} E_{3,i} :=& \neg(C_{0,i} = 1) \\ & \wedge (\neg(C_{1,i} = 1) \vee (C_{1,i} = 1 \wedge (\delta_i' = \delta_i))) \\ & \wedge (\neg(C_{2,i} = 1) \vee (C_{2,i} = 1 \wedge (\delta_i'' = \delta_i))) \\ & \wedge (\neg(C_{3,i} = 1) \vee (C_{3,i} = 1 \wedge (\delta_i''' = \delta_i))). \end{aligned}$$

Claim 9. $\Pr[E_3] = \frac{6 \cdot q_h}{q}$.

Proof. Analogous to Claim 4. □

In the following, we describe how a reduction algorithm B' can simulate $\mathbf{G_3}$ to M and use the obtained representations of the two special signatures to solve the discrete logarithm problem.

Because M is allowed to use any group element in its input as part of the algebraic representation, it is again necessary for the reduction to program the random oracles H_1, H_2 so that it can compute the reduced representation for all input elements of the adversary. The reduction will embed its discrete logarithm challenge in either $\mathbf{y}$ or $\mathbf{z}$ and then try to compute the discrete logarithm from the algebraic representation submitted when originally querying the hash oracle and from the corresponding signature. We describe eight simulators $S_{0,i}, S_{1,i}, S_{2,i}, S_{3,i}$ $(i \in \{1,2\})$ that simulate the OMUF-game to an adversary. $S_{0,i}$ answers the signing queries by simulating using the discrete logarithms of $\mathbf{z}, \mathbf{z}_{1,i}, \mathbf{z}_{2,i}$

and tries to compute the discrete logarithm of $\mathbf{y}$ from the representation of the group element α submitted as part of the queries to H_3. $S_{1,i}, S_{2,i}$, and $S_{3,i}$ answer signing queries using the discrete logarithm x of $\mathbf{y}$. Depending on the algebraic representation of ζ, ζ_1, ζ_2, at least one of them is successful in extracting the discrete logarithm w_0 of $\mathbf{z}$.

Implementation of the hash oracles H_1, H_2, H_3. All simulators implement the hash oracles H_1, H_2, H_3 in almost the same way, namely by lazy sampling similar to the implementations in the proof of Lemma 1. However, the responses to H_2 are sampled with respect to $\mathbf{z}$ and $S_{1,i}, S_{2,i}$ and $S_{3,i}$ hardcode its discrete logarithm challenge at the appropriate place to be returned by H_1 as the public key. We describe the implementation in the following.

- H_1: on a fresh input ξ, H_1 samples $v \xleftarrow{\$} \mathbb{Z}_q$ and returns $\mathbf{g}^v$. It stores $(\xi, \mathbf{g}^v, v)$ in L_1.
  - For $S_{1,i}, S_{2,i}, S_{3,i}$: before making the first call to H_1, $S_{1,i}, S_{2,i}, S_{3,i}$ store $((\mathsf{pp}, \mathbf{g}, \mathbf{h}, \mathbf{y}), \mathbf{U}, \bot)$ in L_1 to be returned as $\mathbf{z}$, where $\mathbf{U}$ is the discrete logarithm challenge.
- H_2: on a fresh input ξ, H_2 samples $v \xleftarrow{\$} \mathbb{Z}_q$ and returns $\mathbf{z}^v$. It stores $(\xi, \mathbf{z}^v, v)$ in L_2.
- H_3: on a fresh input $(\xi, \overrightarrow{\text{rep}})$, H_3 samples $h \xleftarrow{\$} \mathbb{Z}_q$ and returns h. It stores $(\xi, \overrightarrow{\text{rep}}, h)$ in L_3.

Scheduling of signing sessions. As before, all simulators use unique session identifiers sid to match corresponding oracle calls to $\mathbf{Sign}_1$ and $\mathbf{Sign}_2$.

Identification of the forgery. If M wins $\mathbf{G_3}$, the special signatures can easily be identified through the list L_2 used by oracle H_2, since the simulators know the discrete logarithms of the values $\mathbf{z}_{1,i}$ to base $\mathbf{z}$. Thus, the simulators can identify j s.t. $\zeta_{1,i} = \zeta_i^{\mathrm{dlog}_{\mathbf{z}}\, \mathbf{z}_{1,j}}$ for $i \in \{1, 2\}$. In the following, we denote $\mathrm{dlog}\, \mathbf{z}$ as w_0, $\mathrm{dlog}\, \mathbf{y}$ as x, and $\mathrm{dlog}\, \mathbf{h}$ as w. For simplicity and legibility, we restrict ourselves to describing the cases for the first of the two special signatures and assume that the event $E_{3,1}$ has not happend. Hence, in the following, we have omitted the index $i = 1$. For every simulator below, we remark that we can describe an analogous simulator that attempts extraction on the second special signature, i.e., for $i = 2$ and where $E_{3,2}$ has not happened.

Case $C_0 = 1$. We describe the simulator S_0. This simulator embeds the discrete logarithm challenge in $\mathbf{y}$ and uses knowledge of w_0, w_1, w_2 to generate its signatures.

Setup For $\mathbf{DLP}$ challenge $\mathbf{U}$, S_0 samples $w \xleftarrow{\$} \mathbb{Z}_q$ and sets $\mathbf{h} := \mathbf{g}^w$, and $\mathbf{y} := \mathbf{U}$. It initializes all random oracle query lists as $L_1, L_2, L_3 := \emptyset$ and the session id counter with $\mathrm{sid} = 0$. It then calls the oracle $H_1(\mathsf{pp}, \mathbf{g}, \mathbf{h}, \mathbf{y})$ to obtain $\mathbf{z}$ and looks up $w_0 = \mathrm{dlog}\, \mathbf{z}$ in L_1.

Online phase The public key $\mathsf{pk} := (\mathbf{g}, \mathbf{h}, \mathbf{y}, \mathbf{z})$ is given to the adversary. The simulator responds to oracle calls to H_1, H_2, H_3 as described before and to signing queries as follows.

Queries to $\mathbf{Sign}_1$ S_0 increases the session id counter and samples $c_{\mathrm{sid}}, r_{\mathrm{sid}}, u_{1,\mathrm{sid}}, u_{2,\mathrm{sid}} \xleftarrow{\$} \mathbb{Z}_q$. It sets $\mathbf{a}_{\mathrm{sid}} := \mathbf{y}^{c_{\mathrm{sid}}} \cdot \mathbf{g}^{r_{\mathrm{sid}}}$, $\mathbf{b}_{1,\mathrm{sid}} := \mathbf{g}^{u_{1,\mathrm{sid}}}$ and $\mathbf{b}_{2,\mathrm{sid}} := \mathbf{h}^{u_{2,\mathrm{sid}}}$. Furthermore it samples $\mathrm{rnd}_{\mathrm{sid}} \xleftarrow{\$} \{0,1\}^\lambda$ and calls the oracle H_2 on it to define $\mathbf{z}_1$. It outputs $\mathrm{sid}, \mathbf{a}_{\mathrm{sid}}, \mathbf{b}_{2,\mathrm{sid}}, \mathbf{b}_{2,\mathrm{sid}}, \mathrm{rnd}_{\mathrm{sid}}$ as the response.

Queries to $\mathbf{Sign}_2$ On input $e_{\mathrm{sid}}, \mathrm{sid}$, S_0 retrieves the variables for the session marked by sid and computes $d_{\mathrm{sid}} := e_{\mathrm{sid}} - c_{\mathrm{sid}}$. It retrieves $w_{1,\mathrm{sid}} := (\mathrm{dlog}_{\mathbf{z}}\, \mathbf{z}_1) \cdot w_0$ through L_2 as well as $w_{2,\mathrm{sid}} := \frac{w_0 - w_{1,\mathrm{sid}}}{w}$. It computes $s_{1,\mathrm{sid}} := u_{1,\mathrm{sid}} - d \cdot w_{1,\mathrm{sid}}$ and $s_{2,\mathrm{sid}} := u_{1,\mathrm{sid}} - d \cdot w_{2,\mathrm{sid}}$ and returns $c_{\mathrm{sid}}, d_{\mathrm{sid}}, r_{\mathrm{sid}}, s_{1,\mathrm{sid}}, s_{2,\mathrm{sid}}$ to the adversary.

It is straightforward to see that the above simulation of $\mathbf{G_3}$ is perfect if C_0 holds.

Solving the DLP instance. If $C_0 = 0$, S_0 aborts. Otherwise, it looks up the two special signatures and their reduced algebraic representations. Since $C_0 = 1$, S_0 can compute the discrete logarithm x of $\mathbf{y}$ as

$$x = \frac{g_{[\alpha]_{\overrightarrow{K}}} + w_0 \cdot z_{[\alpha]_{\overrightarrow{K}}} - \rho}{\omega - y_{[\alpha]_{\overrightarrow{K}}}}.$$

The simulators S_1, S_2, S_3 embed the $\mathbf{DLP}$ challenge as $\mathbf{z}$. They can thus answer the signing queries using the secret key x as in the original scheme. For a tight proof in the AGM, our techniques differ from Abe's proof technique (which embedded the challenge in one of the $\mathbf{z}_1$). We describe the simulators in the following.

Case $C_0 = C_2 = C_3 = 0 \ \wedge \ C_1 = 1$. We describe the behaviour of simulator S_1.

Setup. For $\mathbf{DLP}$ challenge $\mathbf{U}$, the simulator samples $x, w \xleftarrow{\$} \mathbb{Z}_q$ and sets $\mathbf{y} := \mathbf{g}^x$ and $\mathbf{h} := \mathbf{g}^w$. It initializes all random oracle lists $L_1, L_2, L_3 := \emptyset$ and then programs the random oracle H_1 to return $\mathbf{z} = \mathbf{U}$ on input $(\mathsf{pp}, \mathbf{g}, \mathbf{h}, \mathbf{y})$. It initializes the session counter $\mathrm{sid} := 0$.

Online phase The simulator starts the adversary on input $\mathsf{pk} = (\mathbf{g}, \mathbf{h}, \mathbf{y}, \mathbf{z})$. It responds to hash queries to H_1, H_2, H_3 as above and to signing queries as follows:

Queries to $\mathbf{Sign}_1$. S_1 increases the session counter and samples $u_{\mathrm{sid}}, d_{\mathrm{sid}}, s_{1,\mathrm{sid}}, s_{2,\mathrm{sid}} \xleftarrow{\$} \mathbb{Z}_q$ and $\mathrm{rnd}_{\mathrm{sid}} \xleftarrow{\$} \{0,1\}^\lambda$ to query $\mathbf{z}_{1,\mathrm{sid}} := H_2(\mathrm{rnd}_{\mathrm{sid}})$. It defines $\mathbf{a}_{\mathrm{sid}} := \mathbf{g}^{u_{\mathrm{sid}}}$, $\mathbf{b}_{1,\mathrm{sid}} := \mathbf{z}_{1,\mathrm{sid}}^{d_{\mathrm{sid}}} \cdot \mathbf{g}^{s_{1,\mathrm{sid}}}$ and $\mathbf{b}_{2,\mathrm{sid}} := \mathbf{z}_{2,\mathrm{sid}}^{d_{\mathrm{sid}}} \cdot \mathbf{h}^{s_{2,\mathrm{sid}}}$. It returns $\mathrm{sid}, \mathbf{a}_{\mathrm{sid}}, \mathbf{b}_{1,\mathrm{sid}}, \mathbf{b}_{2,\mathrm{sid}}, \mathrm{rnd}_{\mathrm{sid}}$ to the adversary.

Queries to $\mathbf{Sign}_2$. On input $\mathrm{sid}, e_{\mathrm{sid}}$, S_1 retrieves the exponents used the first signing query of session sid and computes $c_{\mathrm{sid}} := e_{\mathrm{sid}} - d_{\mathrm{sid}}$. It then further computes $r_{\mathrm{sid}} := u_{\mathrm{sid}} - c_{\mathrm{sid}} \cdot x$ and returns $c_{\mathrm{sid}}, d_{\mathrm{sid}}, r_{\mathrm{sid}}, s_{1,\mathrm{sid}}, s_{2,\mathrm{sid}}$.

Solving DLP. If $C_1 = 0$, S_1 aborts. Otherwise it computes the discrete logarithm of $\mathbf{U}$ as follows: If $g_{[\zeta]_{\overrightarrow{K}}} + x \cdot y_{[\zeta]_{\overrightarrow{K}}} \neq 0$ we use

$$\delta' = \frac{g_{[\eta]_{\overrightarrow{K}}} + x \cdot y_{[\eta]_{\overrightarrow{K}}}}{g_{[\zeta]_{\overrightarrow{K}}} + x \cdot y_{[\zeta]_{\overrightarrow{K}}}}$$

Claim 10. $z_{[\eta]_{\overrightarrow{K}}} - \delta \cdot z_{[\zeta]_{\overrightarrow{K}}} - \mu \neq 0$.

Proof. The proof works analogous to the proof of Claim 5. □

By the claim, S_1 can then solve for w_0 as

$$w_0 \coloneqq \frac{(\delta - \delta') \cdot (g_{[\zeta]_{\vec{K}}} + x \cdot y_{[\zeta]_{\vec{K}}})}{z_{[\eta]_{\vec{K}}} - \delta \cdot z_{[\zeta]_{\vec{K}}} - \mu}.$$

Case $C_0 = C_1 = C_3 = 0 \wedge C_2 = 1$. We briefly sketch the simulator S_2. This simulator uses the same implementations for $\mathbf{Setup}, \mathbf{Sign}_1, \mathbf{Sign}_2$ as S_1. It aborts if $C_2 = 0$. If it does not abort, $z_{[\zeta_1]_{\vec{K}}} \neq 0$. In this case, we know that

$$\delta'' \coloneqq \frac{z_{[\beta_1]_{\vec{K}}}}{z_{[\zeta_1]_{\vec{K}}}}$$

is well-defined and moreover $\delta \neq \delta''$. Hence, S_2 can compute the discrete logarithm w_0 of $\mathbf{z}$ as

$$w_0 \coloneqq \frac{g_{[\beta_1]_{\vec{K}}} + x \cdot y_{[\beta_1]_{\vec{K}}} - \delta \cdot (g_{[\zeta_1]_{\vec{K}}} + x \cdot y_{[\zeta_1]_{\vec{K}}}) - \sigma_1}{(\delta - \delta'') \cdot z_{[\zeta_1]_{\vec{K}}}}$$

Case $C_0 = C_1 = C_2 = 0 \wedge C_3 = 1$. Again, we only sketch the behaviour describe S_3, which also uses the same implementations for $\mathbf{Setup}, \mathbf{Sign}_1, \mathbf{Sign}_2$ as S_1. It computes the discrete logarithm from β_2 as described above and aborts if $C_3 = 0$. We know that in this case,

$$\delta''' = \frac{z_{[\beta_2]_{\vec{K}}}}{z_{[\zeta_2]_{\vec{K}}}}$$

is well-defined and $\delta \neq \delta'''$. Hence, S_3 can compute the discrete logarithm w_0 of $\mathbf{z}$ as

$$w_0 = \frac{g_{[\beta_2]_{\vec{K}}} + x \cdot y_{[\beta_2]_{\vec{K}}} - \delta \cdot (g_{[\zeta_2]_{\vec{K}}} + x \cdot y_{[\zeta_2]_{\vec{K}}}) - \sigma_2 \cdot w}{(\delta - \delta''') \cdot z_{[\zeta_2]_{\vec{K}}}}$$

Calculation of success probability. All simulators above give a perfect simulation of game $\mathbf{G_3}$ in their respective cases. The overall reduction algorithm B flips a coin. On heads, it samples one of the 8 simulators at random and runs it. On tails, it instead internally runs $\mathsf{Adv}^{\mathbf{DLP}}_{\mathsf{B}_{\mathsf{rb}}}$. The reduction runs in time

$$t_{\mathsf{B}} \approx t_{\mathsf{M}}$$

and has advantage

$$\begin{aligned}\mathsf{Adv}^{\mathbf{DLP}}_{\mathsf{B}} &\geq \frac{\mathsf{Adv}^{\mathbf{G_3}}_{\mathsf{M}}}{8} \geq \frac{\mathsf{Adv}^{\mathbf{G_2}}_{\mathsf{M}} - \frac{6 \cdot q_h}{q}}{8} \\ &\geq \frac{\mathsf{Adv}^{\mathbf{G_1}}_{\mathsf{M}} - \frac{\ell+1}{q} - \frac{6 \cdot q_h}{q}}{8} \\ &\geq \frac{\mathsf{Adv}^{\mathbf{G_0}}_{\mathsf{M}} - \mathsf{Adv}^{\mathbf{DLP}}_{\mathsf{B}_{\mathsf{rb}}} - \frac{7 \cdot q_h}{q}}{8}.\end{aligned}$$

Moreover, note that $\mathsf{Adv}^{\mathbf{DLP}}_{\mathsf{B}} \geq \frac{1}{2}\mathsf{Adv}^{\mathbf{DLP}}_{\mathsf{B}_{\mathsf{rb}}}$. Hence,

$$\mathsf{Adv}^{\mathbf{DLP}}_{\mathsf{B}} \geq \frac{\mathsf{Adv}^{\mathbf{G_0}}_{\mathsf{M}} - 2\mathsf{Adv}^{\mathbf{DLP}}_{\mathsf{B}} - \frac{7q_h}{q}}{8},$$

and the total advantage of B is

$$\mathsf{Adv}^{\mathbf{DLP}}_{\mathsf{B}} \geq \frac{1}{10}\mathsf{Adv}^{\mathbf{OMUF}_{\mathsf{BSA}}}_{\mathsf{M}} - \frac{7q_h}{10q}.$$

□

V. Benchmarks

We include benchmarks of the sequential blind Schnorr and parallel Abe schemes as motivation for the practical use of our new security bounds of pairing-free blind signature schemes.

As a starting point for a model, we imagine a server such as the cryptocurrency tumbler described in [HBG16]. Such a server would need to be able to carry out a blind signature protocol with perhaps hundreds of clients in a single epoch in order to provide suitable anonymity for all participants. Further, the server would not know in advance the rate at which clients arrive, nor the time it takes for a given client to respond to the server's initiation.

We find that both the Schnorr and Abe schemes are practical at moderate workloads, and that Abe scales extremely well using parallel/interleaved execution, sufficiently so to potentially serve as the algorithm of choice for a high-profile tumbler for a moderately sized cryptocurrency.

1) Experimental Setup: The model we benchmark is a simplified version of the cryptocurrency tumbler. We run blind Schnorr and Abe signing protocols as a local HTTP server, with 1 thread for blind Schnorr, and a varying number of threads for Abe (1, 4, and 16). The Abe server, regardless of number of threads, is allowed to interleave up to 100 sessions, i.e., it can run a hundred (concurrent) sequences of the form $\mathsf{Sign}_1, \mathsf{Sign}_2$. The Schnorr server is not allowed to interleave sessions. If a client connects to the Schnorr server during another client's session, it will receive an HTTP 409 (Conflict) from the server and pause for 75ms before retrying.

For each server configuration, we spawn 100 clients which perform the full blind signature protocol with the server. In order to simulate different (relatively high) workloads, we model each batch of 100 clients as a different Poisson point process with a set expected interarrival time. That is, in a batch with expected interarrival time EIAT milliseconds (set to 1, 10, 50, 90, and 130), each client independently draws from the exponential distribution with mean EIAT and waits that many milliseconds before trying to connect. To execute the protocol, each client performs an HTTP GET to the server's Sign_1 endpoint, computes User_1, then GETs the Sign_2 endpoint using as input the results from User_1. Finally, the client unblinds the signature in User_2 and checks that it is valid. Our benchmarks for each configuration measure the time it takes for all 100 clients to complete their session with the server.

All signature schemes are instantiated using the Ristretto255 group [Val+]—a prime-order group of order about 2^{252}, which is built from Curve25519 [Ber06] arithmetic primitives. Since group elements and scalars are small (32 bytes each), we ignore bandwidth considerations in these benchmarks. We also choose to model server-client network latency for all runs as a normal distribution centered at 30ms with $\sigma = 5$. This is

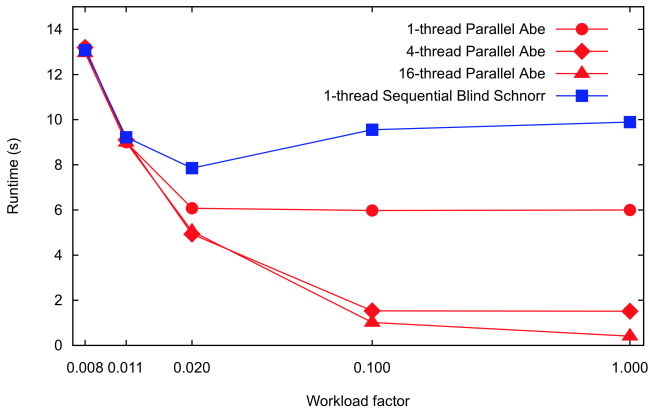

Figure 1. Benchmarking a web service running sequential blind Schnorr and parallel Abe signature schemes. "Work factor" is defined as $1/\mathsf{EIAT}$, where EIAT is the clients' expected interarrival time in milliseconds.

roughly the latency the authors measured between their home laptop and a university server roughly $1/3$ of the way across the United States. Benchmarks were compiled with rustc 1.48-nightly and executed on an AWS EC2 c5a.8xlarge virtual machine, with 32 vCPUs (16 physical cores). Code is available at https://github.com/rozbb/blindsig_benches.

2) Results: We see that for lighter workloads of $\mathsf{EIAT} =$ 130ms and 90ms, all configurations perform optimally: almost all the time spent in each benchmark comes from the client wait times (which are cumulatively 13s and 9s, respectively). This makes Schnorr a good choice in practice for small workloads, since it is far simpler to implement and has smaller signature sizes. As workload increases, though, interleaving/parallelism becomes the dominating factor in performance, and the difference between Schnorr and Abe becomes more drastic. At $\mathsf{EIAT} =$10ms, the 16-thread Abe scheme performs within 4% of the optimum, and at $\mathsf{EIAT} =$ 1ms the 16-thread Abe scheme starts to break down, with a runtime 408% worse than optimal.

Concretely, these numbers are more than sufficient for a tumbler service. For comparison, the size of the ZCash shielded transaction anonymity pool can be conservatively upper-bounded by 10k users. At the current target block rate of 2.5min, this means that a single 4-thread Abe server with anonymity set size 100 can tumble coins for *every shielded transaction user* in the same amount of time it takes to add a block to the chain.

References

[Abe01] Masayuki Abe. 'A Secure Three-Move Blind Signature Scheme for Polynomially Many Signatures'. In: *Advances in Cryptology – EUROCRYPT 2001*. Ed. by Birgit Pfitzmann. Vol. 2045. Lecture Notes in Computer Science. Innsbruck, Austria: Springer, Heidelberg, Germany, 2001, pp. 136–151. DOI: 10.1007/3-540-44987-6_9.

[AHK20] Thomas Agrikola, Dennis Hofheinz and Julia Kastner. 'On Instantiating the Algebraic Group Model from Falsifiable Assumptions'. In: *Advances in Cryptology – EUROCRYPT 2020, Part II*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. Lecture Notes in Computer Science. Zagreb, Croatia: Springer, Heidelberg, Germany, 2020, pp. 96–126. DOI: 10.1007/978-3-030-45724-2_4.

[Bel+03] Mihir Bellare et al. 'The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme'. In: *Journal of Cryptology* 16.3 (June 2003), pp. 185–215. DOI: 10.1007/s00145-002-0120-1.

[Ben+20] Fabrice Benhamouda et al. *On the (in)security of ROS*. Cryptology ePrint Archive, Report 2020/945. https://eprint.iacr.org/2020/945. 2020.

[Ber06] Daniel J. Bernstein. 'Curve25519: New Diffie-Hellman Speed Records'. In: *PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography*. Ed. by Moti Yung et al. Vol. 3958. Lecture Notes in Computer Science. New York, NY, USA: Springer, Heidelberg, Germany, 2006, pp. 207–228. DOI: 10.1007/11745853_14.

[BL13a] Foteini Baldimtsi and Anna Lysyanskaya. 'Anonymous credentials light'. In: *ACM CCS 2013: 20th Conference on Computer and Communications Security*. Ed. by Ahmad-Reza Sadeghi, Virgil D. Gligor and Moti Yung. Berlin, Germany: ACM Press, 2013, pp. 1087–1098. DOI: 10.1145/2508859.2516687.

[BL13b] Foteini Baldimtsi and Anna Lysyanskaya. 'On the Security of One-Witness Blind Signature Schemes'. In: *Advances in Cryptology – ASIACRYPT 2013, Part II*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8270. Lecture Notes in Computer Science. Bengalore, India: Springer, Heidelberg, Germany, 2013, pp. 82–99. DOI: 10.1007/978-3-642-42045-0_5.

[Bol03] Alexandra Boldyreva. 'Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme'. In: *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*. Ed. by Yvo Desmedt. Vol. 2567. Lecture Notes in Computer Science. Miami, FL, USA: Springer, Heidelberg, Germany, 2003, pp. 31–46. DOI: 10.1007/3-540-36288-6_3.

[BR04] Mihir Bellare and Phillip Rogaway. *Code-Based Game-Playing Proofs and the Security of Triple Encryption*. Cryptology ePrint Archive, Report 2004/331. http://eprint.iacr.org/2004/331. 2004.

[BR93] Mihir Bellare and Phillip Rogaway. 'Random Oracles are Practical: A Paradigm for Designing Efficient Protocols'. In: *ACM CCS 93: 1st Confer-*

ence on Computer and Communications Security. Ed. by Dorothy E. Denning et al. Fairfax, Virginia, USA: ACM Press, 1993, pp. 62–73. DOI: 10.1145/168588.168596.

[Cha82] David Chaum. 'Blind Signatures for Untraceable Payments'. In: *Advances in Cryptology – CRYPTO'82*. Ed. by David Chaum, Ronald L. Rivest and Alan T. Sherman. Santa Barbara, CA, USA: Plenum Press, New York, USA, 1982, pp. 199–203.

[FHS15] Georg Fuchsbauer, Christian Hanser and Daniel Slamanig. 'Practical Round-Optimal Blind Signatures in the Standard Model'. In: *Advances in Cryptology – CRYPTO 2015, Part II*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9216. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2015, pp. 233–253. DOI: 10.1007/978-3-662-48000-7_12.

[Fis06] Marc Fischlin. 'Round-Optimal Composable Blind Signatures in the Common Reference String Model'. In: *Advances in Cryptology – CRYPTO 2006*. Ed. by Cynthia Dwork. Vol. 4117. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2006, pp. 60–77. DOI: 10.1007/11818175_4.

[FKL18] Georg Fuchsbauer, Eike Kiltz and Julian Loss. 'The Algebraic Group Model and its Applications'. In: *Advances in Cryptology – CRYPTO 2018, Part II*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10992. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2018, pp. 33–62. DOI: 10.1007/978-3-319-96881-0_2.

[FPS20] Georg Fuchsbauer, Antoine Plouviez and Yannick Seurin. 'Blind Schnorr Signatures and Signed ElGamal Encryption in the Algebraic Group Model'. In: *Advances in Cryptology – EUROCRYPT 2020, Part II*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. Lecture Notes in Computer Science. Zagreb, Croatia: Springer, Heidelberg, Germany, 2020, pp. 63–95. DOI: 10.1007/978-3-030-45724-2_3.

[Gar+11] Sanjam Garg et al. 'Round Optimal Blind Signatures'. In: *Advances in Cryptology – CRYPTO 2011*. Ed. by Phillip Rogaway. Vol. 6841. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2011, pp. 630–648. DOI: 10.1007/978-3-642-22792-9_36.

[GG14] Sanjam Garg and Divya Gupta. 'Efficient Round Optimal Blind Signatures'. In: *Advances in Cryptology – EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. Lecture Notes in Computer Science. Copenhagen, Denmark: Springer, Heidelberg, Germany, 2014, pp. 477–495. DOI: 10.1007/978-3-642-55220-5_27.

[Hau+20] Eduard Hauck et al. 'Lattice-Based Blind Signatures, Revisited'. In: *Advances in Cryptology – CRYPTO 2020*. Ed. by Daniele Micciancio and Thomas Ristenpart. Cham: Springer International Publishing, 2020, pp. 500–529. ISBN: 978-3-030-56880-1.

[HBG16] Ethan Heilman, Foteini Baldimtsi and Sharon Goldberg. 'Blindly Signed Contracts: Anonymous On-Blockchain and Off-Blockchain Bitcoin Transactions'. In: *FC 2016 Workshops*. Ed. by Jeremy Clark et al. Vol. 9604. Lecture Notes in Computer Science. Christ Church, Barbados: Springer, Heidelberg, Germany, 2016, pp. 43–60. DOI: 10.1007/978-3-662-53357-4_4.

[HKL19] Eduard Hauck, Eike Kiltz and Julian Loss. 'A Modular Treatment of Blind Signatures from Identification Schemes'. In: *Advances in Cryptology – EUROCRYPT 2019, Part III*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11478. Lecture Notes in Computer Science. Darmstadt, Germany: Springer, Heidelberg, Germany, 2019, pp. 345–375. DOI: 10.1007/978-3-030-17659-4_12.

[OA03] Miyako Ohkubo and Masayuki Abe. *Security of Some Three-move Blind Signature Schemes Reconsidered*. The 2003 Symposium on Cryptography and Information Security. Hamamatsu,Japan, 2003.

[Oka06] Tatsuaki Okamoto. 'Efficient Blind and Partially Blind Signatures Without Random Oracles'. In: *TCC 2006: 3rd Theory of Cryptography Conference*. Ed. by Shai Halevi and Tal Rabin. Vol. 3876. Lecture Notes in Computer Science. New York, NY, USA: Springer, Heidelberg, Germany, 2006, pp. 80–99. DOI: 10.1007/11681878_5.

[Poi98] David Pointcheval. 'Strengthened Security for Blind Signatures'. In: *Advances in Cryptology – EUROCRYPT'98*. Ed. by Kaisa Nyberg. Vol. 1403. Lecture Notes in Computer Science. Espoo, Finland: Springer, Heidelberg, Germany, 1998, pp. 391–405. DOI: 10.1007/BFb0054141.

[PS00] David Pointcheval and Jacques Stern. 'Security Arguments for Digital Signatures and Blind Signatures'. In: *Journal of Cryptology* 13.3 (June 2000), pp. 361–396. DOI: 10.1007/s001450010003.

[PS96] David Pointcheval and Jacques Stern. 'Provably Secure Blind Signature Schemes'. In: *Advances in Cryptology – ASIACRYPT'96*. Ed. by Kwangjo Kim and Tsutomu Matsumoto. Vol. 1163. Lecture Notes in Computer Science. Kyongju, Korea: Springer, Heidelberg, Germany, 1996, pp. 252–265. DOI: 10.1007/BFb0034852.

[PS97] David Pointcheval and Jacques Stern. 'New Blind Signatures Equivalent to Factorization (Extended

Abstract)'. In: *ACM CCS 97: 4th Conference on Computer and Communications Security*. Ed. by Richard Graveman et al. Zurich, Switzerland: ACM Press, 1997, pp. 92–99. DOI: 10.1145/266420.266440.

[Sch01] Claus-Peter Schnorr. 'Security of Blind Discrete Log Signatures against Interactive Attacks'. In: *ICICS 01: 3rd International Conference on Information and Communication Security*. Ed. by Sihan Qing, Tatsuaki Okamoto and Jianying Zhou. Vol. 2229. Lecture Notes in Computer Science. Xian, China: Springer, Heidelberg, Germany, 2001, pp. 1–12.

[Sho04] Victor Shoup. *Sequences of games: a tool for taming complexity in security proofs*. Cryptology ePrint Archive, Report 2004/332. http://eprint.iacr.org/2004/332. 2004.

[Val+] H. de Valence et al. *The ristretto255 Group*. IETF Informational Memo. https://ietf.org/id/draft-irtf-cfrg-ristretto255-00.html.

[Wag02] David Wagner. 'A Generalized Birthday Problem'. In: *Advances in Cryptology – CRYPTO 2002*. Ed. by Moti Yung. Vol. 2442. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2002, pp. 288–303. DOI: 10.1007/3-540-45708-9_19.

APPENDIX

A. Additional Security Definitions

Definition 7 (Blindness)**.** We define *blindness* of a three-move blind signature scheme BS against an adversary M via the following game:

Game $\mathbf{BLIND}_{\mathsf{BS}}$:

- **Setup.** $\mathbf{BLIND}_{\mathsf{BS}}$ generates a public and secret key pair via $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{BS.KeyGen}(\mathsf{pp})$ and samples $b \xleftarrow{\$} \{0,1\}$. It runs M on input $\mathsf{sk}, \mathsf{pk}, \mathsf{pp}$.
- **Online Phase.** When M outputs messages $\tilde{m}_0$ and $\tilde{m}_1$, $\mathbf{BLIND}_{\mathsf{BS}}$ assigns $m_0 := \tilde{m}_b$ and $m_1 := \tilde{m}_{1-b}$. M is given access to oracles $\mathbf{User}_1, \mathbf{User}_2$, which behaves as follows.
  - Oracle $\mathbf{User}_1$: On input a bit b' and a commitment C, if the session b' is not yet open, the game marks session b' as open and generates a state and challenge as $(\mathsf{st}_{b'}, e) \xleftarrow{\$} \mathsf{BS.User}_1(\mathsf{pk}, m_{b'}, C)$. It returns e to the adversary. Otherwise, it returns $\perp$.
  - Oracle $\mathbf{User}_2$: On input a response R and a bit b', if the session b' is open, the game creates the signature $\mathsf{sig}_{b'}$ as $\mathsf{sig}_{b'} := \mathsf{BS.User}_2(\mathsf{pk}, \mathsf{st}_{b'}, R)$ to obtain a signature $\mathsf{sig}_{b'}$. It marks session b' as closed and outputs $\mathsf{sig}_{b'}$. If both sessions are closed and produced signatures, the oracle outputs the two signatures $\mathsf{sig}_0, \mathsf{sig}_1$ to the adversary.
- **Output Determination.** If both sessions are closed and produced signatures, the game outputs 1 iff the adversary outputs a bit b^* s.t. $b^* = b$. Otherwise, it outputs 0.

We define the advantage of M in game $\mathbf{BLIND}_{\mathsf{BS}}$ as $\mathsf{Adv}^{\mathsf{M}}_{\mathbf{BLIND},\mathsf{BS}} = \left|\Pr\left[\mathbf{BLIND}^{\mathsf{M}} = 1\right] - \frac{1}{2}\right|$. We say that BS is *perfectly blind* if for all adversaries M, advantage is $\mathsf{Adv}^{\mathsf{M}}_{\mathbf{BLIND},\mathsf{BS}} = 0$.