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

Julia Kastner[1*], Julian Loss[2**], and Jiayu Xu[3***]

[1] Department of Computer Science, ETH Zurich, Zurich, Switzerland
julia.kastner@inf.ethz.ch
[2] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
lossjulian@gmail.com
[3] Algorand, Boston, MA, USA, jiayux@uci.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 schemes either 1) rely on (very strong) number theoretic hardness assumptions and/or 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. In this work, we revisit the security of two *pairing-free* blind signature schemes in the Algebraic Group Model (AGM) + Random Oracle Model (ROM). Concretely,

1. We consider the security of Abe's scheme (EUROCRYPT '01), which is known to have a flawed proof in the plain ROM. We adapt the scheme to allow a partially blind variant and give a proof of the new scheme under the discrete logarithm assumption in the AGM+ROM, even for (polynomially many) *concurrent* signing sessions.
2. We then prove that the popular blind Schnorr scheme is secure under the one-more discrete logarithm assumption if the signatures are issued *sequentially*. While the work of Fuchsbauer et al. (EUROCRYPT '20) proves the security of the blind Schnorr scheme for *concurrent* signing sessions in the AGM+ROM, its underlying assumption, ROS, is proven false by Benhamouda et al. (EUROCRYPT '21) when more than *polylogarithmically many* signatures are issued. Given the recent progress, we present the first security analysis of the blind Schnorr scheme in the slightly weaker sequential setting. We also show that our security proof reduces from the weakest possible assumption, with respect to known reduction techniques.

1 Introduction

Blind signatures, first introduced by Chaum [17], are a fundamental cryptographic building block. They find use in many privacy sensitive applications

* Supported by ERC Project PREP-CRYPTO 724307
** Work done while at University of Maryland
*** Work done while at George Mason University

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 using pk. The protocol should ensure two properties [29]: (1) *One-More-Unforgeability:* if the protocol is run ℓ times, the user should not be able to create $\ell + 1$ or more valid signatures (2) *Blindness:* the signer cannot link the transcripts of protocol runs to the signatures that they created. In particular, it does not learn the messages that it signs. In a practical setting, signer and user might however want a more relaxed property to include some shared information, e.g. a date when the signature was issued or an expiration date. To this end, Abe and Fujisaki [2] introduced *Partial Blindness* which guarantees that signatures with the same shared information, the so-called *tag*, are unlinkable to protocol runs using this tag.

In spite of decades of study, the security guarantees of practical blind and partially blind signature schemes remain unsatisfactory. Practical constructions rely on strong number-theoretic hardness assumptions and/or computationally expensive pairing operations over bilinear groups [9, 13, 21, 24, 36]. 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 [3,15,27,28,38,40–42]. The reason for this is that the homomorphic structure of these schemes gives rise to the so-called ROS attack (Random inhomogenities in Overdetermined System of equations) when sufficiently many sessions of the scheme are executed concurrently (i.e., if session can be interleaved arbitrarily). Shortly after its discovery by Schnorr [45], Wagner [47] showed how to carry out the ROS attack in sub-exponential time against the Schnorr and Okamoto-Schnorr [35] blind signature schemes.[4] A recent result of Benhamouda et al. [12] improved the parameters of Wagner's attack, presenting the first polynomial-time attack (assuming that polylogarithmically many signing sessions can be opened concurrently).

1.1 Our Results

In this work, we revisit the security properties of two classic blind signature schemes which do not rely on pairings: Schnorr's blind signature scheme [16, 44] and Abe's blind signature scheme [1]. Neither of these schemes have meaningful security guarantees if the number of concurrent signing sessions is beyond polylogarithmic (in fact, Abe's blind signature scheme has no security proof at all in a non-generic model of computation). Given the popularity of these schemes, we believe that a reassessment of their security properties is long overdue. We give a summary of our results below.

[4] 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.

Abe's Scheme. In the first part of our work, we study the concurrent security properties of Abe's blind signature scheme. 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 Ohkubo [34] pointed out that the original proof contained a flaw and gave a security proof in the generic group model (GGM)+ROM instead. We generalize Abe's scheme to the partially blind setting and prove security of our new scheme in the more realistic AGM+ROM under the DL assumption. (We note that Abe's scheme can be obtained as a special case of our new scheme and thus our proof of security thus applies also to Abe's original scheme). As the work of Abe and Ohkubo is not publicly available, our proof is inspired by Abe's original proof and does not follow the blue print of a 'GGM-style proof.' Instead, we give a more general (and involved) proof that uses the AGM to avoid the rewinding step that causes the problem in Abe's proof. Apart from generalizing Abe's scheme to the partially blind setting, avoiding rewinding has the benefit that our reduction is tight, allowing for relatively practical parameter sizes. We stress that our reduction allows for the scheme to be proven secure with *concurrent signing sessions* and for *polynomially many signatures per tag.*

Schnorr's Scheme. In the second part of our work, we focus on the security of Schnorr's blind signature scheme. As we have already explained, the security of this scheme is completely broken in the concurrent setting for reasonable parameters. In spite of this, the Schnorr scheme continues to be one of the most popular blind signatures due to its simplicity and its efficiency. Hence, it is an important open question to settle what type of security this scheme actually *does achieve (if any).*

We show that the blind Schnorr signature scheme is secure in the algebraic group model (AGM) [22] + random oracle model (ROM) [10] if *signing sessions are sequential*, i.e., if the i-th session is always completed before the $(i+1)$-st session is opened.

In more detail, under the above model assumptions, the blind Schnorr signature scheme is secure against ℓ-sequential one-more-unforgeability (ℓ-SEQ-OMUF) under the ℓ-one-more discrete logarithm (ℓ-OMDL) assumption. This is true even when polynomially many signatures are issued for the same public key pk. 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.

Our result improves upon that of Fuchsbauer et al. [23], which proves that the scheme is secure under the OMDL+ROS assumption (when run concurrently). While the ROS problem is known to be information theoretically hard as long as the number of concurrent signing sessions is polylogarithmic, the recent work of Benhamouda et al. [12] shows a polynomial-time attack for super-polylogarithmically many concurrent signing sessions. Therefore, the blind Schnorr scheme is concurrently secure (in the AGM+ROM) if and only if the signer issues at most polylogarithmically many signatures.

Negative Result (Schnorr). As OMDL is a relatively strong assumption (in fact, [8] showed it is strictly stronger than q-discrete logarithm for known reduction approaches), a natural question is whether it is *actually necessary* for proving Schnorr's scheme secure. We answer this question by showing that our reduction for blind Schnorr signatures in the AGM+ROM is optimal in the sense that it is not possible to reduce ℓ-SEQ-OMUF from $(\ell - 1)$-OMDL (or OMDL with any lower dimension).

We use the meta-reduction technique [18] to rule out reductions in a very strong sense: we show that any algebraic reduction that reduces ℓ-SEQ-OMUF from $(\ell - 1)$-OMDL in the AGM+ROM, can be turned into an efficient solver against $(\ell - 1)$-OMDL. Our result complements that of Baldimtsi and Lysyanskaya [7], which also rules out a certain class of reductions for the blind Schnorr scheme. Concretely, they show that reductions that program the random oracle in a certain predictable way, can be turned into an efficient solver against the underlying hardness assumption. While their approach restricts the type of random oracle programming that the reduction may do, ours allows for arbitrary programming, but restricts the reduction to be algebraic. On the other hand, our (algebraic) reductions may themselves work in the AGM, which further strengthens our result.

1.2 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 [6] proves that an enhanced version of Abe's scheme is secure under DL. Pointcheval and Katz et al. [31, 39] give a transformations that apply (among others) to the blind Schnorr and Okamoto-Schnorr scheme. The resulting schemes remain secure even in the concurrent setting, but require communication that grows linear in the number of signatures that have been issued. In terms of practical parameters, these schemes are also significantly less efficient than the schemes we consider here. Fuchsbauer et al. [23] 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. Nicolosi et al. [33] use a similar strategy to ours (i.e., by restricting concurrency) to prove security of a proactive two-party signature scheme. Interestingly, they encounter similar issues as we do in our work, if concurrent session are permitted. Drijvers et al. [19] show how a ROS based attack can be applied in the context of multi-signatures (and how it can be overcome at the cost of some efficiency). Finally, various constructions of blind signatures in the standard model exist (e.g., [20, 25]), but are usually not considered practical.

The Algebraic Group Model. [22] introduced the *algebraic group model (AGM)* as a formal model to analyze group based cryptosystems. Previous works had considered algebraic algorithms, for example [14, 37]. In the AGM, any 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 [5, 8, 23, 26, 32] and has served as a useful tool to prove the security of schemes that would be too difficult to analyze in the plain model. [43] have furthermore extended the AGM to decisional assumptions.

While the AGM is a weakening of the GGM, proofs in the AGM are inherently different from the GGM in the sense that they are reductions from one problem to another instead of showing information-theoretic hardness. 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. [4] shows that some 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.

Another benefit of AGM proofs (over GGM proofs) is that they offer more insight into how secure a scheme actually is when deployed in real-world applications, as we explain in the following. In the GGM, a proof consists of establishing bounds on the runtime/success probabilities of an adversary attacking a particular signature scheme. These bounds often look similar for different schemes from an asymptotic point of view. Because of this, they do not give much insight into what computational assumptions are needed for the scheme to remain secure when run in the real world. By comparison, AGM proofs are by means of reduction from a computational assumption and thus can be used to assess the real-world disparities between two schemes that 'look equally secure' in the GGM. As a concrete example, our work gives a security proof for Abe's scheme under the discrete logarithm assumption. By comparison, we show that proving Schnorr's scheme secure (even under sequential signing sessions) requires the much stronger OMDL assumption. Arguably, this makes Abe's scheme the more attractive choice (along with allowing for concurrent sessions) for real world systems. This insight could not have been gained from proving these schemes secure in the GGM.

Open Questions. Our work leaves open the question of what can be proven about both the Abe and Schnorr blind signature schemes in the random oracle model only. Interestingly, the already mentioned work of Baldimtsi and Lysyanskaya [7] rules 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 (i.e., without requiring the AGM).

1.3 Organization

We first recall some preliminaries in section 2. In section 3 we introduce our adaption of Abe's scheme to the partially blind setting. We provide a proof of partial blindness under DDH in section 3.1 as well as a proof of one-more-unforgeability in section 3.2. We then provide the proof of sequential security of blind Schnorr signatures in the AGM in section 4 and show that this result is optimal in the number of OMDL-queries in section 4.1.

Acknowledgements We would like to thank Chenzhi Zhu and Stefano Tessaro for pointing out a flaw in a previous version of Claim 5. We would further like to thank the anonymous reviewers for their helpful feedback.

2 Preliminaries

2.1 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 vector $\overrightarrow{x}$ 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* [11, 46] 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 $\mathbf{G}^{\mathsf{A}} = 1$.

2.2 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 assume for simplicity that pp also includes the security parameter λ.) 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 as $[\mathbf{y}]_{\overrightarrow{\mathbf{x}}}$*. 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.

2.3 Hardness Assumptions

We introduce the two main hardness assumptions that we will use in the subsequent sections. As before, we will tacitly assume that some public parameters pp are known and describe a group $\mathbb{G}$ of prime order q with generator $\mathbf{g}$.

Definition 2 (Discrete Logarithm Problem (DLP)). *For an algorithm* A*, we define the game* **DLP** *as follows:*

Setup. *Sample* $x \xleftarrow{\$} \mathbb{Z}_q$ *and run* A *on input* $\mathbf{g}, \mathbf{U} \coloneqq \mathbf{g}^x$.

Output Determination. *When* A *outputs* x'*, return 1 if* $\mathbf{g}^{x'} = \mathbf{U}$*. Otherwise, return 0.*

We define the advantage of A *in* **DLP** *as*

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

Definition 3 (One-More-Discrete Logarithm Problem (OMDL)). *For a stateful algorithm* A *and a positive integer* ℓ*, we define the game* ℓ-**OMDL** *as follows:*

Setup. *Initialize* $C = \emptyset$*. Run* A *on input* $\mathbf{g}$.

Online Phase. A *is given access to the following oracles:*

- **Oracle chal** *takes no input and samples a group element* $\mathbf{y} \xleftarrow{\$} \mathbb{G}$*. It sets* $C \coloneqq C \cup \{\mathbf{y}\}$ *and returns* $\mathbf{y}$.
- **Oracle dlog** *takes as input a group element* $\mathbf{y}$*. It returns* $\mathrm{dlog}_{\mathbf{g}} \mathbf{y}$*. We assume that* **dlog** *can be queried at most* ℓ *many times.*

Output Determination. *When* A *outputs* $(\mathbf{y}_i, x_i)_{i=1}^{\ell+1}$*, return* 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, return* 0.

We define the advantage of A *in* ℓ-**OMDL** *as*

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

Definition 4 (Decsional Diffie-Hellman Problem (DDH)). *For an algorithm* A *we define the game* **DDH** *as follows:*

Setup. *Sample* $x, y, z \xleftarrow{\$} \mathbb{Z}_q$ *and* $b \xleftarrow{\$} \{0,1\}$*. Run* A *on input* $(\mathbf{g}, \mathbf{g}^x, \mathbf{g}^y, \mathbf{g}^{xy+bz})$

Output determination. *When* A *outputs* b'*, return* 1 *if* $b = b'$ *and* 0 *otherwise.*

We define the advantage of A *in* **DDH** *as*

$$\mathsf{Adv}^{\mathbf{DDH}}_{\mathsf{A}} \coloneqq \left|\Pr[\mathbf{DDH}^{\mathsf{A}} = 1] - \frac{1}{2}\right|.$$

2.4 (Partially) Blind Signature Schemes

In this section, we introduce the syntax and security definitions of partially blind (three-move) signature schemes [27]. We note that a fully blind signature scheme is a special case of a partially blind signature scheme where there is only one tag info, the empty string. We will refer to schemes where the tag is always the empty string as *blind signature schemes*.

Definition 5 (Three-Move Partially Blind Signature Scheme). *A three-move partially 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 parameters* 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 a tag* info *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*, a tag* info *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 *and a tag* info*. It outputs either 1 (accept) or 0 (reject).*

Definition 6 (Correcntess). *We say that a partially blind signature scheme* $\mathsf{BS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{User}, \mathsf{Verify})$ *is* correct *if for all messages* m*, all tags* info *the following holds:*

$$\Pr\left[\mathsf{Verify}(\mathsf{pk}, \mathsf{sig}, m, \mathsf{info}) = 1 \colon \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{pp}) \\ (C, \mathsf{st}_S) \xleftarrow{\$} \mathsf{Sign}_1(\mathsf{sk}, \mathsf{info}) \\ (e, \mathsf{st}_U) \xleftarrow{\$} \mathsf{User}_1(\mathsf{pk}, m, \mathsf{info}, C) \\ R \xleftarrow{\$} \mathsf{Sign}_2(\mathsf{sk}, \mathsf{st}_S, e) \\ \sigma \xleftarrow{\$} \mathsf{User}_2(\mathsf{pk}, \mathsf{st}_U, R) \end{array}\right] = 1$$

Definition 7 (Partial blindness under chosen keys). *We define* partial blindness *of a three-move partially blind signature scheme* BS *against an adversary* M *via the following game:*

Setup. *Sample* $b \xleftarrow{\$} \{0, 1\}$ *and run* M *on input* pp*.*

Online Phase. *When* M *outputs messages* $\tilde{m}_0$ *and* $\tilde{m}_1$, $\widetilde{\mathsf{info}_0}$ *and* $\widetilde{\mathsf{info}_1}$, *and a public key* pk, *check if* pk *is a valid*[5] *public key, and* $\widetilde{\mathsf{info}_0} = \widetilde{\mathsf{info}_1}$. *If so, assign* $m_0 := \tilde{m}_b$, $\mathsf{info}_0 := \widetilde{\mathsf{info}_0}$, $m_1 := \tilde{m}_{1-b}$, *and* $\mathsf{info}_1 := \widetilde{\mathsf{info}_1}$. *If* pk *is not a valid public key or* $\mathsf{info}_0 \neq \mathsf{info}_1$, *abort and output* 0. M *is given access to oracles* $\mathbf{User}_1, \mathbf{User}_2$, *which behave 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* $(\mathsf{st}_{b'}, e) \xleftarrow{\$} \mathsf{BS.User}_1(\mathsf{pk}, m_{b'}, C, \mathsf{info}_{b'})$. *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, return* 1 *if the adversary outputs a bit* b^* *s.t.* $b^* = b$. *Otherwise, return* 0.

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

$$\mathsf{Adv}^{\mathbf{BLIND},\mathsf{BS}}_{\mathsf{M}} := \left| \Pr\left[\mathbf{BLIND}^{\mathsf{M}} = 1\right] - \frac{1}{2} \right|.$$

Definition 8 (ℓ-(Sequential-)One-More-Unforgeability (ℓ-(SEQ-)OMUF)). *For a stateful algorithm* A, *a three-move partially blind signature scheme* BS, *and a positive integer* ℓ, *we define the game* ℓ-$\mathbf{OMUF}_{\mathsf{BS}}$ *(*ℓ-$\mathbf{SEQ\text{-}OMUF}_{\mathsf{BS}}$*) as follows:*

Setup. *Sample* $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{BS.KeyGen}(\mathsf{pp})$ *and run* A *on input* $(\mathsf{pk}, \mathsf{pp})$.

Online Phase. A *is given access to the oracles* $\mathbf{Sign}_1$ *and* $\mathbf{Sign}_2$ *that behave as follows.*

- **Oracle** $\mathbf{Sign}_1$**:** *On input* info, *it samples a fresh session identifier* $\mathtt{id}$ *(If sequential, it checks if* $\mathtt{session}_{\mathtt{id}-1} = \mathtt{open}$ *and returns* $\perp$ *if yes). If* info *has not been requested before, it initializes a counter* $\ell_{\mathtt{closed},\mathsf{info}} := 0$. *It sets* $\mathtt{session}_{\mathtt{id}} := \mathtt{open}$ *and generates* $(C_{\mathtt{id}}, \mathsf{st}_{\mathtt{id}}) \xleftarrow{\$} \mathsf{BS.Sign}_1(\mathsf{sk}, \mathsf{info})$. *Then it returns* $C_{\mathtt{id}}$ *and* $\mathtt{id}$.
- **Oracle** $\mathbf{Sign}_2$**:** *If* $\sum_{\mathsf{info}} \ell_{\mathtt{closed},\mathsf{info}} < \ell$, $\mathbf{Sign}_2$ *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* $\ell_{\mathtt{closed},\mathsf{info}} := \ell_{\mathtt{closed},\mathsf{info}} + 1$ *and* $\mathtt{session}_{\mathtt{id}} := \mathtt{closed}$. *Then it generates the response* R *via* $R \xleftarrow{\$} \mathsf{BS.Sign}_2(\mathsf{sk}, \mathsf{st}_{\mathtt{id}}, e)$ *and returns* R.

Output Determination. *When* A *outputs tuples* $(m_1, \sigma_1, \mathsf{info}_1), \ldots, (m_k, \sigma_k, \mathsf{info}_k)$, *return* 1 *if there exists a tag* $\overline{\mathsf{info}}$ *such that* $\left|\left\{(m_i, \sigma_i, \mathsf{info}_i) \middle| \mathsf{info}_i = \overline{\mathsf{info}}\right\}\right| \geq \ell_{\mathtt{closed},\overline{\mathsf{info}}} + 1$ *(where by convention* $\ell_{\mathtt{closed},\mathsf{info}} := 0$ *for any* info *that has not*

[5] We include this in case the scheme permits such a check - for example, one can think of schemes where the public key consists of group elements, in which case a user may be able to check that the public key consists of valid encodings of group elements. Another example of such a check is in the original version of Abe's scheme [1] where $\mathbf{z} = H_1(\mathbf{g}, \mathbf{h}, \mathbf{y})$ which a user may check.

been requested to the signing oracles) and for all $i \in [k] : \mathsf{BS.Verify}(\mathsf{pk}, \sigma_i, m_i, \mathsf{info}_i) = 1$ *and* $(m_i, \sigma_i, \mathsf{info}_i) \neq (m_j, \sigma_j, \mathsf{info}_j)$ *for all* $j \neq i$*. Otherwise, return* 0*.*

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

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

And, respectively for $\mathbf{SEQ\text{-}OMUF}_{\mathsf{BS}}$

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

3 Adaption of Abe's blind Signature Scheme to allow partial blindness

We begin by describing an adaption of Abe's blind signature scheme BSA [1] to the partially blind setting. A figure depicting an interaction between signer and user can be found in the full version [30]. 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 $\mathbf{y} \coloneqq \mathbf{g}^x$. It sets $\mathsf{sk} \coloneqq x$, $\mathsf{pk} \coloneqq (\mathbf{g}, \mathbf{h}, \mathbf{y})$ and returns $(\mathsf{sk}, \mathsf{pk})$.
- Sign_1: On input $\mathsf{sk}, \mathsf{info}$, Sign_1 samples $\mathrm{rnd} \xleftarrow{\$} \{0,1\}^\lambda$ and $u, d, s_1, s_2 \xleftarrow{\$} \mathbb{Z}_q$. It computes $\mathbf{z} \coloneqq H_1(\mathsf{pk}, \mathsf{info})$, $\mathbf{z}_1 \coloneqq H_2(\mathrm{rnd})$, $\mathbf{z}_2 \coloneqq \mathbf{z}/\mathbf{z}_1$, $\mathbf{a} \coloneqq \mathbf{g}^u$, $\mathbf{b_1} \coloneqq \mathbf{g}^{s_1} \cdot \mathbf{z}_1^d$, $\mathbf{b_2} \coloneqq \mathbf{h}^{s_2} \cdot \mathbf{z}_2^d$. It returns a commitment $(\mathrm{rnd}, \mathbf{a}, \mathbf{b_1}, \mathbf{b_2})$ and a state $\mathsf{st}_S = (u, d, s_1, s_2, \mathsf{info})$.
- Sign_2: On input a secret key sk, a challenge e, and state $\mathsf{st}_S = (u, d, s_1, s_2, \mathsf{info})$, Sign_2 computes $c \coloneqq e - d \mod q$, $r \coloneqq u - c \cdot \mathsf{sk} \mod q$ and returns the response (c, d, r, s_1, s_2).
- User_1: On input a public key pk and a commitment $(\mathrm{rnd}, \mathbf{a}, \mathbf{b_1}, \mathbf{b_2})$, a tag info, 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 $\mathbf{z} \coloneqq H_1(\mathsf{pk}, \mathsf{info})$, $\mathbf{z}_1 \coloneqq H_2(\mathrm{rnd})$, $\alpha \coloneqq \mathbf{a} \cdot \mathbf{g}^{t_1} \cdot \mathbf{y}^{t_2}$, $\zeta \coloneqq \mathbf{z}^\gamma$, $\zeta_1 \coloneqq \mathbf{z}_1^\gamma$, $\zeta_2 \coloneqq \zeta/\zeta_1$. Next, it sets $\beta_1 \coloneqq \mathbf{b}_1^\gamma \cdot \mathbf{g}^{t_3} \cdot \zeta_1^{t_4}$, $\beta_2 \coloneqq \mathbf{b}_2^\gamma \cdot \mathbf{h}^{t_5} \cdot \zeta_2^{t_4}$, $\eta \coloneqq \mathbf{z}^\tau$, and $\varepsilon \coloneqq H_3(\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta, m, \mathsf{info})$. Finally, it computes a challenge $e \coloneqq \varepsilon - t_2 - t_4 \mod q$, the state $St_U \coloneqq (\gamma, \tau, t_1, t_2, t_3, t_4, t_5, m)$ 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 $\rho \coloneqq r + t_1$, $\omega \coloneqq c + t_2$, $\sigma_1 \coloneqq \gamma \cdot s_1 + t_3$, $\sigma_2 \coloneqq \gamma \cdot s_2 + t_5$, and $\delta \coloneqq d + t_4$. Then, it computes $\mu \coloneqq \tau - \delta \cdot \gamma$ and $\varepsilon \coloneqq H_3(\zeta, \zeta_1, \mathbf{g}^\rho \mathbf{y}^\omega, \mathbf{g}^{\sigma_1} \zeta_1^\delta, \mathbf{h}^{\sigma_2} \zeta_2^\delta, \mathbf{z}^\mu \zeta^\delta, m)$. It returns the signature $\sigma \coloneqq (\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu)$ if $\delta + \omega = \varepsilon$; otherwise, it returns $\perp$.[6]

[6] We note that the check for $\varepsilon = \omega + \delta$ implicitly checks that $c + d = e$ as well as $\mathbf{a} = \mathbf{y}^c \mathbf{g}^r, \mathbf{b}_1 = \mathbf{z}_1^d \mathbf{g}^{s_1}, \mathbf{b}_2 = \mathbf{z}_2^d \mathbf{h}^{s_2}$, i.e. it checks that the output of $\mathsf{Sign} - 2$ was valid.

– 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 first $\mathbf{z} := H_1(\mathsf{pk}, \mathsf{info})$ and then $\varepsilon := H_3(\zeta, \zeta_1, \mathbf{g}^{\rho}\mathbf{y}^{\omega}, \mathbf{g}^{\sigma_1}\zeta_1^{\delta}, \mathbf{h}^{\sigma_2}\zeta_2^{\delta}, \mathbf{z}^{\mu}\zeta^{\delta}, m, \mathsf{info})$. It returns 1 if $\delta + \omega = \varepsilon$; otherwise, it returns 0.

We note that the only change we made to Abe's scheme is that in our variant, the $\mathbf{z}$ part of the public key is derived as a hash of pk and a tag info instead of as a hash of the other elements of the public key. It is easy to see that by using an empty info this yields the original scheme and thus our proofs about the adapted scheme also apply to the original.

We note that Abe refers to $\mathbf{z}, \mathbf{z}_1, \zeta, \zeta_1$ as the tags of a signing session or signature. However, as we are considering partial blindness, we will refer to them as the *linking components*. By [1], the original scheme is computationally blind under the Decisional Diffie-Hellman assumption. For completeness, we provide a detailed proof of the partial computational blindness of our variant in section 3.1.

3.1 Partial Blindness of the adapted Abe scheme

We provide a formal proof of partial blindness under chosen keys for the Abe blind signature scheme. Abe [1] proved the scheme to be blind for keys selected by the challenger.

Lemma 1. *Under the decisional Diffie-Hellman assumption in $\mathbb{G}$, Abe's blind signature scheme* BSA *is computationally blind in the random oracle model.*

Proof. We use similar techniques as [6].
Game $\mathbf{G_1}$ The first game is identical to the blindness game from Definition 7 for Abe's blind signature scheme.

Setup. $\mathbf{G_1}$ samples $b \xleftarrow{\$} \{0,1\}$.
Simulation of oracle H_1. $\mathbf{G_1}$ simulates H_1 by lazy sampling of group elements.
Online Phase. When M outputs a public key $(\mathbf{g}, \mathbf{y}, \mathbf{h})$ and messages $\tilde{m}_0$ and $\tilde{m}_1$, and tags info_0, info_1, $\mathbf{G_1}$ verifies $\mathsf{info}_0 = \mathsf{info}_1$ assigns $m_0 = \tilde{m}_b$ and $m_1 = \tilde{m}_{b-1}$
Oracle User$_1$. works the same as described in Definition 7
Oracle User$_2$. works the same as described in Definition 7
Simulation of H_2. H_2 is simulated through lazy sampling
Simulation of H_3. H_3 is simulated through lazy sampling
Output determination. as described in Definition 7

The second game replaces the signature for m_0 by a signature that is independent of the run with the signer.
Game $\mathbf{G_2}$ The second game generates the signature on m_0 independently of the corresponding signing session.

Setup. $\mathbf{G_2}$ samples $b \xleftarrow{\$} \{0,1\}$.

Simulation of oracle H_1. $\mathbf{G_2}$ simulates H_1 by lazy sampling of group elements.

Online Phase. When M outputs a public key $(\mathbf{g}, \mathbf{y}, \mathbf{h})$ and messages $\tilde{m}_0$ and $\tilde{m}_1$ and $\widetilde{\mathsf{info}_0}, \widetilde{\mathsf{info}_1}$, $\mathbf{G_2}$ verifies that the key is well-formed and that $\widetilde{\mathsf{info}_0} = \widetilde{\mathsf{info}_1}$ and aborts with output 0 if this check fails. It further assigns $m_0 = \tilde{m}_b$ and $m_1 = \tilde{m}_{b-1}$ as well as $\mathsf{info}_0 = \widetilde{\mathsf{info}_0}$ and $\mathsf{info}_1 = \widetilde{\mathsf{info}_1}$.

Oracle $\mathbf{User}_1$. For message m_1, the oracle behaves the same as in $\mathbf{G_1}$. For message m_0, it checks that session 0 is not open yet and opens session 0. Then the game picks $\delta, \omega, \sigma_1, \sigma_2, \rho, \mu$ uniformly at random from $\mathbb{Z}_q$. It further draws two random group elements ζ and ζ_1 and sets $\zeta_2 \coloneqq \zeta/\zeta_1$. It then sets $H_3(\mathbf{y}^\omega \cdot \mathbf{g}^\rho, \zeta_1^\delta \cdot \mathbf{g}^{\sigma_1}, \zeta_2^\delta \cdot \mathbf{h}^{\sigma_2}, \zeta^\delta \cdot \mathbf{z}^\mu, m_0, \mathsf{info}_0) \coloneqq \delta + \omega$. It draws $e \xleftarrow{\$} \mathbb{Z}_q$ uniformly at random and returns e as a challenge to the adversary.

Oracle $\mathbf{User}_2$. For message m_1, the oracle behaves the same as in $\mathbf{G_1}$. For message m_0, on input c, d, r, s_1, s_2, the game does the following checks[7]: $e = d + c$, $\mathbf{a}_0 = \mathbf{g}^r \cdot \mathbf{y}^c$, $\mathbf{b}_{1,0} = \mathbf{g}^{s_1} \cdot \mathbf{z}_{1,0}^d$, $\mathbf{b}_{2,0} = \mathbf{h}^{s_2} \cdot \mathbf{z}_{2,0}^d$. It considers the produced signature to be the one generated in $\mathbf{User}_1$.

Simulation of H_2. H_2 is simulated through lazy sampling

Simulation of H_3. For values not programmed in $\mathbf{User}_1$, $\mathbf{G_2}$ simulates H_3 via lazy sampling

Output determination. as described in Definition 7

Claim 1. The advantage of an adversary B to tell the difference between $\mathbf{G_1}$ and $\mathbf{G_2}$ is $\mathsf{Adv}_{\mathsf{B}}^{\mathbf{G_1},\mathbf{G_2}} = \left|\Pr\left[{\mathbf{G_1}}^{\mathsf{B}} = 1\right] - \Pr\left[{\mathbf{G_2}}^{\mathsf{B}} = 1\right]\right| \leq \mathsf{Adv}_{\mathsf{B}'}^{\mathbf{DDH}}$.

Proof. We provide a reduction B' that receives a random-generator DDH challenge $(\mathbf{W}, \mathbf{X}, \mathbf{Y}, \mathbf{Z})$ and simulates either $\mathbf{G_1}$ or $\mathbf{G_2}$ to the adversary. During the first phase of the online phase, the reduction programs the random oracle H_1 to return values $\mathbf{W}^{f_i}$ $f_i \in \mathbb{Z}_q$. For simulation of H_2, the reduction chooses exponents $g_i \xleftarrow{\$} \mathbb{Z}_q$ and returns values $\mathbf{X}^{g_i}$, yielding uniformly random values from the group $\mathbb{G}$. In $\mathbf{User}_1$ for m_0, when the adversary sends the commitment which contains a random string rnd to be queried to the oracle H_2, the reduction identifies the $g = g_i$ that was used as the random exponent for $\mathbf{z}_1 = \mathbf{X}^g$. Denote further by f the f_i used for generation of $\mathbf{z} = H_1(\mathsf{pk}, \mathsf{info}_1)$. It sets $\zeta = \mathbf{Y}^f$ and $\zeta_1 = \mathbf{Z}^{f \cdot g}$. The reduction then proceeds to generate a signature by programming the random oracle H_3 as described in $\mathbf{G_2}$. For m_1, the reduction participates honestly in the signing protocol. In $\mathbf{User}_2$, for m_0, the reduction checks that the adversary produces a valid signing transcript as described in $\mathbf{G_2}$. If both interactions yield valid signatures (i.e. the adversary produced a valid transcript for m_0 and a valid signature for m_1), the reduction outputs both signatures, otherwise $\perp$. If the adversary outputs it was playing game $\mathbf{G_1}$, the reduction outputs 0, otherwise it outputs 1.

We argue that if the challenge is a Diffie-Hellman tuple, the reduction simulates $\mathbf{G_1}$ perfectly. For a tuple $\mathbf{W}, \mathbf{W}^a, \mathbf{W}^b, \mathbf{W}^{ab}$, the tuple $\mathbf{z} = \mathbf{W}^f, \mathbf{z}_1 =$

[7] We note that these checks need to be done explicitly here, as they are no longer implicitly performed through checking that $\varepsilon = \omega + \delta$,

$\mathbf{W}^{a\cdot f\cdot \frac{g}{f}}, \zeta = \mathbf{W}^{b\cdot f}, \zeta_1 = \mathbf{W}^{a\cdot b\cdot f\cdot g}$ is a valid Diffie-Hellman tuple w.r.t generator $\mathbf{W}^f$. Furthermore, the user tags ζ and ζ_1 can be computed from $\mathbf{z}$ and $\mathbf{z}_1$ using blinding factor $\gamma = b$. Furthermore, for any c, d, r, s_1, s_2 and signature components $\omega, \delta, \rho, \sigma_1, \sigma_2, \mu$ there are unique choices of $t_1 = \rho - r, t_2 = \omega - c, t_3 = \sigma_1 - \gamma \cdot s_1, t_4 = \delta - d, t_5 = \sigma_2 - \gamma \cdot s_2, \tau = \mu + \delta \cdot \gamma$ that explain the signature in combination with the transcript. Thus, the produced combination of signature and transcript is identically distributed as an honestly generated signature.

If the challenge is not a Diffie-Hellman tuple, then the reduction simulates $\mathbf{G_2}$ perfectly as the linking components $\zeta_i, \zeta_{1,i}$ look like random group elements and the reduction computes the same steps as $\mathbf{G_2}$ to generate the signatures and its outputs to the adversary. □

We describe the final game $\mathbf{G_3}$ where both signatures are independent from the runs with the signer.

Game $\mathbf{G_3}$

Setup. $\mathbf{G_3}$ samples $b \xleftarrow{\$} \{0,1\}$.

Simulation of oracle H_1. $\mathbf{G_3}$ simulates H_1 by lazy sampling of group elements.

Online Phase. When M outputs a public key $(\mathbf{g}, \mathbf{y}, \mathbf{h})$ and messages $\tilde{m}_0$ and $\tilde{m}_1$, $\mathbf{G_3}$ verifies that the key is well-formed and checks that $\mathsf{info}_0 = \mathsf{info}_1$ and aborts with output 0 if this check fails. It further assigns $m_0 = \tilde{m}_b$ and $m_1 = \tilde{m}_{b-1}$

Oracle $\mathbf{User}_1$. For session b', the game checks that session b' is not open yet and opens session b'. It sets $\mathbf{z} := H_1(\mathsf{info})$. Then the game picks $\delta, \omega, \sigma_1, \sigma_2, \rho, \mu$ uniformly at random from $\mathbb{Z}_q$. It further draws two random group elements ζ and ζ_1 and sets $\zeta_2 := \zeta/\zeta_1$. It then sets $H_3(\mathbf{y}^\omega \cdot \mathbf{g}^\rho, \zeta_1^\delta \cdot \mathbf{g}^{\sigma_1}, \zeta_2^\delta \cdot \mathbf{h}^{\sigma_2}, \zeta^\delta \cdot \mathbf{z}^\mu, m_{b'}, \mathsf{info}_{b'}) := \delta + \omega$. It draws $e \xleftarrow{\$} \mathbb{Z}_q$ uniformly at random and returns e as a challenge to the adversary.

Oracle $\mathbf{User}_2$. For both sessions (denoted by $i = 0, 1$), on input $c_i, d_i, r_i, s_{1,i}, s_{2,i}$, the game does the following checks: $e_i = d_i + c_i$, $\mathbf{a}_i = \mathbf{g}^{r_i} \cdot \mathbf{y}^{c_i}$, $\mathbf{b}_{1,i} = \mathbf{g}^{s_{1,i}} \cdot \mathbf{z}_{1,i}^{d_i}$, $\mathbf{b}_{2,i} = \mathbf{h}^{s_{2,i}} \cdot \mathbf{z}_{2,i}^{d_i}$. It considers the output signature to be the one generated for this session in $\mathbf{User}_1$.

Simulation of H_2. H_2 is simulated through lazy sampling

Simulation of H_3. For values not programmed in $\mathbf{User}_1$, $\mathbf{G_2}$ simulates H_3 via lazy sampling

Output determination. as described in Definition 7

Claim 2. The advantage of an adversary B *to tell the difference between* $\mathbf{G_1}$ *and* $\mathbf{G_2}$ *is* $\mathsf{Adv}_{\mathsf{B}'''}^{\mathbf{G_2},\mathbf{G_3}} = \Pr\left[{\mathbf{G_2}}^{\mathsf{B}'''} = 1\right] - \Pr\left[{\mathbf{G_3}}^{\mathsf{B}'''} = 1\right] \leq \mathsf{Adv}_{\mathsf{B}''}^{\mathbf{DDH}}$.

Proof. Follows along the same lines as Claim 1, embedding the **DDH** challenge in the signature for m_1 this time. □

In game $\mathbf{G_3}$, the adversary cannot win, as both signatures are completely independent from the two runs. As game $\mathbf{G_3}$ needs to program the random oracle H_3 twice to generate the signatures (this fails with probability at most $\frac{2q_h}{q^4 \cdot 2^{|m_0|}}$, i.e.

if the adversary has made the exact same requests before), we get the following overall advantage of

$$\mathsf{Adv}_{\mathsf{M}}^{\mathbf{BLIND}_{\mathsf{BSA}}} = \frac{2 \cdot q_h}{q^4 \cdot 2^{|m_0|}} + \mathsf{Adv}_{\mathsf{B}'}^{\mathbf{DDH}} + \mathsf{Adv}_{\mathsf{B}''}^{\mathbf{DDH}}$$

□

3.2 One-More-Unforgeability

In the following, we provide a proof for the one-more-unforgeability. Similar to [1] 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. (Note that the scheme is only computationally blind, and an unbounded algorithm can link signatures and sessions since $(\mathbf{z}, \mathbf{z}_1, \zeta, \zeta_1)$ forms a DDH tuple. We call such tuples *linking components*, and refer to $\mathbf{z}, \mathbf{z}_1$ as "signer-side" and ζ, ζ_1 as "user-side".) 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 ℓ-**OMUF** 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 in the tags $\mathbf{z}$. 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 one of the $\mathbf{z}$. 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 identify which signature is the "forgery". This, in combination with not having to run the protocol twice for forking, renders a tight proof.

Comparison to the original standard model proof by Abe [1] We briefly recall that similar to our proof, the original proof also shows the restrictive blinding lemma first, which, shows that an adversary that wins the **OMUF** game and at the same time produces a signature where $\mathrm{dlog}_{\zeta}\, \zeta_1 \neq \mathrm{dlog}_{\mathbf{z}}\, \mathbf{z}_{1,i}$ for all sessions i, can be used to solve the discrete logarithm problem. The proof uses the forking technique, i.e. it rewinds the adversary to obtain a second set of signatures with different hash responses to H_3. The original proof of the restrictive blinding lemma also uses two signers, one that embeds in $\mathbf{y}$ and signs using the $\mathbf{z}$-side witness, another that embeds in $\mathbf{h}$ and signs using the secret key x. These two signers are indistinguishable for a single run, however, two forking

runs using the same witness reveal the witness being used internally. In particular, a forking pair of runs using the secret key x to sign, cannot be reproduced by a signer that does not know the x-side witness. Therefore, the distribution of signatures obtained from forking runs, in particular the components δ and ω may depend on which witness was used internally. We note that for example in 'honestly generated' signatures (i.e. when the adversary followed the User_1 and User_2 algorithms to generate signatures), the a pair of signatures at the forking hash query reveals exactly the same witness as the signer used to sign while forking, so it is not clear why a similar thing may not also hold for 'dishonestly generated' signatures.

As our reduction for the restrictive blinding lemma works in the AGM, we can avoid the rewinding step. The adversary submits representations of all the group elements contained in a hash query, which gives the reduction information that would otherwise be obtained from the previous run. As the scheme is perfectly witness indistinguishable, the representations submitted by the adversary are independent of the witness used internally. We show in Claim 5, that even a so-called *reduced representation* that does use factors that are only determined after all signing sessions were closed, is likely to reveal enough information for the reduction to be able to solve the discrete logarithm problem.

The Restrictive Blinding Lemma. We first provide a reduction for the restrictive blinding lemma in the AGM + ROM. We therefore define the game ℓ-**RB-OMUF**$_{\mathsf{BSA}}$ as follows:

Setup: Sample keys via $(\mathsf{sk} = x, \mathsf{pk} = (\mathbf{g}, \mathbf{h}, \mathbf{y})) \xleftarrow{\$} \mathsf{BSA.KeyGen}(\mathsf{pp})$.

Online Phase: M is given access to oracles $\mathbf{Sign}_1, \mathbf{Sign}_2$ that emulate the behavior of the honest signer in BSA. It is allowed to arbitrarily many calls to $\mathbf{Sign}_1$ and allowed to make ℓ queries to $\mathbf{Sign}_2$. In addition, it is given access to random oracles H_1, H_2, H_3. Let ℓ_{info} denote the number of interactions that M completes with oracle $\mathbf{Sign}_2$ in this phase for each tag info.

Output Determination: When M outputs a list L of tuples $(m_1, \mathsf{sig}_1, \mathsf{info}_1), \ldots, (m_k, \mathsf{sig}_k, \mathsf{info}_k)$, proceed as follows:

- If the list contains a tuple $(m, \mathsf{sig}, \mathsf{info})$ s.t. $\mathsf{Verify}(\mathsf{pk}, m, \mathsf{sig}, \mathsf{info}) = 0$, or does not contain $\ell_{\overline{\mathsf{info}}} + 1$ pairwise-distinct tuples for some tag $\overline{\mathsf{info}}$, return 0.
- Let $\mathbf{z}_j, \mathbf{z}_{1,j}$ denote the values of $\mathbf{z}$ and $\mathbf{z}_1$ used in the j-th invocation of $\mathbf{Sign}_1$. If there exists $(m, \mathsf{sig}, \overline{\mathsf{info}}) \in L$ with signature components $\zeta \neq \zeta_1$ (equivalently, $\zeta_2 \neq \epsilon$), s.t. for all j with $H_1(\mathsf{pk}, \overline{\mathsf{info}}) = \mathbf{z}_j$ whose sessions were closed with an invocation of $\mathbf{Sign}_2$, $\zeta^{\mathrm{dlog}_{\mathbf{z}_j} \mathbf{z}_{1,j}} \neq \zeta_1$, then return 1. Otherwise, return 0. We call the first signature in L with these mismatched linking components the *special signature*.

Define $\mathsf{Adv}^{\mathbf{RB\text{-}OMUF}}_{\mathsf{M},\ell,\mathsf{BSA}} := \Pr[\ell\text{-}\mathbf{RB\text{-}OMUF}^{\mathsf{M}}_{\mathsf{BSA}} = 1]$. We show that an algebraic forger M that wins ℓ-**RB-OMUF**$_{\mathsf{BSA}}$ can be used to solve the discrete logarithm problem. This reduction is tight and does not require rewinding of the adversary.

Lemma 2 (Restrictive Blinding, see Lemma 3 in [1]). *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.*

$$\begin{aligned}\mathsf{Adv}^{\mathbf{DLP}}_{\mathsf{B}} \geq & \frac{1}{2}\mathsf{Adv}^{\mathbf{RB\text{-}OMUF}}_{\mathsf{M},\ell,\mathsf{BSA}} - \frac{\ell+1}{2q} \\ & - (\frac{3q_h}{q} + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{R}_1} + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{R}_2} + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{R}_3} + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{R}_4})\end{aligned}$$

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 outputs a list of $\ell+1$ tuples. The proof goes by a series of games, which we describe below.

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

Game $\mathbf{G_1}$. To define $\mathbf{G_1}$, 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 $(m, \mathsf{sig}, \mathsf{info}) = (m, (\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu), \mathsf{info}) \in L$ s.t. for all j whose sessions were closed with an invocation of $\mathbf{Sign}_2$, $\zeta^{\mathrm{dlog}_{\mathbf{z}_j}\mathbf{z}_{1,j}} \neq \zeta_1 \wedge \zeta_2 \neq \epsilon$ and M did not make a query of the form $H_3(\zeta, \zeta_1, \mathbf{g}^\rho\mathbf{y}^\omega, \mathbf{g}^{\sigma_1}\zeta_1^\delta, \mathbf{h}^{\sigma_2}\zeta_2^\delta, \mathbf{z}^\mu\zeta^\delta, m, \mathsf{info})$. In the following, we refer to the first tuple $(m, \mathsf{sig}, \mathsf{info}) \in L$ 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 3. $\Pr[E_1] = \frac{\ell+1}{q}$

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 and info.

As M is algebraic, it also provides a representation of these group elements to the basis of elements $\mathbf{g}, \mathbf{h}, \mathbf{y}, \overrightarrow{\mathbf{z}}, \overrightarrow{\mathbf{a}}, \overrightarrow{\mathbf{b}_1}, \overrightarrow{\mathbf{b}_2}, \overrightarrow{\mathbf{z}_1}$ that it has previously obtained via calls to $H_1, H_2, \mathbf{Sign}_1$, or $\mathbf{Sign}_2$. We note that by programming the oracles H_1 and H_2 the reduction knows a representation of its responses $\mathbf{z}_i$ and $\mathbf{z}_{1,i}$. 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 $[\mathbf{o}]_{\overrightarrow{I}}$ and its components as $g_{[\mathbf{o}]_{\overrightarrow{I}}}, h_{[\mathbf{o}]_{\overrightarrow{I}}}, y_{[\mathbf{o}]_{\overrightarrow{I}}}$, respectively, where $\overrightarrow{I} := (\mathbf{g}, \mathbf{h}, \mathbf{y})$. If M wins, we denote the special message/signature pair in its winning output as $(m, \mathsf{info}, (\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu))$. The algebraic coefficients of this tuple define the following integers which we call "preliminary values":

$$\begin{aligned}
\omega' &:= y_{[\alpha]_{\overrightarrow{I}}} \\
\delta' &:= \frac{g_{[\beta_2]_{\overrightarrow{I}}} + x \cdot y_{[\beta_2]_{\overrightarrow{I}}}}{x \cdot y_{[\zeta_2]_{\overrightarrow{I}}} + g_{[\zeta_2]_{\overrightarrow{I}}}} \\
\delta'' &:= \frac{h_{[\beta_1]_{\overrightarrow{I}}}}{h_{[\zeta_1]_{\overrightarrow{I}}}}, \delta''' := \frac{h_{[\eta]_{\overrightarrow{I}}}}{h_{[\zeta]_{\overrightarrow{I}}}}.
\end{aligned}$$

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

$$\begin{aligned}
C_0 &:= (\omega' \neq \omega) & C_1 &:= (\omega' = \omega) \wedge (x \cdot y_{[\zeta_2]_{\overrightarrow{I}}} + g_{[\zeta_2]_{\overrightarrow{I}}} \neq 0) \\
C_2 &:= (\omega' = \omega) \wedge (h_{[\zeta_1]_{\overrightarrow{I}}} \neq 0) & C_3 &:= (\omega' = \omega) \wedge (h_{[\zeta]_{\overrightarrow{I}}} \neq 0)
\end{aligned}$$

Claim 4. $\bigvee_i C_i = 1$.

Proof. Since $\mathbf{G_2^M} = 1 \Rightarrow \zeta_2 \neq \epsilon$, it follows that $x \cdot y_{[\zeta_2]_{\overrightarrow{I}}} + g_{[\zeta_2]_{\overrightarrow{I}}}$ and $h_{[\zeta_2]_{\overrightarrow{I}}}$ cannot both be 0 when $\mathbf{G_2^M} = 1$. 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 $[\zeta_2]_{\overrightarrow{I}} = [\zeta]_{\overrightarrow{I}} - [\zeta_1]_{\overrightarrow{I}}$, either $h_{[\zeta_1]_{\overrightarrow{I}}} \neq 0$ or $h_{[\zeta]_{\overrightarrow{I}}} \neq 0$, whenever $h_{[\zeta_2]_{\overrightarrow{I}}} \neq 0$. Therefore, $(h_{[\zeta_1]_{\overrightarrow{I}}} \neq 0 \vee h_{[\zeta]_{\overrightarrow{I}}} \neq 0 \vee x \cdot y_{[\zeta_2]_{\overrightarrow{I}}} + g_{[\zeta_2]_{\overrightarrow{I}}} \neq 0) = 1$ and thus $C_1 \vee C_2 \vee C_3 = (\omega' = \omega)$. The lemma follows immediately. □

We now define E_2 as the following event: $\omega' = \omega$, and for any of $\delta', \delta'', \delta'''$, as long as its denominator is not 0 (i.e., it is well-defined), then it is equal to δ. That is,

$$\begin{aligned}
E_2 :=& (C_0 = 0) \wedge (C_1 = 0 \vee (C_1 = 1 \wedge (\delta' = \delta))) \\
& \wedge (C_2 = 0 \vee (C_2 = 1 \wedge (\delta'' = \delta))) \wedge (C_3 = 0 \vee (C_3 = 1 \wedge (\delta''' = \delta))).
\end{aligned}$$

Claim 5. $\Pr[E_2] \leq \frac{3q_h}{q} + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{R}_1} + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{R}_2} + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{R}_3} + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{R}_4}$

The proof for this claim can be found in the full version [30].

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, S_1 simulate the random oracles H_1, H_2, H_3. These simulations are common to both S_ι 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 $\varepsilon \xleftarrow{\$} \mathbb{Z}_q$ and returns ε. It stores $(\xi, \overrightarrow{\text{rep}}, \varepsilon)$ 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 $\mathbf{G_2}$. 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., $\mathbf{G_2}^{\mathsf{M}} = 1$. 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 $(m, \mathsf{info}, \mathsf{sig})$ be special, i.e., that $\zeta^{\text{dlog}_{\mathbf{z}_j} \mathbf{z}_{1,j}} \neq \zeta_1$ for all $\mathbf{z}_j$ and $\mathbf{z}_{1,j}$ (where again $\mathbf{z}_j$ and $\mathbf{z}_{1,j}$ corresponds to the value of $\mathbf{z}$ and $\mathbf{z}_1$, respectively, derived during the j-th interaction with oracle $\mathbf{Sign}_1$).

From the verification equation of the special signature $(m, \mathsf{info}, \mathsf{sig})$, one obtains the equations $\alpha = \mathbf{g}^\rho \cdot \mathbf{y}^\omega$, $\beta_1 = \zeta_1^\delta \cdot \mathbf{g}^{\sigma_1}$, $\beta_2 = \zeta_2^\delta \cdot \mathbf{h}^{\sigma_2}$, $\eta = \mathbf{z}_j^\mu \cdot \zeta^\delta$. Denoting $w_{0,j} \coloneqq \text{dlog}\, \mathbf{z}_j$, $w \coloneqq \text{dlog}\, \mathbf{h}$, we obtain the reduced equations

$$g_{[\alpha]_{\overrightarrow{I}}} + x \cdot y_{[\alpha]_{\overrightarrow{I}}} + w \cdot h_{[\alpha]_{\overrightarrow{I}}} = \rho + x \cdot \omega \tag{1}$$

$$g_{[\beta_1]_{\overrightarrow{I}}} + x \cdot y_{[\beta_1]_{\overrightarrow{I}}} + w \cdot h_{[\beta_1]_{\overrightarrow{I}}} = (g_{[\zeta_1]_{\overrightarrow{I}}} + w \cdot h_{[\zeta_1]_{\overrightarrow{I}}} + x \cdot y_{[\zeta_1]_{\overrightarrow{I}}}) \cdot \delta + \sigma_1 \tag{2}$$

$$g_{[\beta_2]_{\overrightarrow{I}}} + x \cdot y_{[\beta_2]_{\overrightarrow{I}}} + w \cdot h_{[\beta_2]_{\overrightarrow{I}}} = (g_{[\zeta_2]_{\overrightarrow{I}}} + w \cdot h_{[\zeta_2]_{\overrightarrow{I}}} + x \cdot y_{[\zeta_2]_{\overrightarrow{I}}}) \cdot \delta + \sigma_2 \cdot w \tag{3}$$

$$g_{[\eta]_{\overrightarrow{I}}} + w \cdot h_{[\eta]_{\overrightarrow{I}}} + x \cdot y_{[\eta]_{\overrightarrow{I}}} = w_{0,j} \cdot \mu + (g_{[\zeta]_{\overrightarrow{I}}} + w \cdot h_{[\zeta]_{\overrightarrow{I}}} + x \cdot y_{[\zeta]_{\overrightarrow{I}}}) \cdot \delta. \tag{4}$$

We continue by describing simulators S_0 which covers case C_0, and S_1 which covers C_1, C_2, 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 all $\mathbf{z}_i$ 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 $\mathbf{U} \coloneqq \mathbf{g}^x$, it behaves as follows:

Setup: S_0 samples $w \xleftarrow{\$} \mathbb{Z}_q$ and computes the public key pk as $\mathsf{pk} \coloneqq (\mathbf{g}, \mathbf{h} \coloneqq \mathbf{g}^w, \mathbf{y} \coloneqq \mathbf{U})$, which implicitly sets $\mathsf{sk} \coloneqq x$.

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 $\mathbf{Sign}_1(\mathsf{info})$ to open session sid, S_0 checks in L_1 if $\mathsf{pk}, \mathsf{info}$ has been previously requested from H_1 and if yes sets $w_{0,\text{sid}}$ accordingly, otherwise samples $w_{0,\text{sid}}$ and programs $H_1(\mathsf{pk}, \mathsf{info}) \coloneqq \mathbf{g}^{w_{0,j}}$. It samples $\text{rnd}_{\text{sid}} \xleftarrow{\$} \{0,1\}^\lambda$ and sets $\mathbf{z}_{1,\text{sid}} \coloneqq \mathbf{g}^{w_{1,\text{sid}}} = H_2(\text{rnd}_{\text{sid}})$,

which places the tuple $(\mathrm{rnd}_{\mathrm{sid}}, \mathbf{z}_{1,\mathrm{sid}}, w_{1,\mathrm{sid}})$ into L_2. It then sets $\mathbf{z}_{2,\mathrm{sid}} \coloneqq \mathbf{z}_{\mathrm{sid}}/\mathbf{z}_{1,\mathrm{sid}}$, $w_{2,\mathrm{sid}} \coloneqq \frac{w_{0,\mathrm{sid}} - w_{1,\mathrm{sid}}}{w}$, $c_{\mathrm{sid}}, r_{\mathrm{sid}}, u_{1,\mathrm{sid}}, u_{2,\mathrm{sid}} \xleftarrow{\$} \mathbb{Z}_q$, $\mathbf{a}_{\mathrm{sid}} \coloneqq \mathbf{y}^{c_{\mathrm{sid}}} \cdot \mathbf{g}^{r_{\mathrm{sid}}}$, $\mathbf{b_1}_{,\mathrm{sid}} \coloneqq \mathbf{g}^{u_{1,\mathrm{sid}}}$, $\mathbf{b_2}_{,\mathrm{sid}} \coloneqq \mathbf{h}^{u_{2,\mathrm{sid}}}$ 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 $d_{\mathrm{sid}} \coloneqq e_{\mathrm{sid}} - c_{\mathrm{sid}}$, $s_{1,\mathrm{sid}} \coloneqq u_{1,\mathrm{sid}} - d_{\mathrm{sid}} \cdot w_{1,\mathrm{sid}}$, $s_{2,\mathrm{sid}} \coloneqq u_{2,\mathrm{sid}} - d_{\mathrm{sid}} \cdot w_{2,\mathrm{sid}}$ 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. (1) and the fact that $C_0 = 1 \Leftrightarrow \omega \neq y_{[\alpha]_{\overrightarrow{I}}}$, to (efficiently) compute and output the value x as $x = (\rho - g_{[\alpha]_{\overrightarrow{I}}} - w \cdot h_{[\alpha]_{\overrightarrow{I}}})/(y_{[\alpha]_{\overrightarrow{I}}} - \omega)$. (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 $C_0 = 0$ We describe simulator S_1, which simulates $\mathbf{G_2}$ using x. On input a discrete logarithm instance $\mathbf{U} \coloneqq \mathbf{g}^w$, it behaves as follows.

Setup. S_1 samples $x \xleftarrow{\$} \mathbb{Z}_q$. It sets $\mathsf{pk} \coloneqq (\mathbf{g}, \mathbf{h} \coloneqq \mathbf{U}, \mathbf{y} \coloneqq \mathbf{g}^x)$, $\mathsf{sk} \coloneqq x$.

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 $\mathbf{Sign}_1(\mathsf{info})$ to open session sid, S_1 checks if info was requested to H_1 already and if so sets $w_{0,j}$ accordingly, otherwise it samples $w_{0,j} \xleftarrow{\$} \mathbb{Z}_q$ and sets $H_1(\mathsf{pk}, \mathsf{info}) \coloneqq w_{0,j}$. It then samples $\mathrm{rnd}_{\mathrm{sid}} \xleftarrow{\$} \{0,1\}^\lambda$ and sets $\mathbf{z}_{1,\mathrm{sid}} \coloneqq \mathbf{g}^{w_{1,\mathrm{sid}}} = H_2(\mathrm{rnd}_{\mathrm{sid}})$ (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 $\mathbf{a}_{\mathrm{sid}} \coloneqq \mathbf{g}^{u_{\mathrm{sid}}}$, $\mathbf{b}_{1,\mathrm{sid}} \coloneqq \mathbf{g}^{s_{1,\mathrm{sid}}} \cdot \mathbf{z}_{1,\mathrm{sid}}^{d_{\mathrm{sid}}}$, $\mathbf{b}_{2,\mathrm{sid}} \coloneqq \mathbf{h}^{s_{2,\mathrm{sid}}} \cdot \mathbf{z}_{2,\mathrm{sid}}^{d_{\mathrm{sid}}}$ 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 $c_{\mathrm{sid}} \coloneqq e_{\mathrm{sid}} - d_{\mathrm{sid}}$, $r_{\mathrm{sid}} \coloneqq u_{\mathrm{sid}} - c_{\mathrm{sid}} \cdot x$ 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. Since $C_0 = 0$ it holds that $C_1 = 1 \vee C_2 = 1 \vee C_3 = 1$. S_1 uses one of the following extraction strategies.

If $C_1 = 1$: S_1 uses Eq. (3) and the fact that $C_1 = 1 \Rightarrow (x \cdot y_{[\zeta_2]_{\overrightarrow{I}}} + g_{[\zeta_2]_{\overrightarrow{I}}} \neq 0)$, to (efficiently) compute and output the value w as follows. S_1 first computes δ' as $\delta' \coloneqq (g_{[\beta_2]_{\overrightarrow{I}}} + x \cdot y_{[\beta_2]_{\overrightarrow{I}}})/(x \cdot y_{[\zeta_2]_{\overrightarrow{I}}} + g_{[\zeta_2]_{\overrightarrow{I}}})$, which gives the equality

$$\delta' \cdot (g_{[\zeta_2]_{\overrightarrow{I}}} + x \cdot y_{[\zeta_2]_{\overrightarrow{I}}}) + w \cdot h_{[\beta_2]_{\overrightarrow{I}}} = g_{[\beta_2]_{\overrightarrow{I}}} + x \cdot y_{[\beta_2]_{\overrightarrow{I}}} + w \cdot h_{[\beta_2]_{\overrightarrow{I}}}. \quad (5)$$

Eqs. (5) and (3) yield

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

If $h_{[\beta_2]_{\overrightarrow{I}}} - \delta \cdot h_{[\zeta_2]_{\overrightarrow{I}}} - \sigma_2 \neq 0$, S_1 outputs $w = ((\delta - \delta') \cdot (g_{[\zeta_2]_{\overrightarrow{I}}} + x \cdot h_{[\zeta_2]_{\overrightarrow{I}}}))/(h_{[\beta_2]_{\overrightarrow{I}}} - \delta \cdot h_{[\zeta_2]_{\overrightarrow{I}}} - \sigma_2)$. We prove the following claim.

Claim 6. $h_{[\beta_2]_{\overrightarrow{I}}} - \delta \cdot h_{[\zeta_2]_{\overrightarrow{I}}} - \sigma_2 \neq 0$.

Proof. Since $C_1 = 1$ and event E_2 does not happen (since otherwise $\mathbf{G}_{\mathbf{2}}^{\mathsf{M}} = 0$), we know that $\delta \neq \delta'$. Hence, it suffices to show that if $\delta \neq \delta'$, then $h_{[\beta_2]_{\overrightarrow{I}}} - \delta \cdot h_{[\zeta_2]_{\overrightarrow{I}}} - \sigma_2 \neq 0$. Due to Eq. (3) we get

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

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

If $C_1 = 0$ and $C_2 = 1$: S_1 uses Eq. (2) and the fact that $C_2 = 1 \Leftrightarrow (\omega = y_{[\alpha]_{\overrightarrow{I}}}) \wedge (h_{[\zeta_1]_{\overrightarrow{I}}} \neq 0)$, to compute and output the discrete logarithm w of the instance $\mathbf{U}$ as follows. S_1 first computes $\delta'' \coloneqq \frac{h_{[\beta_1]_{\overrightarrow{I}}}}{h_{[\zeta_1]_{\overrightarrow{I}}}}$ which leads to the equality

$$\delta'' \cdot w \cdot h_{[\zeta_1]_{\overrightarrow{I}}} + g_{[\beta_1]_{\overrightarrow{I}}} + x \cdot y_{[\beta_1]_{\overrightarrow{I}}} = g_{[\beta_1]_{\overrightarrow{I}}} + x \cdot y_{[\beta_1]_{\overrightarrow{I}}} + w \cdot h_{[\beta_1]_{\overrightarrow{I}}}. \quad (6)$$

Eqs. (6) and (2) yield

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

By the same argument as in the previous case, $\delta \neq \delta''$, and S_1 can compute and output w as $w = (\delta \cdot (g_{[\zeta_1]_{\overrightarrow{I}}} + x \cdot y_{[\zeta_1]_{\overrightarrow{I}}}) + \sigma_1 - g_{[\beta_1]_{\overrightarrow{I}}} + x \cdot y_{[\beta_1]_{\overrightarrow{I}}})/((\delta - \delta'') \cdot h_{[\zeta_1]_{\overrightarrow{I}}})$, as $C_2 = 1$ implies that $h_{[\zeta_1]_{\overrightarrow{I}}} \neq 0$. Moreover, S_1's simulation of $\mathbf{G_2}$ is perfect if $C_2 = 1$ holds.

If $C_1 = C_2 = 0$ and $C_3 = 1$: In this case, S_1 uses Eq. (4) and the fact that $C_3 = 1 \Leftrightarrow (\omega = y_{[\alpha]_{\overrightarrow{I}}}) \wedge (h_{[\zeta]_{\overrightarrow{I}}} \neq 0)$, to compute and output the discrete logarithm w of the instance $\mathbf{U}$ as we described below. S_1 computes $\delta''' \coloneqq h_{[\eta]_{\overrightarrow{I}}}/h_{[\zeta]_{\overrightarrow{I}}}$, leading to

$$\delta''' \cdot w \cdot h_{[\zeta]_{\overrightarrow{I}}} + g_{[\eta]_{\overrightarrow{I}}} + x \cdot y_{[\eta]_{\overrightarrow{I}}} = g_{[\eta]_{\overrightarrow{I}}} + x \cdot y_{[\eta]_{\overrightarrow{I}}} + w \cdot h_{[\eta]_{\overrightarrow{I}}}. \quad (7)$$

Equations (4) and (7) imply that

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

As in the previous cases, $\delta \neq \delta'''$, so S_1 can output w by computing $w = (\delta \cdot (g_{[\zeta]_{\overrightarrow{l}}} + x \cdot y_{[\zeta]_{\overrightarrow{l}}}) + \mu \cdot w_0 - (g_{[\eta]_{\overrightarrow{l}}} + x \cdot y_{[\eta]_{\overrightarrow{l}}}))/((\delta''' - \delta) \cdot h_{[\zeta]_{\overrightarrow{l}}})$, since $h_{[\zeta]_{\overrightarrow{l}}} \neq 0$ due to $C_3 = 1$. Moreover, S_1's simulation of $\mathbf{G_2}$ is perfect if $C_3 = 1$ holds. Since both simulators provide a perfect simulation (in their respective cases) and cover all cases that can happen whenever $\mathbf{G}_2^{\mathsf{M}} = 1$, B can run the correct simulator to extract the discrete logarithm with advantage $\mathsf{Adv}_{\mathsf{B}}^{\mathbf{DLP}} \geq \mathsf{Adv}_{\mathsf{M}}^{\mathbf{G_2}}/2$. Moreover, we have $\mathsf{Adv}_{\mathsf{M}}^{\mathbf{G_2}} \geq \mathsf{Adv}_{\mathsf{M}}^{\mathbf{G_1}} - -(\frac{3q_h}{q} + \mathsf{Adv}_{\mathsf{R}_1}^{\mathrm{dlog}} + \mathsf{Adv}_{\mathsf{R}_2}^{\mathrm{dlog}} + \mathsf{Adv}_{\mathsf{R}_3}^{\mathrm{dlog}} + \mathsf{Adv}_{\mathsf{R}_4}^{\mathrm{dlog}}) \geq \mathsf{Adv}_{\mathsf{M}}^{\mathbf{G_0}} - -(\frac{3q_h}{q} + \mathsf{Adv}_{\mathsf{R}_1}^{\mathrm{dlog}} + \mathsf{Adv}_{\mathsf{R}_2}^{\mathrm{dlog}} + \mathsf{Adv}_{\mathsf{R}_3}^{\mathrm{dlog}} + \mathsf{Adv}_{\mathsf{R}_4}^{\mathrm{dlog}}) - \frac{\ell+1}{q}$. Hence, $t_{\mathsf{B}} \approx t_{\mathsf{M}}$ and

$$\begin{aligned}\mathsf{Adv}_{\mathsf{B}}^{\mathbf{DLP}} \geq & \frac{1}{2}\mathsf{Adv}_{\mathsf{M},\ell,\mathsf{BSA}}^{\mathbf{RB\text{-}OMUF}} - \frac{\ell+1}{2q} \\ & - (\frac{3q_h}{q} + \mathsf{Adv}_{\mathsf{R}_1}^{\mathrm{dlog}} + \mathsf{Adv}_{\mathsf{R}_2}^{\mathrm{dlog}} + \mathsf{Adv}_{\mathsf{R}_3}^{\mathrm{dlog}} + \mathsf{Adv}_{\mathsf{R}_4}^{\mathrm{dlog}})\end{aligned}$$

□

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 1. *Let* M *be an algebraic algorithm that runs in time* t_{M}*, makes at most* ℓ *queries to oracle* $\mathbf{Sign}_2$ *in* ℓ-$\mathbf{OMUF}_{\mathsf{BSA}}$ *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*

$$\begin{aligned}\mathsf{Adv}_{\mathsf{B}}^{\mathbf{DLP}} \geq & \frac{1}{4}\mathsf{Adv}_{\mathsf{M},\ell,\mathsf{BSA}}^{\mathbf{OMUF}} - \frac{3q_h}{q} - \mathsf{Adv}_{\mathsf{R}_1}^{\mathbf{DLP}} - \mathsf{Adv}_{\mathsf{R}_2}^{\mathbf{DLP}} - \mathsf{Adv}_{\mathsf{R}_3}^{\mathbf{DLP}} \\ & - (\mathsf{Adv}_{\mathsf{R}'_1}^{\mathbf{DLP}} + \mathsf{Adv}_{\mathsf{R}'_2}^{\mathbf{DLP}} + \mathsf{Adv}_{\mathsf{R}'_3}^{\mathbf{DLP}} + \mathsf{Adv}_{\mathsf{R}'_4}^{\mathbf{DLP}})\end{aligned}$$

Proof. The proof is similar to the proof of lemma 2. We give a brief overview, the full proof can be found in the full version [30].

The reduction embeds the discrete logarithm challenge in either $\mathbf{y}$ or all the $\mathbf{z}_j$ and $\mathbf{z}_{1,j}$ by programming the random oracle H_1 and H_2. I.e. on input of a discrete logarithm challenge $\mathbf{U}$, the reduction sets either $\mathbf{y} = \mathbf{U}$ and generates $\mathbf{z}_j, \mathbf{z}_{1,j}, \mathbf{h}$ with known discrete logarithms to base $\mathbf{g}^{v_i}$ for randomly chosen $v_i \xleftarrow{\$} \mathbb{Z}_q$, or the reduction sets $\mathbf{y} = \mathbf{g}^x$ for known $x \xleftarrow{\$} \mathbb{Z}_q$, $\mathbf{h} := \mathbf{g}^v$ for a known $v \in \mathbb{Z}_q$, and generates all $\mathbf{z}_j, \mathbf{z}_{1,j}$ as $\mathbf{U}^{v_i}$ for $v_i \xleftarrow{\$} \mathbb{Z}_q$. This allows the reduction to either generate signatures using its knowledge of the discrete logarithms of $\mathbf{z}_j, \mathbf{z}_{1,j}$, and $\mathbf{h}$, or its knowledge of the secret key x. Due to lemma 2 we can assume that there is one session that produces two signatures. As the responses for H_2 have been programmed, this session can be identified and a representation of all group elements to $\mathbf{g}, \mathbf{y}, \mathbf{z}_j, \mathbf{z}_{1,j}$ is known to the reduction. Similar to the proof of lemma 2 the algebraic representations of the group elements submitted in hash queries to H_3 can be used to compute preliminary ω' and $\delta', \delta'', \delta'''$ for both of the special signatures belonging to the same session. As at least one of the

signatures was not created through a run of the honest signing protocol, using similar arguments as for the special signature in lemma 2, thus the witness can be computed by the reduction which yields the statement. □

4 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. A figure depicting an interaction can be found in the full version [30].[8] Let $H\colon \{0,1\}^* \to \mathbb{Z}_q$ be a hash function.

- KeyGen: On input pp, KeyGen samples $x \xleftarrow{\$} \mathbb{Z}_q$ and sets $\mathbf{x} \coloneqq \mathbf{g}^x$. It sets $\mathsf{sk} \coloneqq x, \mathsf{pk} \coloneqq \mathbf{x}$ and returns $(\mathsf{sk}, \mathsf{pk})$.
- Sign_1: On input sk, Sign_1 samples $r \xleftarrow{\$} \mathbb{Z}_q$ and returns the commitment $\mathbf{r} \coloneqq \mathbf{g}^r$ and the state $St_S \coloneqq r$.
- Sign_2: On input a secret key sk, a state $St_S = r$ and a challenge c, Sign_2 computes $s \coloneqq c \cdot \mathsf{sk} + r \mod q$ 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 $\mathbf{r}' \coloneqq \mathbf{r} \cdot \mathbf{g}^\alpha \cdot \mathsf{pk}^\beta$ and $c' \coloneqq H(\mathbf{r}', m), c \coloneqq c' + \beta \mod q$. It returns the challenge c and the state $St_U \coloneqq (\mathbf{r}, c, \alpha, \beta, m)$.
- 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 $\mathbf{r}' \coloneqq \mathbf{r} \cdot \mathbf{g}^\alpha \cdot \mathsf{pk}^\beta$ and $s' \coloneqq s + \alpha$ and returns the signature $\sigma \coloneqq (\mathbf{r}', s')$.
- Verify: On input a public key pk, a signature $\sigma = (\mathbf{r}', s')$ and a message m, Verify computes $c' \coloneqq H(\mathbf{r}', m)$ and checks whether $g^{s'} = \mathbf{r}' \cdot \mathsf{pk}^{c'}$. If so, it returns 1; otherwise, it returns 0.

Theorem 2. *Let* M *be an algebraic adversary that runs in time* t_{M}*, makes at most* ℓ *queries to* $\mathbf{Sign}_2$ *in* ℓ-$\mathbf{SEQ\text{-}OMUF}_{\mathsf{BSS}}$*, and at most* q_h *random oracle queries to* H*. Then there exists an adversary* B *such that*

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

and B *runs in time* $t_{\mathsf{B}} = t_{\mathsf{M}} + O(\ell + q_h)$*.*

Proof. Let M be as in the theorem statement. Without loss of generality, we assume that M makes exactly $\ell + 1$ many $\mathbf{Sign}_1()$ and exactly ℓ many $\mathbf{Sign}_2$ queries, and returns exactly $\ell + 1$ valid signatures $(\mathbf{r}_1^*, s_1^*), \ldots, (\mathbf{r}_{\ell+1}^*, s_{\ell+1}^*)$ of

[8] 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.

messages $m_1^*, \ldots, m_{\ell+1}^*$.[9] 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 ℓ-**SEQ-OMUF**$_{\mathsf{BSS}}$ as we prove in the following simple claim.

Claim 7. The pairs $(m_i^*, \mathbf{r}_i^*), \ldots, (m_j^*, \mathbf{r}_j^*)$ are pairwise distinct for all $i, j \in [\ell + 1]$.

Proof. Suppose $(m_i^*, \mathbf{r}_i^*) = (m_j^*, \mathbf{r}_j^*)$ 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 $(\mathbf{r}_i^*, s_i^*)$ and $(\mathbf{r}_i^*, s_j^*)$ are both valid signatures of m_i^*, since given m_i^* and $\mathbf{r}_i^*$, s_i^* as in the valid signature is uniquely defined (as in Eq. (**??**)). □

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 ℓ-SEQ-OMUF game. We have that

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

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. (If M has not seen a certain $\mathbf{r}_j$ when outputting $\mathbf{r}_i^*$, then the game naturally sets $\rho_{i,j}^* = 0$, as M is not allowed to use $\mathbf{r}_j$ as a base.) 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 8. $\Pr[E] \leq \frac{q_h^2 + q_h + 2}{2q}$

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 in $\mathbb{Z}_q$ (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

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

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 (where $j \in [j_0]$) 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, $\Pr[\chi_i = c_i^* + \xi_i^* - \sum_{j=1}^{j_0} \rho_{i,j}^* c_j = 0] = \frac{1}{q}$, 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 $(i+1)$-st 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 (where $j \in [j_0-1]$) 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 with unknown c_{j_0}

$$\begin{cases} \chi_i = c_i^* + \xi_i^* - \sum_{j=1}^{j_0} \rho_{i,j}^* c_j = 0, \\ \chi_{i+1} = c_{i+1}^* + \xi_{i+1}^* - \sum_{j=1}^{j_0} \rho_{i+1,j}^* c_j = 0. \end{cases} \tag{8}$$

Denote $A := \begin{pmatrix} \rho^*_{i,j_0} & c^*_i + \xi^*_i - \sum_{j=1}^{j_0-1} \rho^*_{i,j} c_j \\ \rho^*_{i+1,j_0} & c^*_{i+1} + \xi^*_{i+1} - \sum_{j=1}^{j_0-1} \rho^*_{i+1,j} c_j \end{pmatrix}$ and $B := \begin{pmatrix} \rho^*_{i,j_0} \\ \rho^*_{i+1,j_0} \end{pmatrix}$ the augmented matrix and coefficient matrix, respectively, of (8). We first note that if $\rho^*_{i,j_0} = \rho^*_{i+1,j_0} = 0$ all factors in eq. (8) are fixed when M makes his query. Thus, the probability that $\chi_i = \chi_{i+1} = 0$ is at most $\frac{1}{q}$ over the choice of c^*_i and c^*_{i+1}. In the following we assume that $\rho^*_{i,j_0} \neq 0$ or $\rho^*_{i+1,j_0} \neq 0$. Then

$$\begin{aligned}
&\Pr[\chi_i = \chi_{i+1} = 0] = \Pr[c_{j_0} \text{ is the solution of (8)}] \leq \Pr[\text{(8) has a solution}] \\
&= \Pr\left[\operatorname{rank}(A) = \operatorname{rank}(B)\right] \leq \Pr\left[\operatorname{rank}(A) \leq 1\right] = \Pr\left[\det(A) = 0\right] \\
&= \Pr\begin{bmatrix} \rho^*_{i,j_0} c^*_{i+1} + \rho^*_{i,j_0}(\xi^*_{i+1} - \sum_{j=1}^{j_0-1} \rho^*_{i+1,j} c_j) \\ -\rho^*_{i+1,j_0}(c^*_i + \xi^*_i - \sum_{j=1}^{j_0-1} \rho^*_{i,j} c_j) = 0 \end{bmatrix} = \frac{1}{q},
\end{aligned}$$

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 true 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 ℓ-OMDL. We now upper bound $\mathsf{Adv}^{\mathbf{G_1}}_{\mathsf{M}}$ via a reduction B from ℓ-**OMDL**. B runs on input $(\mathbb{G}, \mathbf{g}, q)$, and is given oracle access to **chal** and **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 $\mathbf{r}_{\ell+1} := \mathbf{g}^{r_{\ell+1}}$.) 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 ℓ-**OMDL** if M wins $\mathbf{G_1}$. Since M is algebraic, we have that

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

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

$$\mathbf{x}^{c_i^* + \xi_i^*} \cdot \prod_{j=1}^{\ell} \mathbf{r}_j^{\rho_{i,j}^*} = \mathbf{g}^{s_i^* - \gamma_i^* - \rho_{i,\ell+1}^* r_{\ell+1}}. \tag{9}$$

By definition of s_j, we have that

$$\mathbf{r}_j = \frac{\mathbf{g}^{s_j}}{\mathbf{x}^{c_j}}, \tag{10}$$

substituting (10) into (9), 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 (10) again, $r_j = \operatorname{dlog} \mathbf{r}_j$. This means that B wins ℓ-**OMDL**. We have that

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

We conclude that

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

completing the proof. □

4.1 Optimality of Our Reduction

In this section, we show an impossibility result which states (roughly) that reducing ℓ-sequential one-more unforgeability of Schnorr's blind signature scheme from ℓ-**OMDL** (as shown in section 4) is the best one can hope for. Concretely, we show that any algebraic reduction B that solves $(\ell-1)$-**OMDL** when provided with black-box access to a successful algebraic forger A in ℓ-$\mathbf{SEQ\text{-}OMUF}_{\mathsf{BSS}}$, can be turned into an efficient adversary M against $(\ell-1)$-**OMDL**.

Algebraic Black Boxes. We consider a type of algebraic adversary that, apart from providing algebraic representations for each of its output group elements to the reduction, does not provide any further access (beyond black-box access). In particular, the reduction does not get access to the code of the adversary. This notion was previously put forth and used by Bauer et al. [8].

Theorem 3. [10] *Let* B *be an algebraic reduction that satisfies the following: if algorithm* A *is an algebraic black-box algorithm that runs in time* t_{A} *then*

$$\mathsf{Adv}^{\mathbf{OMDL}}_{\mathsf{B},\ell-1} = \epsilon_{\mathsf{B}}\left(\mathsf{Adv}^{\mathbf{SEQ\text{-}OMUF}}_{\mathsf{A},\ell,\mathsf{BSS}}\right)$$

and B *runs in time* $t_{\mathsf{B}}(t_{\mathsf{A}})$*. (Here,* ϵ_{B} *and* t_{B} *are functions in the success probability and running time of* A*). Then there exists an algorithm* M *(the meta-reduction) such that*

$$\mathsf{Adv}^{\mathbf{OMDL}}_{\mathsf{M},\ell-1} \geq \epsilon_{\mathsf{B}}\left(\left(1-\frac{1}{q}\right)^{\ell}\right)$$

and M *runs in time* $t_{\mathsf{M}} = t_{\mathsf{B}}(O(\ell^3))$*.*

Proof Idea. We give a brief overview of the proof here, the detailed proof can be found in the full version [30]. We employ the meta-reduction technique [18]. Our meta-reduction provides the reduction with interfaces from the one-more discrete logarithm game as well as an algebraic black box forger for blind Schnorr signatures. It plays the OMDL game itself and forwards all oracle queries and responses, thereby providing the reduction with the interfaces of an OMDL challenger. The meta-reduction (in the role of the forger) first opens and closes all signing sessions before it makes its first hash query. We note that up to this point the only outputs made by the meta-reduction in the role of the forger have been uniformly random queries to the $\mathbf{Sign}_2$ oracle provided by the reduction, and thus independent of the algebraic representations output by the meta-reduction during the process. It then uses the algebraic representations output by the reduction as well as the responses from $\mathbf{Sign}_2$ to compute the secret key through means of linear algebra. The meta-reduction then starts making queries to the random oracle provided by the reduction and generating signatures, providing the discrete logarithm of its random commitments as a representation. Thus, all representations as well as all queries made by the reduction are independent from the algebraic representations that the reduction provides to the meta-reduction but not a to a real adversary. When the meta-reduction has output its signatures to the reduction, the reduction solves the OMDL challenge. The meta-reduction at this point only forwards the solution to its own OMDL challenger and wins whenever the reduction wins.

Doesn't this also contradict Section 3? One may ask if it is possible to apply a similar meta-reduction technique to Abe's blind signature scheme or our partially blind variant, which would contradict our result from Section 3. However, this is not possible as the algebraic representations output by the reduction break the witness-indistinguishability of the scheme. The meta-reduction would only be able to compute the witness used by the reduction. Thus, the combination of representations provided by the adversary and signatures provided by the adversary would be dependent on the algebraic representations provided by the reduction.

[10] This theorem even holds for a weaker version of ℓ-$\mathbf{SEQ\text{-}OMUF}_{\mathsf{BSS}}$ where the adversary A is required to output signatures for $\ell+1$ *distinct* messages.

References

1. Abe, M.: A secure three-move blind signature scheme for polynomially many signatures. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 136–151. Springer, Heidelberg (May 2001)
2. Abe, M., Fujisaki, E.: How to date blind signatures. In: Kim, K., Matsumoto, T. (eds.) ASIACRYPT'96. LNCS, vol. 1163, pp. 244–251. Springer, Heidelberg (Nov 1996)
3. Abe, M., Okamoto, T.: Provably secure partially blind signatures. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 271–286. Springer, Heidelberg (Aug 2000)
4. Agrikola, T., Hofheinz, D., Kastner, J.: On instantiating the algebraic group model from falsifiable assumptions. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 96–126. Springer, Heidelberg (May 2020)
5. Alper, H.K., Burdges, J.: Two-round trip schnorr multi-signatures via delinearized witnesses. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 157–188. Springer, Heidelberg, Virtual Event (Aug 2021)
6. Baldimtsi, F., Lysyanskaya, A.: Anonymous credentials light. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013. pp. 1087–1098. ACM Press (Nov 2013)
7. Baldimtsi, F., Lysyanskaya, A.: On the security of one-witness blind signature schemes. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 82–99. Springer, Heidelberg (Dec 2013)
8. Bauer, B., Fuchsbauer, G., Loss, J.: A classification of computational assumptions in the algebraic group model. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 121–151. Springer, Heidelberg (Aug 2020)
9. Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. Journal of Cryptology 16(3), 185–215 (Jun 2003)
10. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 93. pp. 62–73. ACM Press (Nov 1993)
11. Bellare, M., Rogaway, P.: Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331 (2004), `https://eprint.iacr.org/2004/331`
12. Benhamouda, F., Lepoint, T., Loss, J., Orrù, M., Raykova, M.: On the (in)security of ROS. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 33–53. Springer, Heidelberg (Oct 2021)
13. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In: Desmedt, Y. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (Jan 2003)
14. Boneh, D., Venkatesan, R.: Breaking RSA may not be equivalent to factoring. In: Nyberg, K. (ed.) EUROCRYPT'98. LNCS, vol. 1403, pp. 59–71. Springer, Heidelberg (May / Jun 1998)
15. Bouaziz-Ermann, S., Canard, S., Eberhart, G., Kaim, G., Roux-Langlois, A., Traoré, J.: Lattice-based (partially) blind signature without restart. Cryptology ePrint Archive, Report 2020/260 (2020), `https://eprint.iacr.org/2020/260`
16. Brands, S.: Untraceable off-line cash in wallets with observers (extended abstract). In: Stinson, D.R. (ed.) CRYPTO'93. LNCS, vol. 773, pp. 302–318. Springer, Heidelberg (Aug 1994)

17. Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) CRYPTO'82. pp. 199–203. Plenum Press, New York, USA (1982)
18. Coron, J.S.: Optimal security proofs for PSS and other signature schemes. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 272–287. Springer, Heidelberg (Apr / May 2002)
19. Drijvers, M., Edalatnejad, K., Ford, B., Kiltz, E., Loss, J., Neven, G., Stepanovs, I.: On the security of two-round multi-signatures. In: 2019 IEEE Symposium on Security and Privacy. pp. 1084–1101. IEEE Computer Society Press (May 2019)
20. Fischlin, M.: Round-optimal composable blind signatures in the common reference string model. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 60–77. Springer, Heidelberg (Aug 2006)
21. Fuchsbauer, G., Hanser, C., Slamanig, D.: Practical round-optimal blind signatures in the standard model. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 233–253. Springer, Heidelberg (Aug 2015)
22. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62. Springer, Heidelberg (Aug 2018)
23. Fuchsbauer, G., Plouviez, A., Seurin, Y.: Blind schnorr signatures and signed ElGamal encryption in the algebraic group model. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 63–95. Springer, Heidelberg (May 2020)
24. Garg, S., Gupta, D.: Efficient round optimal blind signatures. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 477–495. Springer, Heidelberg (May 2014)
25. Garg, S., Rao, V., Sahai, A., Schröder, D., Unruh, D.: Round optimal blind signatures. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 630–648. Springer, Heidelberg (Aug 2011)
26. Ghoshal, A., Tessaro, S.: Tight state-restoration soundness in the algebraic group model. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part III. LNCS, vol. 12827, pp. 64–93. Springer, Heidelberg, Virtual Event (Aug 2021)
27. Hauck, E., Kiltz, E., Loss, J.: A modular treatment of blind signatures from identification schemes. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part III. LNCS, vol. 11478, pp. 345–375. Springer, Heidelberg (May 2019)
28. Hauck, E., Kiltz, E., Loss, J., Nguyen, N.K.: Lattice-based blind signatures, revisited. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 500–529. Springer, Heidelberg (Aug 2020)
29. Juels, A., Luby, M., Ostrovsky, R.: Security of blind digital signatures (extended abstract). In: Kaliski Jr., B.S. (ed.) CRYPTO'97. LNCS, vol. 1294, pp. 150–164. Springer, Heidelberg (Aug 1997)
30. Kastner, J., Loss, J., Xu, J.: On pairing-free blind signature schemes in the algebraic group model. Cryptology ePrint Archive, Report 2020/1071 (2020), `https://eprint.iacr.org/2020/1071`
31. Katz, J., Loss, J., Rosenberg, M.: Boosting the security of blind signature schemes. ASIACRYPT 2021, to appear (2021), `https://eprint.iacr.org/2021/806`
32. Nick, J., Ruffing, T., Seurin, Y.: MuSig2: Simple two-round Schnorr multi-signatures. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 189–221. Springer, Heidelberg, Virtual Event (Aug 2021)
33. Nicolosi, A., Krohn, M.N., Dodis, Y., Mazières, D.: Proactive two-party signatures for user authentication. In: NDSS 2003. The Internet Society (Feb 2003)

34. Ohkubo, M., Abe, M.: Security of some three-move blind signature schemes reconsidered. The 2003 Symposium on Cryptography and Information Security (2003)
35. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) CRYPTO'92. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (Aug 1993)
36. Okamoto, T.: Efficient blind and partially blind signatures without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 80–99. Springer, Heidelberg (Mar 2006)
37. Paillier, P., Vergnaud, D.: Discrete-log-based signatures may not be equivalent to discrete log. In: Roy, B.K. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 1–20. Springer, Heidelberg (Dec 2005)
38. Papachristoudis, D., Hristu-Varsakelis, D., Baldimtsi, F., Stephanides, G.: Leakage-resilient lattice-based partially blind signatures. IET Information Security 13(6), 670–684 (2019)
39. Pointcheval, D.: Strengthened security for blind signatures. In: Nyberg, K. (ed.) EUROCRYPT'98. LNCS, vol. 1403, pp. 391–405. Springer, Heidelberg (May / Jun 1998)
40. Pointcheval, D., Stern, J.: Provably secure blind signature schemes. In: Kim, K., Matsumoto, T. (eds.) ASIACRYPT'96. LNCS, vol. 1163, pp. 252–265. Springer, Heidelberg (Nov 1996)
41. Pointcheval, D., Stern, J.: New blind signatures equivalent to factorization (extended abstract). In: Graveman, R., Janson, P.A., Neuman, C., Gong, L. (eds.) ACM CCS 97. pp. 92–99. ACM Press (Apr 1997)
42. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. Journal of Cryptology 13(3), 361–396 (Jun 2000)
43. Rotem, L., Segev, G.: Algebraic distinguishers: From discrete logarithms to decisional uber assumptions. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part III. LNCS, vol. 12552, pp. 366–389. Springer, Heidelberg (Nov 2020)
44. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (Aug 1990)
45. Schnorr, C.P.: Security of blind discrete log signatures against interactive attacks. In: Qing, S., Okamoto, T., Zhou, J. (eds.) ICICS 01. LNCS, vol. 2229, pp. 1–12. Springer, Heidelberg (Nov 2001)
46. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004), `https://eprint.iacr.org/2004/332`
47. Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 288–303. Springer, Heidelberg (Aug 2002)