# Stronger Security for Blind Signatures with Attributes

Foteini Baldimtsi[1], Julia Kastner[2], Julian Loss[3], and Omar Renawi[4]

[1] George Mason University and Mysten Labs
Fairfax, VA, USA
foteini@gmu.edu
[2] Centrum Wiskunde & Informatica
Amsterdam, Netherlands
julia.kastner@cwi.nl
[3] Ruhr University Bochum
Bochum, Germany
julian.loss@rub.de
[4] CISPA Helmholtz Center for Information Security and Saarland University
Saarbrucken, Germany
omar.renawi@cispa.de

**Abstract.** Single-use anonymous credentials are issued once and allow users to unlinkably authenticate exactly once. Recent anonymous-token and age-verification systems increasingly rely on such credentials to minimize credential reuse and simplify revocation. These systems are typically instantiated from blind signatures, yielding lightweight credentials whose presentation reduces essentially to signature verification. Many deployments additionally require embedded attributes and selective attribute disclosure. While this functionality can be achieved through Blind Signatures with Attributes (BSAs), only a handful of blind-signature constructions support attributes, and the security guarantees of selective disclosure are not yet fully understood or properly formalized.
We close this gap by first revisiting the security definitions for BSAs and introducing stronger notions capturing secure showing under partial attribute reveal. We then present a generic transformation that lifts a broad class of two- and three-move blind signatures into BSAs supporting embedded attributes and selective disclosure. Our construction uses Merkle commitments and cut-and-choose techniques to enforce attribute consistency and achieves concurrent one-more unforgeability, upgraded blindness, and the stronger secure-showing notion. Finally, we prove that the well-known Anonymous Credentials Light (ACL) construction satisfies these stronger notions without modification in the ROM+AGM.

## 1 Introduction

Blind signatures are a fundamental cryptographic primitive that allows a user $U$ to obtain a signature from a signer $S$ on a hidden message and later prove validity to a verifier $V$ without linking issuance and showing [23]. They are

especially useful for building privacy-preserving protocols and, in particular, for private user authentication. A blind signature can almost directly instantiate a lightweight Anonymous Credentials (AC) scheme, where each issued signature functions as a *single-show* token.

Single-show ACs (often called *anonymous tokens*) naturally arise in applications such as anonymous rate limiting, abuse prevention, couponing, and age verification, where credentials are intended to be used only once or only a small number of times. As a result, such single-show ACs have attracted significant attention in recent years, both from the research community [29,47,9,57,19,52,5,48] and industry (e.g., Cloudflare [50], Apple [4], Google [34], Facebook [31]), while they are also undergoing standardization through the IETF Privacy Pass working group [40]. Related privacy-preserving token-based approaches have also been considered in the context of European age-verification proposals [49].

Many deployments additionally require credentials to support embedded *attributes* and *selective disclosure*. For example, age-verification systems may need to support multiple age predicates (e.g., over 13, over 15, over 18), while other deployments may require attributes such as residency, jurisdiction, or eligibility classes to be selectively revealed at presentation time without linking back to issuance. This functionality can be achieved through Blind Signatures with Attributes (BSAs), an extension of blind signatures that allows embedding commitments to user attributes inside the blindly signed message and enforcing attribute consistency at showing. In this sense, BSAs naturally give rise to single-show ACs with attribute support and can be viewed as a generalization of blind-signature-based anonymous tokens.

Unfortunately, very few blind-signature constructions support attributes. The only notable example is Anonymous Credentials Light (ACL) [7], which allows efficient attribute embedding and selective disclosure and is therefore considered one of the most practical single-show AC constructions. However, ACL has so far only been proven secure in the Algebraic Group Model (AGM). Moreover, the definitional landscape around secure showing under selective attribute disclosure remains incomplete.

Single-show credentials are not the only approach to privacy-preserving authentication. Standard multi-show anonymous credentials and attribute-based anonymous credentials [23,16,53,13,17,15,10,8,20,28,11,55,36,14] are particularly useful in settings where users require long-lived reusable credentials and repeated anonymous authentication and are at the center of ongoing efforts to deploy privacy-preserving digital identity systems, including the European Digital Identity Wallet framework [24]. The choice between multi- and single-show credentials is inherently application-dependent. While reusable credentials are well-suited for long-lived identity systems, many deployments naturally require one-time or few-time authentication semantics and benefit from explicit limits on credential reuse. Reusable anonymous credentials also require revocation and misuse-mitigation mechanisms to manage long-lived credentials and repeated use.

A natural question is then how such single-show credentials should be realized. One possibility is to start from a conventional reusable credential and enforce single use during presentation, for example, by proving in zero knowledge both possession of a valid credential and correct derivation of a nullifier. This is expressive but moves additional work to the showing phase. In group-based constructions, credential presentation commonly relies on pairing-based verification or zero-knowledge proofs over the signature-verification relation, and when this relation itself involves a random oracle (RO), the latter additionally requires reasoning about a concrete instantiation of that oracle inside the proof system.

Blind signatures offer a particularly lightweight realization of single-show credentials: issuance and presentation are unlinkable, while showing essentially reduces to signature verification. The challenge is to add embedded attributes and selective disclosure without losing this simplicity. In particular, we seek pairing-free constructions that avoid zero-knowledge proofs about random-oracle computations and admit an end-to-end security proof in the same idealized model as the underlying blind signature.

Despite the practical relevance of this setting, little is known about how to obtain such attribute-supporting single-show credentials generically from standard blind signatures and under standard assumptions. In this work, we investigate exactly this question. Concretely, we investigate whether such constructions can be obtained generically, rather than requiring new, scheme-specific constructions and proofs each time. We show the following:

- We present Pachirisu[5], a generic transform that transforms a class of blind signature schemes, BS, into a BSA, and thus a single-show AC with attribute support, in the random oracle model. Our transform applies to many popular constructions such as the Schnorr and BLS blind signature schemes. In this manner, we are able to obtain reasonably efficient single-show ACs with attribute support in the random oracle model from various standard assumptions including DLOG, RSA, factoring, the linear code equivalence problem, and GA-DLOG.
- As a contribution of independent interest, we revisit security notions from the literature related to the showing process of embedded attributes in BSAs. Existing notions consider only the oversimplified scenario in which *all* embedded attributes must be shown at presentation. As we discuss below, the correct way to formalize secure showing when only *a part of* the attribute vector is revealed is still unsettled. In this work, we give the first concise definitions for this important and natural scenario.
- In the final part of our work, we apply our new security notions to ACL and prove security without modifying the original scheme.

*On the need for new security definitions.* Existing notions for related primitives do not fully capture the single-show setting with selective disclosure. Standard

[5] Pachirisu is closely related to PI-Cut-Choo [18], but it stores attributes in (Merkle) trees.

anonymous-credential definitions target reusable credentials, where repeated presentations of the same credential are legitimate. They therefore do not express the accounting property required here: every accepted showing, including a partial showing, should be attributable to a distinct issuance.

Partially blind signatures (PBS) [2,3,43] are also not an adequate substitute. In a PBS, the public information is fixed and agreed upon during issuance, whereas in a BSA the attributes to be disclosed may only be chosen at showing time. Treating an attribute commitment as the public information of a PBS therefore creates a tension with unlinkability: reusing the same commitment makes different showings linkable, while using a fresh randomized commitment prevents it from serving as the fixed public information associated with the issued signature.

More importantly, prior BSA definitions leave a concrete gap exactly at the intersection of single use and selective disclosure. Although the BSA syntax allows users to reveal only part of their attribute vector, existing secure-showing definitions consider only full attribute vectors. They therefore do not specify what it means for multiple accepted partial showings to be consistent with the credentials that were actually issued. We close this gap by requiring that every set of accepted partial showings admit an injective matching to distinct issuance sessions with compatible attribute vectors. This captures both attribute consistency and the single-show accounting property.

*Toward practical BSAs.* We view Pachirisu as a *baseline feasibility* result toward obtaining attribute-friendly single-show ACs from standard assumptions with reasonable efficiency. Our work establishes an interesting modular link between blind signatures and blind signatures with attributes (BSAs) by adapting the well-known 'boosting' technique of Katz et al. [46] and Chairattana-Apirom et al. [18] to the attribute setting. Although concerted community efforts have improved on the efficiency of these early works significantly, we stress that schemes obtained via boosting were considered to be at the forefront of efficient blind signature constructions from standard assumptions as recently as three years ago at the time of writing. We believe that, analogous to how the blind-signature landscape has evolved through successive optimizations on these and other techniques, our work will serve as a starting point for a more systematic effort toward practical BSAs, and thus attribute-based ACs, from standard assumptions.

## 2 Technical Overview

At a high level, our goal is to augment plain blind signatures with commitments to user attributes and selective disclosure while preserving the lightweight presentation semantics of single-show credentials. To this end, we present Pachirisu, a generic transformation that lifts blind signature schemes into blind signatures with attributes (BSAs). By $\mathbf{P}[\mathsf{BS}]$ we denote the BSA obtained from an underlying blind signature scheme BS.

### 2.1 Our Transformation

We introduce Pachirisu (denoted $\mathbf{P}[\cdot]$), a transformation that lifts a *key-only-secure two- or three-move* blind signature scheme BS into a blind signature with attributes (BSA). Here, key-only security refers to unforgeability against adversaries that only obtain the public verification key but never interact with the signing protocol or see valid signatures. The resulting scheme $\mathbf{P}[\mathsf{BS}]$ achieves selective secure showing as well as one-more unforgeability with a polynomial number of concurrent signing sessions.

Conceptually, the construction can be viewed as a variant of *blind-signature boosting*: starting from a plain, key-only-secure blind signature scheme, we obtain a blind signature with attributes supporting selective disclosure while preserving blindness and strengthening unforgeability.

A crucial feature of our construction is ensuring that the signer actually issues signatures with respect to the "right" attributes. A natural idea would be to require the user to provide a non-interactive zero-knowledge (NIZK) proof that the committed attributes were encoded honestly. However, because many efficient blind signatures (including most of the schemes we target) are proven secure in the Random Oracle Model (ROM), plugging in a NIZK is delicate. In the ROM, a NIZK proof of knowledge requires proving knowledge of the *random-oracle output* itself, creating circular dependencies where the simulator must both program the oracle and extract from proofs that depend on that programming. This interaction breaks standard simulation strategies and is incompatible with the way ROM-based blind signatures are analyzed.

**Boosting.** To avoid these issues, we instead rely on the (parallel-instance) cut-and-choose strategy first introduced in [46] and further developed in [18]. These transforms were originally designed to strengthen blind signature schemes so that they achieve one-more unforgeability under a larger number of concurrent signing sessions. Importantly, they already address many of the ROM-related challenges described above, making them an ideal starting point for $\mathbf{P}[\cdot]$.

We specifically build on the variant by [18], which only requires key-only security of the underlying scheme. We briefly recall the transform: The signer generates $K$ public keys of the underlying (key-only secure) blind signature scheme. To sign a message, the signer and user run $N$ parallel sessions for each of the public keys (where $N$ is a number that is chosen depending on how many signing sessions have already been opened with the signer). To ensure honest behavior by the user, the user has to commit to all of their randomness at the beginning of each session. Later on, before receiving the last message from the signer, the user is asked to open all but one (random) commitment among the parallel sessions corresponding to each of the $K$ public keys. This allows the user to show that the first user messages were computed honestly. Since the commitments were chosen at random, it is unlikely that the user gets away with cheating (i.e., using randomness different from that in the commitment) in *all* of the unopened sessions. In fact, through a simple concentration argument, it is possible to show that for one of the public keys pk, the adversary will always have to use pre-committed

randomness in order to close the final part of the signing protocol. This allows a reduction from the key-only security of the scheme corresponding to $\mathsf{pk}$. Blindness is preserved because the signer only closes the session for which it did not ask to open the commitment. The final signature consists of $K$ subsignatures $(\sigma_1, \ldots, \sigma_K)$.

**Adding Attributes.** As the focus of our work is building blind signatures with attributes for single-show anonymous credentials, it is important that we also achieve strong one-more unforgeability, so that the same attribute vector cannot be shown more often than it was signed. It turns out that this is not achieved by the original parallel-instance cut-and-choose transform. Namely, for two signature tuples $(\sigma_1, \ldots, \sigma_K)$ and $(\sigma'_1, \ldots, \sigma'_K)$ on the same message $M$, $(\sigma'_1, \sigma_2, \ldots, \sigma_K)$ is also a valid signature on $M$.

Our main insight is that one can modify the construction of [18] so that the message for subsignature $\sigma_i$ contains the subsignature $\sigma_{i-1}$. We therefore run the cut-and-choose sessions for the different public keys sequentially. As a result, a strong forgery (i.e., two signatures on the same message) can be translated into a forgery on two different messages with respect to the first index where the strong forgeries begin to diverge. Some additional properties (see below) of the underlying scheme $\mathsf{BS}$ are needed to prevent a strong forgery in which only the outermost signature indices differ.

This explanation, of course, sweeps some technical details under the rug. First, to achieve blindness, the user needs to use a commitment to their message as the message for the underlying scheme in each signing session, so that when they reveal their randomness and the computation of the first user message, they do not reveal the underlying message that they are trying to get signed. On the other hand, the user has to provide a commitment to their attributes inside the message. We achieve this using Merkle-tree commitments with fresh randomness for each public key and each parallel session. Merkle-tree commitments can be opened partially, which comes in handy when proving our new, stronger secure-showing property for our construction.

**Compatible Schemes.** We identify a class of schemes that are compatible with our construction. We dub this class *three-move simulatable blind signature schemes*. These schemes have the following (simplified) properties:

- They have three or fewer moves. If they have two moves, we consider them to have three moves but with an empty first message from the signer.
- During issuance, the user makes one random oracle query that is repeated during verification of the resulting signature. This query is linked[6] to the signing session, user randomness, and the message to be signed.
- There exists a signer simulation algorithm that produces a simulated first signer message and a state. Using the state, the message to be signed, and

[6] The exact meaning of this will be specified in more detail in section 4.

the user's random coins, random oracle responses can be simulated, and there exists a simulator algorithm that uses this information to produce a matching second signer message.

As key-only unforgeability is sufficient for our security reductions, we can use a wide range of schemes captured by the above definition without having to worry about ROS-style attacks. Candidate schemes include many Schnorr-style blind signatures, for example, Schnorr [56] (from DLOG), Abe-Okamoto [3] (from DLOG), Abe's blind signature [1] (from DLOG), Tessaro-Zhu [58] (from DLOG), Okamoto-Schnorr [51] (from DLOG), CSI-Otter [45] (from GA-DLOG instantiated with CSI), Tanuki [35] (from GA-DLOG instantiated with the CSI or LESS group action), and the schemes based on linear function families described in [37] (from DLOG, factoring, or RSA). Our transform can also be used as in [18] for two-move schemes such as blind BLS [12] (from CDH in pairing groups) or blind RSA [22] (from RSA) by setting the first signer message to be an empty string and making only very minor modifications to the security proofs.

### 2.2 The Right Security Framework for BSAs

The first security definition for BSAs was given in [7], where one-more unforgeability and secure showing of attributes were covered by one definition. In a recent work, Kastner et al. [41] provide a new, modular security definition that separates the security of the showing procedure from one-more unforgeability. In doing so, however, their work also uncovers a subtlety with the original secure-showing definition of Baldimtsi and Lysyanskaya. In their security experiment, the adversary interacts with the signer and obtains a set of $\ell$ signatures $\sigma_1, \ldots, \sigma_\ell$ with corresponding attribute vectors $\boldsymbol{L}'_1, \ldots, \boldsymbol{L}'_\ell$. The adversary is considered successful in breaking the secure-showing property if, at the end of the experiment, it can produce an alternative multiset of vectors $\boldsymbol{L}_1, \ldots, \boldsymbol{L}_\ell$ that differs from the original one in at least one component, yet still verifies against $\sigma_1, \ldots, \sigma_\ell$. (We consider multisets, as the same attribute vector may appear multiple times among $\boldsymbol{L}'_1, \ldots, \boldsymbol{L}'_\ell$.)

While this intuitively ensures that the adversary cannot alter their attributes post facto, closer scrutiny reveals that this definition does not cover the natural scenario in which the user only *does a selective opening of their attributes* in the showing procedure (partial reveal). Although the option of partial attribute reveal is provided by the syntax of the showing procedure, the security experiment corresponding to the showing definition only allows for the full set of attributes to be shown!

**Secure Showing with Selective Reveal** In Section 6.1, we formalize a stronger secure showing definition for blind signatures with attributes that remedies the aforementioned shortcomings by supporting selective attribute reveal. We note that our new definition is the natural one needed to use single-use BSAs in applications (since most of them require partial showing for attributes).

Our new definition, and later the proofs that our generic transform and ACL satisfy it, come with several unexpected subtleties both at a definitional and technical level. In our newly proposed notion, the adversary can again query the challenger of the security experiment for signatures corresponding to attribute vectors $\boldsymbol{L}'_1, \ldots, \boldsymbol{L}'_k$. Similar to the original definition, the adversary has to provide, at the end of the experiment, a set of vectors $\boldsymbol{L}_1, \ldots, \boldsymbol{L}_\ell$ that each verify against at least one of the signatures $\sigma_i$. However, different from before, $\boldsymbol{L}_i$ can also be *partial* vectors; that is, some entries can be replaced by a special empty symbol $\perp$ (for honest users, these partial vectors would be subvectors of the previously signed vectors $\boldsymbol{L}'_i$, i.e., they would agree on the non-$\perp$ entries).

**A New Winning Condition.** At the heart of our modified experiment lies a new winning condition that takes into account the above discussion. Intuitively, this condition has to check whether the set of opened partial vectors cannot be explained by a set of honest users opening their signatures. For such an honest set of users, there would be an explanation for each of the opened vectors $\boldsymbol{L}_i$ by a signing session with a vector $\boldsymbol{L}'_j$ such that $\boldsymbol{L}_i$ is a subvector of $\boldsymbol{L}'_j$, and no signing session is used to explain more than one opened signature. More formally, the game checks if there is an *injective* mapping from the opened (partial) vectors $\boldsymbol{L}_1, \ldots, \boldsymbol{L}_\ell$ to the signed vectors $\boldsymbol{L}'_1, \ldots, \boldsymbol{L}'_k$ (multiply signed vectors appear multiple times) such that the opened vectors are subvectors of the corresponding signed vectors. If no such mapping exists, the adversary has won the game, as they must have altered at least one of the signed vectors post facto. Alternatively, this means that for any attempt at explaining the opened partial vectors through the signatures issued, there is at least one signature with opened vector $\boldsymbol{L}_i$ that cannot be explained by a signing session.

For example, consider an adversary that queried the challenger for one signature corresponding to $\boldsymbol{L}'_1 = (a, b)$ and two signatures corresponding to $\boldsymbol{L}'_2 = \boldsymbol{L}'_3 = (a, c)$. If they output three signatures $\sigma_1, \sigma_2, \sigma_3$ with (verifying) corresponding partial openings to attributes $(a, c), (\perp, c), (\perp, c)$, they win this game. Namely, since both $(a, c)$ and $(\perp, c)$ are subvectors only of $\boldsymbol{L}'_2 = \boldsymbol{L}'_3$, but not $\boldsymbol{L}'_1$, it is impossible to assign the three openings as subvectors injectively to the (multi-)set of previously signed attribute vectors $\{\boldsymbol{L}'_1, \boldsymbol{L}'_2, \boldsymbol{L}'_3\}$. On the other hand, an adversary that opens the signatures to $(a, \perp), (\perp, c), (\perp, c)$ would not win the game, as the $(a, \perp)$ now constitutes a subvector of any of $\boldsymbol{L}'_1, \boldsymbol{L}'_2$, and $\boldsymbol{L}'_3$, and the opened vectors can be explained by $(a, \perp) \mapsto \boldsymbol{L}'_1$, $(\perp, c) \mapsto \boldsymbol{L}'_2$, $(\perp, c) \mapsto \boldsymbol{L}'_3$.

At first glance, it may seem unclear whether this new secure showing definition is satisfiable *at all*, as it seemingly asks the challenger to check an exponentially complex set of constraints related to the winning condition of the security experiment. In Section 6.1, we explain how to overcome this obstacle by abstracting the secure showing experiment as a bipartite graph $G$ and using the Hopcroft-Karp algorithm [39] to find a maximum cardinality matching.

**Upgrading the Blindness Definition.** The standard blindness notion in the literature concerns *issuance* only: it hides which signing session produced which signature, but it says nothing about privacy *after showing*. If a user later reveals *all* attributes, they may be trivially identifiable when their attribute vector is unique, so pre-showing blindness is essentially the best guarantee in that case. In contrast, under *partial* reveal an honest user should still enjoy anonymity within the cohort of users who revealed the *same* attributes (e.g., same age or residency).

Motivated by this, we propose a natural extension to the blindness notion that explicitly includes registration and showing. It guarantees unlinkability between issuance sessions and showings among all presentations that reveal identical (fully or partially opened) attributes. The definition does not assume that attributes are hidden at registration, accommodating applications where some or all attributes must be disclosed to the issuer during enrollment.

### 2.3 Security Proofs for Our Transformation

**One-More Unforgeability.** The proof largely follows that of the CDH-based scheme in [18], but we use the properties of the underlying generic scheme instead of concrete scheme-specific simulation strategies.

The proof strategy is to provide a reduction that breaks key-only security of the underlying scheme $\mathsf{BS}$. The reduction is thus tasked with providing signatures for the underlying scheme $\mathsf{BS}$ without having access to a signing oracle. The property we use for this is that if given the random coins used to generate the user message, as well as the capability to program the random oracle at some points, the reduction can answer signing queries using the simulators. For Schnorr-style schemes, this is essentially the simulator of the underlying Fiat-Shamir proof, whereas for RSA and BLS, this is done by programming the random oracle with a point for which the reduction knows the signature. The reduction then needs to apply the appropriate blinding factors to its solution. Therefore, we need to make sure that there is at least one public key for which the reduction can always extract the user's randomness so that it knows all the blinding factors. As the adversary is tasked with committing to all of its randomness, we show that with a high probability, the adversary cannot "cheat", where by cheating we mean that there is a session where it manages to successfully reveal all of its randomness, but there is no matching commitment for the unrevealed sessions. This can be achieved through an appropriate choice of the number $K$ of keys and the number of cut-and-choose sessions $N$.

**Secure Showing** The proof of our secure showing property mirrors the one-more unforgeability proof. There are three ways a user can break secure showing:

- They "cheat" such that the signer sees attribute vector $\boldsymbol{L}$ revealed during signing, but the unrevealed sessions are using a different attribute vector $\boldsymbol{L}'$. Such cheating requires guessing all cut-and-choose indices correctly for that session. This can be ruled out using the same argument as before.

– They open a commitment to a different attribute (sub)vector. This can be ruled out using the binding property of the underlying Merkle-tree commitment scheme.
– They generate a novel signature without the help of the signer. This can be ruled out using key-only security like before.

**Blindness** For blindness, we use the underlying scheme's blindness to gradually swap which message is used during signing across sessions. To do this, we first decouple the opened messages from their commitments, as well as decouple the Merkle-tree commitments from the attribute vectors. We achieve this with standard random oracle techniques, since all commitments are computed via random oracles. This enables swapping messages in a hybrid argument based on the blindness of BS. For the opening phase, we are able to simulate the openings of the Merkle-tree-based attribute commitments using the random oracle.

### 2.4 An efficient scheme: ACL

Anonymous Credentials Light is the most efficient scheme for blind signatures with attributes known to date. It is a pairing-free scheme with lightweight verification (8 scalar multiplications, 5 curve additions, 1 field addition, and 1 hash operation) and a short signature size consisting of 8 elements. Implementing it on the BN254 elliptic curve yields a signature size of $2\ \mathbb{G} \times 64 + 6\ \mathbb{Z}_p \times 32 = 320$ bytes. Its security was only proven under the weaker blindness and secure showing definitions due to [41]. We investigate this scheme further and show that it fulfills our new secure showing definition, as well as the upgraded blindness definition. In Appendix G.5 , we also show how a simple modification can be used to detect double spending.

To prove secure showing, we rely on similar techniques in the Algebraic Group Model to those [41] used for OMUF. Recall that in ACL, the attributes are embedded as a Pedersen commitment in a component of the signature, and this same component forms a Diffie-Hellman tuple with components from the signer messages. Therefore, each signature has a (computationally hidden) link to at most one signing session, and this allows us to categorize attacks on secure showing in the following way:

– The adversary provides a signature that is not linked to a signing session. This can be ruled out using similar techniques to those for the OMUF proof provided in [41]; in particular, we prove a variant of what is often dubbed the "restrictive-blinding lemma" [1,7,44,41].
– The adversary provides two signatures linked to the same session. This can also be ruled out using similar techniques to those in [41] and essentially boils down to a 2-1-ROS problem, which is information-theoretically hard.
– Lastly, the adversary could open a commitment to different attributes. Using extraction techniques for the proof of knowledge of an opening in the AGM, we extract the full opening and show that this breaks the binding property of the underlying Pedersen commitment.

The blindness proof is very similar to that in [7], but it takes into account that the adversary may also request the user(s) to show some of their attributes. We simulate the showing of these attributes by simulating the proof of knowledge in the ROM.

For double-spending security, we provide a simple modification of the ACL scheme, which is based on a similar protocol proposed by Abe [1] for the underlying blind signature scheme. This same scheme was also implemented for ACL in [38], but without defining or proving its security. The key idea is to add the first flow of a $\Sigma$-protocol to be used in signature showing to the signature in such a way that it cannot be removed or exchanged, namely by appending it to the string that is hashed at signature generation time. Then, the showing protocol also involves proving knowledge of the user's attributes using this first flow contained in the signature. We implement this using the Fiat-Shamir (FS) heuristic. We note that both the user and the signer include a current timestamp to derive their FS challenge. This is important to ensure that if the user shows the same signature twice with different timestamps, the challenge from the hash function will be different. As the user must always use the same first flow contained in the signature, the two different transcripts reveal their identity.

## 3 Preliminaries and Definitions

*Notation.* For a natural number $n$, we denote by $[n]$ the set of numbers $\{1, \ldots, n\}$ and by $[n]_0$ the set $\{0\} \cup [n]$. Throughout the paper, we fix a group $\mathbb{G}$ of prime order $p$ with generator $g$ and assume that all algorithms have access to this group's parameters even when not given as an explicit input. We use $:=$ for deterministic assignments and $\leftarrow_\$$ to denote that a value is sampled at random from a set or the output of a randomized algorithm. More explicitly, we use the notation $b \leftarrow_\$ \mathcal{A}(a)$ to denote a randomized algorithm with input $a$ and output $b$. We write $\mathcal{A}^{\mathcal{O}(\cdot)}$ if $\mathcal{A}$ is equipped with oracle access to an algorithm $\mathcal{O}$.

Let $\boldsymbol{L}$ be a vector and $L_i$ be the $i$-th entry of $\boldsymbol{L}$. We say a vector $\boldsymbol{L}'$ is a *partial* vector if for some indices $i$, $L'_i = \perp$, where $\perp$ is a special empty symbol. For a vector $\boldsymbol{L}$ of length $n$ and a (partial) vector $\boldsymbol{L}'$, we say $\boldsymbol{L}'$ is a *subvector* of $\boldsymbol{L}$, denoted by $\boldsymbol{L}' \subseteq \boldsymbol{L}$, if for each index $i \in [n]$ it holds that either $L'_i = L_i$ or $L'_i = \perp$. A *multiset* is a generalization of a set where each element can appear multiple times. We call the number of times an element $x$ appears in a multiset the *multiplicity* of $x$. For multisets $M$ and $M'$, we say that $M'$ is a *subset* of $M$, denoted by $M' \subseteq M$, if each element in $M'$ is also contained in $M$, and the multiplicity of each element contained in both sets is less than or equal in $M'$ to its multiplicity in $M$.

In the following sections, we assume that there exist public parameters $\mathsf{pp}$ that are known to all algorithms even when not explicitly given as input.

### 3.1 Multi-index Vector Commitment

In this work, we use a commitment notion called Multi-index Vector Commitment abstracting the properties of Merkle-tree commitments that we need, de-

fined by algorithms $\mathsf{MiVC} \coloneqq (\mathcal{C}, \mathcal{O}, \mathcal{CV})$. $\mathcal{C}$ allows committing to a vector $\boldsymbol{L}$ using randomness $\boldsymbol{R}$. $\mathcal{O}$ allows partially opening the commitment at indices $\boldsymbol{I_{op}}$. $\mathcal{CV}$ allows verifying an opening of a commitment at the set of indices. The security properties are straightforward. We provide a formal definition in section A.5 .

**Merkle-Tree-Based Multi-index Vector Commitment**

**Construction 1.** We describe Multi-index Vector Commitment schemes $\mathsf{MT} \coloneqq (\mathcal{C}, \mathcal{O}, \mathcal{CV})$ that are based on Merkle trees. Let $\mathsf{H_{MT}} \colon \{0,1\}^* \to \{0,1\}^\lambda$ be a hash function treated as a random oracle.

- $\mathcal{C}(\boldsymbol{L}, \boldsymbol{R})$. Take as input a message vector $\boldsymbol{L} = (L_1, \ldots, L_\nu)$ and a randomness vector $\boldsymbol{R} = (R_1, \ldots, R_\nu)$. First, ensure that no $L_i = \perp$ and abort otherwise. Compute the sequence of leaves $S = (s_1, \ldots, s_\nu)$ where:

$$s_i \coloneqq \begin{cases} \mathsf{H_{MT}}(L_i, R_i) & \text{if } R_i \neq \perp \\ L_i & \text{otherwise} \end{cases}$$

  Then construct a binary Merkle tree over the sequence $S$ and output the root as the commitment $C$.
- $\mathcal{O}(C, \boldsymbol{L}, \boldsymbol{R}, \boldsymbol{I_{op}})$. Parse $(L_1, \ldots, L_\nu) \coloneqq \boldsymbol{L}$ and $(R_1, \ldots, R_\nu) \coloneqq \boldsymbol{R}$. For all $i \in \boldsymbol{I_{op}}$, let $R'_i \coloneqq R_i$ and $L'_i \coloneqq L_i$, and for all $j \in [\nu] \setminus \boldsymbol{I_{op}}$, let $R'_j \coloneqq \perp$ and $L'_j \coloneqq \mathsf{H_{MT}}(L_j, R_j)$. Let $\boldsymbol{R'} \coloneqq (R'_1, \ldots, R'_\nu)$ and $\boldsymbol{L'} \coloneqq (L'_1, \ldots, L'_\nu)$, and return $(\boldsymbol{L'}, \boldsymbol{R'})$.
- $\mathcal{CV}(C, \boldsymbol{L'}, \boldsymbol{R'}, \boldsymbol{I_{op}})$. Parse $(R'_1, \ldots, R'_\nu) \coloneqq \boldsymbol{R'}$ and output 0 if $\exists i \in \boldsymbol{I_{op}}$ with $R'_i = \perp$. Output 1 if $\mathcal{C}(\boldsymbol{L'}, \boldsymbol{R'}) = C$; otherwise, output 0.

Our scheme is correct(Lemma 5), hiding(Lemma 6), and binding(Lemma 7), as proven in section A.9 .

### 3.2 Blind Signature with Attributes

As a starting point, we use the definition presented in [41].

**Definition 1.** *A* blind signature with attributes *is a tuple of algorithms* $\mathsf{BSA} \coloneqq (\mathcal{G}, \mathcal{RG}, \mathsf{S}, \mathsf{U}, \mathcal{V}, \mathcal{SH})$ *with the following properties:*

- *The* key generation algorithm $\mathcal{G}$*: Take as input public parameters* $\mathsf{pp}$ *and output a pair of keys* $(\mathsf{pk}, \mathsf{sk})$ *for the signer. In the rest of the paper, we assume that* $\mathsf{pk}$ *implicitly contains a copy of* $\mathsf{pp}$ *and* $\mathsf{sk}$ *implicitly contains a copy of* $\mathsf{pk}$*.*
- *The* registration protocol $\mathcal{RG}$ *between the user and the signer consists of two algorithms,* $\mathcal{RG}_\mathsf{U}$ *and* $\mathcal{RG}_\mathsf{S}$*:*
  - *The* user's registration algorithm $\mathcal{RG}_\mathsf{U}$*: Take as input a vector of attributes* $\boldsymbol{L}$ *and output a commitment* $C$*, a user's registration state* $\mathsf{rst_U}$*, and a proof* $\pi$*. We assume the state* $\mathsf{rst_U}$ *contains the commitment* $C$*, which can be accessed as* $\mathsf{rst_U}.C$*. It further contains information that the user needs to participate in signing and showing interactions.*

  - *The* signer's registration algorithm $\mathcal{RG}_{\mathsf{S}}$*: Take as input a commitment* $C$*, a proof* $\pi$*, and a signer's registration state* $\mathsf{rst}'_{\mathsf{S}}$*. Output an updated signer's registration state* $\mathsf{rst}_{\mathsf{S}}$ *or* $\perp$ *(*Error*) to the signer, and return* 1 *(*complete*) or* 0 *(*Error*) to the user. The signer uses* $\mathsf{rst}_{\mathsf{S}}$ *later to identify registered commitments.*
- *The interactive* signing/issuance algorithm $\mathsf{S}$*: Take as input a secret key* $\mathsf{sk}$*, a registration state* $\mathsf{rst}_{\mathsf{S}}$*, and a commitment* $C$*, and output* 1 *(*complete*) or* $\perp$ *(*Error*).*
- *The interactive* user algorithm $\mathsf{U}$*: Take as input the signer's public key* $\mathsf{pk}$*, a registration state* $\mathsf{rst}_{\mathsf{U}}$[7]*, and a message* $m$*. Output a signature* $\sigma$ *and an opening state* $\mathsf{ost}_{\mathcal{U}}$*. We assume the signature* $\sigma$ *contains a blinded version of the commitment* $\mathsf{rst}_{\mathsf{U}}.C$ *and* $\mathsf{ost}_{\mathcal{U}}$ *contains an opening for* $\mathsf{rst}_{\mathsf{U}}.C$*.*
- *The interactive* show algorithm $\mathcal{SH}$ *is split into two algorithms,* $\mathcal{SH}_{\mathsf{U}}$ *and* $\mathcal{SH}_{\mathcal{V}}$*:*
  - *The* show generation algorithm $\mathcal{SH}_{\mathsf{U}}$*: Take as input a public key* $\mathsf{pk}$*, the user's registration and opening states* $\mathsf{rst}_{\mathsf{U}}, \mathsf{ost}_{\mathcal{U}}$*, a message* $m$*, a signature* $\sigma$*, and an index set* $I_{op}$*. Output a partial vector* $\boldsymbol{L}'$ *and a proof* $\bar{\pi}$*.*
  - *The deterministic* show verification algorithm $\mathcal{SH}_{\mathcal{V}}$*: Take as input a public key* $\mathsf{pk}$*, a message* $m$*, a signature* $\sigma$*, a (partial) attribute vector* $\boldsymbol{L}'$*, an index set* $I_{op}$ *corresponding to the non-*$\perp$ *indices in* $\boldsymbol{L}'$*, and a proof* $\bar{\pi}$*. Output* 1 *(*Accept*) or* 0 *(*Reject*).*
- *The deterministic* signature verification algorithm $\mathcal{V}$*: Take as input a public key* $\mathsf{pk}$*, a message* $m$*, and a signature* $\sigma$*. Output* 1 *(*Accept*) or* 0 *(*Reject*).*

*Remark 1.* Blind Signatures with Attributes are equipped with two verification algorithms, $\mathcal{V}$ and $\mathcal{SH}_{\mathcal{V}}$. While $\mathcal{V}$ checks a signature's validity, i.e., whether the signature was issued by an honest signer, $\mathcal{SH}_{\mathcal{V}}$ checks whether a *show* tuple consists of a valid signature and a valid proof of knowledge of a vector of attributes for which the signature was granted.

*Remark 2.* In our registration algorithm, we do not require the user attributes to be hidden in the commitment, i.e., the commitment can be the vector $\boldsymbol{L}$ of the attributes, and the proof can be $\perp$. Depending on the protocol or the application, the user may be required to reveal some or all of their attributes at registration. In fact, in our transformation $\mathbf{P}[\mathsf{BS}]$, the user is required to reveal all of their attributes, whereas in $\mathsf{ACL}$, all attributes may remain hidden at registration.

**Definition 2 (Correctness of Blind Signatures with Attributes).** *We say that a blind signature with attributes scheme* $\mathsf{BSA} = (\mathcal{G}, \mathcal{RG}, \mathsf{S}, \mathsf{U}, \mathcal{V}, \mathcal{SH})$ *is perfectly correct if the following holds: for all public parameters* $\mathsf{pp}$*, all vectors of attributes* $\boldsymbol{L}$*, all signer registration states* $\mathsf{rst}'_{\mathsf{S}}$*, all index sets* $I_{op} \subset [\nu]$*, and all messages* $m$*:*

1. *Generate a pair of keys* $(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathcal{G}(\mathsf{pp})$*.*
2. *Run the registration protocol honestly by executing* $(\mathsf{rst}_{\mathsf{U}}, \pi) \leftarrow_{\$} \mathcal{RG}_{\mathsf{U}}(\boldsymbol{L})$ *followed by* $\mathsf{rst}_{\mathsf{S}} \leftarrow \mathcal{RG}_{\mathsf{S}}(\mathsf{rst}_{\mathsf{U}}.C, \pi, \mathsf{rst}'_{\mathsf{S}})$*.*

[7] Recall that $\mathsf{rst}_{\mathsf{U}}$ encodes a commitment accessible by $\mathsf{rst}_{\mathsf{U}}.C$.

3. *Run the signing protocol honestly by the signer* $\mathsf{S}(\mathsf{sk}, \mathsf{rst}_\mathsf{S}, \mathsf{rst}_\mathsf{U}.C)$ *and the user* $\mathsf{U}(\mathsf{pk}, \mathsf{rst}_\mathsf{U}, m)$ *resulting in the signer outputting* 1 *and the user obtaining a signature* $\sigma$ *and an opening state* $\mathsf{ost}_\mathcal{U}$.
4. *Run the user's showing procedure honestly by executing* $(\boldsymbol{L}', \bar{\pi}) \leftarrow_\$ \mathcal{SH}_\mathsf{U}(\mathsf{pk}, \mathsf{rst}_\mathsf{U}, m, \sigma, \mathsf{ost}_\mathcal{U}, I_{op})$.
5. *Both verifications* $\mathcal{V}(\mathsf{pk}, m, \sigma)$ *and* $\mathcal{SH}_\mathcal{V}(\mathsf{pk}, m, \sigma, \boldsymbol{L}', I_{op}, \bar{\pi})$ *output 1.*

We recall the definition of strong one-more unforgeability, where an adversary wins if they are able to output $\ell + 1$ valid message-signature pairs after completing only $\ell$ signing sessions. In the setting of blind signatures with attributes, this game is augmented with a registration oracle. The registration oracle $\mathcal{RG}_\mathsf{S}$ allows the adversary to register users with attributes of their choice. The game keeps the internal state the signer would keep regarding registered users, which includes a list of the registered attribute commitments. It responds with 1/0 if the signer-side registration algorithm outputs $\mathsf{rst}_\mathsf{S}/\perp$, respectively. The signing oracle $\mathsf{S}(\mathsf{sk}, \cdot)$ simulates the signing algorithm for the adversary. In schemes with multiple rounds, we denote the signing algorithm to be split into multiple sub-algorithms. Then, the adversary is provided with an oracle for each sub-algorithm, and the experiment maintains a state internally. If the scheme consists of multiple rounds, the adversary may query the signing oracles concurrently and in an interleaved manner (e.g., make many calls to $\mathcal{S}_1$ before making the call with the same session identifier to $\mathcal{S}_2$).

**Definition 3 (Strong One-More Unforgeability (OMUF)).** *For a blind signature with attributes* $\mathsf{BSA} = (\mathcal{G}, \mathcal{RG}, \mathsf{U}, \mathsf{S}, \mathcal{V}, \mathcal{SH})$*, public parameters* $\mathsf{pp}$*, a positive integer* $\ell \in \mathbb{Z}^+$*, and an adversary* $\mathcal{A}$*, we define Game* $\ell$-$\mathsf{OMUF}$ *as follows:*

1. *Generate key pair* $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathcal{G}(\mathsf{pp})$.
2. *Invoke* $\mathcal{A}^{\mathcal{RG}_\mathsf{S}(\cdot), \mathsf{S}(\mathsf{sk}, \cdot)}(\mathsf{pk})$. $\mathcal{A}$ *may query their oracles* $\mathcal{RG}_\mathsf{S}(\cdot)$ *and* $\mathsf{S}(\mathsf{sk}, \cdot)$ *arbitrarily and in an interleaved fashion as long as they complete at most* $\ell$ *sessions with* $\mathsf{S}$.
3. $\mathcal{A}$ *outputs* $\ell + 1$ *distinct tuples* $(\sigma_1, m_1), \ldots, (\sigma_{\ell+1}, m_{\ell+1})$.
4. *Output 1 if for all* $i \in [\ell + 1]$, $\mathcal{V}(\mathsf{pk}, m_i, \sigma_i) = 1$.

*We define* $\mathcal{A}$*'s advantage in winning the game* $\ell$-$\mathsf{OMUF}$ *against a blind signature with attributes* $\mathsf{BSA}$ *as*

$$\mathbf{Adv}^{\ell\text{-}\mathsf{OMUF}}_{\mathsf{BSA}}(\mathcal{A}) \coloneqq \Pr[\ell\text{-}\mathsf{OMUF}^{\mathcal{A}}_{\mathsf{BSA}} = 1].$$

*We say that* $\mathsf{BSA}$ *is* $(t, \varepsilon, \ell)$-$\mathsf{OMUF}$*-secure if for all adversaries* $\mathcal{A}$ *that run in time at most* $t$*,* $\mathbf{Adv}^{\ell\text{-}\mathsf{OMUF}}_{\mathsf{BSA}}(\mathcal{A}) \leq \varepsilon$.

## 4 Three-Move Simulatable Blind Signature Schemes

In this section, we describe the class of blind signature schemes underlying our generic construction.

**Definition 4 (Three-Move Simulatable Blind Signature Schemes).** *For a security parameter $\lambda$, let $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$ be a hash function treated as a random oracle and let $\mathsf{pp}$ be a public parameter. We consider blind signature schemes $\mathsf{BS}^{\mathsf{H}} = (\mathcal{G}^{\mathsf{H}}, \mathcal{S}, \mathcal{SIM}_{\mathcal{S}}, \mathcal{U}^{\mathsf{H}}, \mathcal{V}^{\mathsf{H}})$ that have the following structure:*

- ***Key Generation*** *$\mathcal{G}$: Take as input public parameters $\mathsf{pp}$ and output a key pair $(\mathsf{pk}, \mathsf{sk})$.*
- ***Signer Algorithms*** *$\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$. The signer algorithms are split into two:*
  - *$\mathcal{S}_1^{\mathsf{H}}(\mathsf{sk})$ The* probabilistic *first-stage signer algorithm: Take as input the secret key $\mathsf{sk}$ and output a first signer message $\mathsf{msgS1}$ and the signer's randomness $\mathsf{rnd_S}$.*
  - *$\mathcal{S}_2^{\mathsf{H}}(\mathsf{sk}, \mathsf{rnd_S}, \mathsf{msgU})$ The* deterministic *second-stage signer algorithm: Take as input the secret key $\mathsf{sk}$, the signer's randomness $\mathsf{rnd_S}$, and the user message $\mathsf{msgU}$, and output a response $\mathsf{msgS2}$.*
- ***User Algorithms*** *$\mathcal{U}^{\mathsf{H}} = (\mathcal{U}_1^{\mathsf{H}}, \mathcal{U}_2^{\mathsf{H}})$. The user algorithms are split into two:*
  - *$\mathcal{U}_1^{\mathsf{H}}(\mathsf{pk}, m, \mathsf{msgS1}; \mathsf{Urnd})$ The* probabilistic *first-stage user algorithm: Take as input the public key $\mathsf{pk}$, the message $m$, and a first signer message $\mathsf{msgS1}$. We write the entire user randomness $\mathsf{Urnd}$ explicitly here. Output a user message $\mathsf{msgU}$.*
  - *$\mathcal{U}_2^{\mathsf{H}}(\mathsf{pk}, m, \mathsf{Urnd}, \mathsf{msgS2})$ The* deterministic *second-stage user algorithm: Take as input the public key $\mathsf{pk}$, the message $m$, the first-stage user randomness $\mathsf{Urnd}$, and the second signer message $\mathsf{msgS2}$. Output a signature $\sigma$.*
- ***Verification*** *$\mathcal{V}^{\mathsf{H}}(\mathsf{pk}, m, \sigma)$ The deterministic verification algorithm: Take as input a public key $\mathsf{pk}$, a message $m$, and a signature $\sigma$, and output 1 (Accept) or 0 (Reject).*
- ***Signer Simulation Algorithms*** *$\mathcal{SIM}_{\mathcal{S}}^{\mathsf{H}} = (\mathcal{SIM}_{\mathcal{S}_1}^{\mathsf{H}}, \mathcal{SIM}_{\mathcal{S}_2}^{\mathsf{H}}, \mathcal{SIM}_{\mathsf{H}})$. They are defined as follows:*
  - *$\mathcal{SIM}_{\mathcal{S}_1}(\mathsf{pk})$: Take as input a public key $\mathsf{pk}$ and output a signer message $\mathsf{msgS1}$ and a state $\mathsf{simSt}$.*
  - *$\mathcal{SIM}_{\mathcal{S}_2}(\mathsf{pk}, \mathsf{msgS1}, \mathsf{simSt}, \mathsf{Urnd}, m)$: Take as input a public key $\mathsf{pk}$, a signer first message $\mathsf{msgS1}$, a simulation state $\mathsf{simSt}$, a message $m$, and user randomness $\mathsf{Urnd}$. Output a signer message $\mathsf{msgS2}$, such that $\mathsf{msgS1}$ and $\mathsf{msgS2}$ are identically distributed to a real execution of $\mathcal{S}_1$ and $\mathcal{S}_2$ with $\mathsf{msgU} = \mathcal{U}_1^{\mathsf{H}'}(\mathsf{pk}, m, \mathsf{msgS1}; \mathsf{Urnd})$ where $\mathsf{H}'$ returns the hash response provided by $\mathcal{SIM}_{\mathsf{H}}$.*
  - *$\mathcal{SIM}_{\mathsf{H}}(\mathsf{pk}, \mathsf{msgS1}, \mathsf{simSt}, \mathsf{Urnd}, m)$: Take as input a public key $\mathsf{pk}$, a signer first message $\mathsf{msgS1}$, a simulation state $\mathsf{simSt}$, a message $m$, and user randomness $\mathsf{Urnd}$. Output a corresponding hash response $c$. In the case of a two-move scheme, we assume that $\mathcal{SIM}_{\mathsf{H}}$ takes as input only the public key $\mathsf{pk}$ and the message $m$, and outputs a hash response $c$ and a simulation state $\mathsf{simSt}$.*

*Throughout this work, we often drop the superscript $\mathsf{H}$ when it is clear from the context.*

Our generic construction requires the underlying blind signature scheme $\mathsf{BS}$ to satisfy the following properties.

*Property 1 (Unique Hash Query).* In an honest execution of $\mathsf{BS}$, the user algorithm $\mathcal{U}_1^{\mathsf{H}}(\mathsf{pk}, m, \mathsf{msgS1}; \mathsf{Urnd})$ makes a single hash query to the random oracle $\mathsf{H}$. We denote this query as $\mathsf{hq}(\mathsf{pk}, m, \mathsf{msgS1}; \mathsf{Urnd})$. Similarly, for each valid signature-message pair $(\sigma, m)$, the verification procedure $\mathcal{V}^{\mathsf{H}}(\mathsf{pk}, m, \sigma)$ makes exactly one query to $\mathsf{H}$. If the signature $\sigma$ was generated honestly, the query made by the verifier $\mathcal{V}$ is identical to the query $\mathsf{hq}(\mathsf{pk}, m, \mathsf{msgS1}; \mathsf{Urnd})$ made by $\mathcal{U}_1$. This query is uniquely determined by the signer message $\mathsf{msgS1}$, the user coins $\mathsf{Urnd}$, and the message $m$, and it contains the message $m$ as an input.

*Property 2 (Signature-Hash Linking).* Let $\mathcal{A}$ be an adversary running in time at most $t_{\mathsf{SHL}}$. $\mathcal{A}$ is invoked with a public key $\mathsf{pk}$ as input and has oracle access to the signing oracles $\mathcal{S}_1(\mathsf{sk}, \cdot)$ and $\mathcal{S}_2(\mathsf{sk}, \cdot)$ in addition to the random oracle $\mathsf{H}$. Once $\mathcal{A}$ terminates, they output a message-signature pair $(m, \sigma)$, and they win if $\mathcal{V}(\mathsf{pk}, m, \sigma) = 1$ and the single random oracle query $Q$ made by $\mathcal{V}$ to $\mathsf{H}$ was not seen before. It must hold

$$\Pr\left[\begin{array}{c|c} \mathcal{V}^{\mathsf{H}}(\mathsf{pk}, m, \sigma) = 1 & (\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathcal{G}(\lambda) \\ \wedge \mathsf{T}[Q] = \bot & (m, \sigma) \leftarrow_{\$} \mathcal{A}^{\mathcal{S}_1, \mathcal{S}_2, \mathsf{H}}(\mathsf{pk}) \end{array}\right] \leq \varepsilon_{\mathsf{SHL}}.$$

*Property 3 (Unforgeability for $\mathcal{U}_1^{\mathsf{H}}$).* Let $\mathcal{A}$ be an adversary running in time at most $t_{\mathsf{UUF}}$ and outputting $(\mathsf{msgU}, \mathsf{pk}, m, \mathsf{msgS1}, \mathsf{Urnd})$, and let $\mathsf{T}$ be a snapshot of the hash table of the oracle $\mathsf{H}$ by the time $\mathcal{A}$ submits their output. Let $Q \coloneqq \mathsf{hq}(\mathsf{pk}, m, \mathsf{msgS1}, \mathsf{Urnd})$ be the corresponding unique hash query made by $\mathcal{U}_1^{\mathsf{H}}(\mathsf{pk}, m, \mathsf{msgS1}; \mathsf{Urnd})$. Then it holds

$$\Pr\left[\begin{array}{c|c} \mathsf{msgU}' = \mathsf{msgU} & (\mathsf{msgU}, \mathsf{pk}, m, \mathsf{msgS1}, \mathsf{Urnd}) \leftarrow_{\$} \mathcal{A} \\ \wedge \mathsf{T}[Q] = \bot & Q \coloneqq \mathsf{hq}(\mathsf{pk}, m, \mathsf{msgS1}, \mathsf{Urnd}) \\ & \mathsf{msgU}' \coloneqq \mathcal{U}_1^{\mathsf{H}}(\mathsf{pk}, m, \mathsf{msgS1}; \mathsf{Urnd}) \end{array}\right] \leq \varepsilon_{\mathsf{UUF}}.$$

*Property 4 (Unique $\mathcal{U}_1$ Output).* Let $\mathcal{A}$ be an adversary that runs in time at most $t_{\mathsf{UUO}}$. $\mathcal{A}$ takes as input a public key $\mathsf{pk}$ and a signer message $\mathsf{msgS1}$, and they output two distinct pairs $(m_1, \mathsf{Urnd}_1)$ and $(m_2, \mathsf{Urnd}_2)$. It holds

$$\Pr\left[\begin{array}{c|c} (m_1, \mathsf{Urnd}_1) \neq (m_2, \mathsf{Urnd}_2) & (m_1, \mathsf{Urnd}_1, m_2, \mathsf{Urnd}_2) \leftarrow_{\$} \mathcal{A}(\mathsf{pk}, \mathsf{msgS1}) \\ \mathsf{msgU}_1 = \mathsf{msgU}_2 & \mathsf{msgU}_1 \coloneqq \mathcal{U}_1^{\mathsf{H}}(\mathsf{pk}, m_1, \mathsf{msgS1}; \mathsf{Urnd}_1) \\ & \mathsf{msgU}_2 \coloneqq \mathcal{U}_1^{\mathsf{H}}(\mathsf{pk}, m_2, \mathsf{msgS1}; \mathsf{Urnd}_2) \end{array}\right] \leq \varepsilon_{\mathsf{UUO}}.$$

*Property 5 (Unique Link from Hash Query to Signing Session).* For any adversary $\mathcal{A}$ that runs in time at most $t_{\mathsf{UL}}$, any $(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathcal{G}$, and any (not necessarily distinct) $\mathsf{msgS1}_1 \leftarrow_{\$} \mathcal{S}_1(\mathsf{sk}), \mathsf{msgS1}_2 \leftarrow_{\$} \mathcal{S}_1(\mathsf{sk})$, it holds

$$\Pr\left[\begin{array}{c|c} (\mathsf{msgS1} \neq \bot \wedge \mathsf{msgS1}' \neq \bot) \wedge & \\ (\mathsf{msgS1}; \mathsf{Urnd}) \neq (\mathsf{msgS1}'; \mathsf{Urnd}') & (m, \mathsf{Urnd}, \mathsf{Urnd}') \leftarrow_{\$} \\ \wedge \mathsf{hq}(\mathsf{pk}, m, \mathsf{msgS1}; \mathsf{Urnd}) & \mathcal{A}(\mathsf{pk}, \mathsf{msgS1}, \mathsf{msgS1}') \\ = \mathsf{hq}(\mathsf{pk}, m, \mathsf{msgS1}'; \mathsf{Urnd}') & \end{array}\right] \leq \varepsilon_{\mathsf{UL}}.$$

*Property 6 (Double Signature Property).* We require that the blind signature scheme has $(t, \varepsilon)$-double-signature resistance as defined below.

We define the double signature game for an adversary $\mathcal{A}$ as follows:

- **Setup:** The game samples a key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{BS}.\mathcal{G}$.
- **Online Phase:** The adversary gets access to the following oracles:
  - **Signer Simulation Algorithm:** Allow $\mathcal{A}$ to generate simulated signatures via oracle access to the subalgorithms $(\mathcal{SIM}_{\mathcal{S}_1}, \mathcal{SIM}_{\mathcal{S}_2}, \mathcal{SIM}_{\mathsf{H}})$. The game internally maintains the simulation state $\mathsf{simSt}$ generated by the algorithms and routes it to subsequent subalgorithms for each session, outputting only the simulated messages (e.g., $\mathsf{msgS1}$) to $\mathcal{A}$.
  - **Random Oracle Queries $\mathsf{H}$:** Answer via lazy sampling.
- **Output Determination:** The adversary outputs two signatures $\sigma_1$ and $\sigma_2$ on the same message $m$. They win the game if $\mathcal{V}^{\mathsf{H}}(\mathsf{pk}, m, \sigma_1) = \mathcal{V}^{\mathsf{H}}(\mathsf{pk}, m, \sigma_2) = 1$ and the singular hash query that is made during verification of both signatures is the same.

The advantage of an adversary in this double signature game is the probability that the game outputs 1. We say the scheme has $(t_{\mathsf{DS}}, \varepsilon_{\mathsf{DS}})$-double-signature resistance if for any adversary that runs in time $t_{\mathsf{DS}}$, the advantage is at most $\varepsilon_{\mathsf{DS}}$.

Many existing blind signature schemes fit our definition 4 and satisfy the properties we defined above, if we apply a slight modification:

- Three-move blind signature schemes such as Abe-Okamoto [3], Abe [1], Blind Schnorr [56], Tessaro and Zhu [58], and $\mathsf{BS}_{[f]}$ [27]. The modification required for these schemes is to sample the user's randomness outside the user algorithm and pass it as input to $\mathcal{U}_1$.
- Two-move blind signature schemes such as blind BLS [12] and blind RSA [22]. To adapt our definition to these schemes, we can define $\mathcal{S}_1$ and $\mathcal{SIM}_{\mathcal{S}_1}$ to output an empty message $\perp$, in addition to sampling the user's randomness outside the user's algorithm.

We give concrete examples showing how blind Schnorr signatures and Abe-Okamoto partially blind signatures fit our definition in section D (Blind Schnorr) and section E (Abe-Okamoto) . For the rest of the schemes, we provide informal arguments as to why they fit this definition below.

*Property 1.* This property requires the random oracle query to be singular and explicitly contain the message $m$ in plain.

- **Abe, Tessaro-Zhu, and $\mathsf{BS}_{[f]}$:** The verification equations for these schemes rely on a standard Fiat-Shamir transformation, evaluating a single hash query that securely binds the reconstructed commitments to the message (e.g., $\mathsf{H}(R, m)$). The $\mathcal{U}_1^{\mathsf{H}}$ algorithm naturally constructs and evaluates this exact query to generate the challenge.
- **Blind BLS and Blind RSA:** Both the initial user blinding phase in $\mathcal{U}_1^{\mathsf{H}}$ and the final verification algorithm $\mathcal{V}^{\mathsf{H}}$ evaluate exactly one hash query consisting strictly of the plain message, $h = \mathsf{H}(m)$.

*Property 2.* This property requires that an adversary cannot forge a valid signature without the corresponding hash query existing in the random oracle table.

- **Abe, Tessaro-Zhu, and $\mathsf{BS}_{[f]}$:** To forge a signature without querying the random oracle, the adversary must blindly guess the output of the hash function to satisfy the verification equations (e.g., $c = \mathsf{H}(g^s y^c, m)$). In the random oracle model, the probability of guessing the oracle's output is at most $2^{-\lambda}$.
- **Blind BLS and Blind RSA:** The verifier maps $\mathsf{H}(m)$ to a group element (BLS) or a full-domain integer (RSA); hence, generating a valid signature without a prior hash query on $m$ implies guessing the random hash output, which, again, is bounded by $2^{-\lambda}$.

*Property 3.* This property ensures the user's output message $\mathsf{msgU}$ is tied to the random oracle's output.

- **Abe, Tessaro-Zhu, and $\mathsf{BS}_{[f]}$:** The user message $\mathsf{msgU}$ (the blinded challenge) is computed as a sum or product of the hash output and the user's randomness (e.g., in Abe, $c' = c - t_2 - t_4$). An adversary cannot output a valid $\mathsf{msgU}$ without knowing the hash output, except by guessing it, which occurs with negligible probability.
- **Blind BLS and Blind RSA:** The blinded user message $\mathsf{msgU}$ is the product of the hash output with a blinding factor, computed as $\mathsf{msgU} \coloneqq \mathsf{H}(m) \cdot g^r$ (BLS) or $\mathsf{msgU} \coloneqq \mathsf{H}(m) \cdot r^e \pmod N$ (RSA). It cannot be generated without querying the oracle, except by guessing, which succeeds only with negligible probability.

*Property 4.* This property requires uniqueness, meaning that distinct message-randomness pairs cannot produce the exact same user message $\mathsf{msgU}$.

- **Abe, Tessaro-Zhu, and $\mathsf{BS}_{[f]}$:** The user message $\mathsf{msgU}$ is the blinded challenge sent to the signer (e.g., $c' = c - t_2 - t_4$). This challenge is derived by blinding a hash output that evaluates group elements reconstructed by the user combining $\mathsf{msgS1}$ with their blinding factors (e.g., $\alpha = g^{t_1} \cdot y^{t_2} \cdot a$). If an adversary finds two distinct sets of inputs that yield the exact same $\mathsf{msgU}$, it mathematically reduces to either finding a hash collision in the random oracle or breaking the algebraic binding of how the user combines the signer's message with the blinding factors (which trivially allows extracting the discrete logarithm of the public key).
- **Blind BLS and Blind RSA:** The user message $\mathsf{msgU}$ is a multiplicatively blinded hash, computed as $\mathsf{msgU} = \mathsf{H}(m) \cdot g^r$ (BLS) or $\mathsf{msgU} = \mathsf{H}(m) \cdot r^e \pmod N$ (RSA). Equating $\mathsf{msgU}$ for distinct inputs forces equations such as $\mathsf{H}(m_1) \cdot g^{r_1} = \mathsf{H}(m_2) \cdot g^{r_2}$. If the messages are identical ($m_1 = m_2$), it follows that $r_1 = r_2$, meaning the inputs were never distinct. If the messages differ ($m_1 \neq m_2$), the collision trivially breaks the underlying cryptographic hardness assumptions (discrete logarithm or RSA) or yields a random oracle collision.

*Property 5.* This property ensures that the exact same hash query cannot be tied to two different signing sessions or distinct user randomness.

- **Abe, Tessaro-Zhu, and $\mathsf{BS}_{[f]}$:** The hash query uniquely evaluates the combination of the signer's first message $\mathsf{msgS1}$ and the user's randomness $\mathsf{Urnd}$. As formally proven for Abe-Okamoto and Blind Schnorr, mapping the same hash query to different $\mathsf{msgS1}$ or $\mathsf{Urnd}$ variables forces a collision in the group elements, which directly allows for discrete logarithm extraction.
- **Blind BLS and Blind RSA:** This is trivially satisfied because the condition $\mathsf{msgS1} \neq \perp$ rules out two-move schemes, since we assume that $\mathsf{msgS1} = \perp$ for two-move signature schemes.

*Property 6.* This property dictates that an adversary cannot output two distinct valid signatures on the same message linked to the same hash query.

- **Abe, Tessaro-Zhu, and $\mathsf{BS}_{[f]}$:** These schemes rely on the Fiat-Shamir transform (e.g., $s = r + c \cdot x$). Two distinct signatures mapping to the same hash query provide two different linear combinations for the same underlying commitments, trivially allowing the extraction of the discrete logarithm (the secret key).
- **Blind BLS and Blind RSA:** Signatures in BLS ($\sigma = \mathsf{H}(m)^x$) and RSA ($\sigma = \mathsf{H}(m)^d \pmod N$) are unique; hence, it is impossible for an adversary to output two distinct valid signatures for the same message, making the advantage exactly 0.

We use the standard definitions for correctness and blindness of blind signature schemes and recall them in section A.3 for completeness.

## 5 A Generic Compiler for Blind Signatures with Attributes

In this section, we present our compiler $\mathsf{Pachirisu}$ that transforms any three-move simulatable blind signature scheme $\mathsf{BS}$ into a blind signature with attributes scheme $\mathbf{P}[\mathsf{BS}]$. $\mathsf{Pachirisu}$ utilizes the cut-and-choose technique [46,18] to sign a commitment to the user's attributes using the underlying scheme $\mathsf{BS}$. The user can later present the signature with an opening for the signed commitment, allowing them to present their attributes selectively alongside the signature.

In contrast to the schemes in [46,18], which run $K$ instances of the signing protocol concurrently, $\mathsf{Pachirisu}$ performs nested signing. While concurrent execution is more efficient, it falls short of achieving strong one-more unforgeability due to mix-and-match attacks. We resolve this by introducing dependencies between the different subsignatures in $\mathsf{Pachirisu}$ and therefore achieve strong one-more unforgeability. For a set of keys $\{\mathsf{sk}_1, \ldots, \mathsf{sk}_k\}$, $\mathsf{Pachirisu}$ runs the cut-and-choose protocol sequentially starting from key $\mathsf{sk}_1$. After each interaction, it uses the message-subsignature pair of the previous interaction (under key $\mathsf{sk}_{i-1}$) as the message to be signed in the current interaction (under key $\mathsf{sk}_i$).

Each $\mathbf{P}[\mathsf{BS}]$ signing session starts with an initialization round. The overview of such a session is depicted in fig. 1. In this round, the signer sends the current cut-and-choose parameter $N$ and commits to their cut-and-choose indices.

This is followed by executing the following subroutine described in fig. 2 $K$ times, once for each of the underlying keys.

For each round, an underlying message $m_i$ is generated, which will be kept secret from the signer. In the first round, this message $m_1$ is just a hash of the actual message $M$ to be signed. In all subsequent rounds, the message $m_i$ is a hash of the previous message $m_{i-1}$ along with components of the previous subsignature. This link between the subsignatures for the different keys ensures strong one-more unforgeability, as it prevents a malicious user from rearranging the subsignatures to generate multiple signatures on the same message.

In the $i$-th run of the subroutine, the signer and user essentially run $N$ subsessions of the signing protocol of $\mathsf{BS}$ with key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ in parallel, but the user is expected to show that they behaved honestly by revealing their random coins for all but one subsession before the signer closes it.

To achieve this, the user is first required to commit to the randomness $\mathsf{Urnd}_{i,j}$ for $j \in [N]$ they will use in each subsession, and the signer computes and sends $N$ first signer messages $\mathsf{msgS1}$ for the underlying scheme after receiving the user's randomness commitments $com_{i,j}$.

Upon receiving the signer's messages, the user generates the messages $\Upsilon_{i,j}$ that will be signed by the underlying scheme. To preserve blindness in the cut-and-choose process, the signed messages $\Upsilon_{i,j}$ contain commitments to the message $m_i$ with fresh Merkle-tree commitments $\zeta_{i,j}$ to all of the user's attributes for each of the cut-and-choose subsessions.

The user runs the user algorithm $\mathsf{BS}.\mathcal{U}_1$ for $\mathsf{pk}_i$ with the message $\Upsilon_{i,j}$, the first signer message $\mathsf{msgS1}_{i,j}$, and the user randomness $\mathsf{Urnd}_{i,j}$ they had previously committed to, to generate the user messages $\mathsf{msgU}_{i,j}$.

After the user sends the user messages $\mathsf{msgU}_{i,j}$ for all cut-and-choose subsessions, the signer opens the commitment to the current subsession's cut-and-choose index $J_i$.

The user now has to open the user randomness commitments and also provide the openings for the Merkle trees used for all subsessions, except the $J_i$-th, to show that they behaved honestly in the signing sessions and that the correct user attributes were committed to.

Only if the commitments are opened correctly does the signer send the final signer message $\mathsf{msgS2}$ for the index $J_i$, from which the user derives the subsignature $\sigma_i$. The $i$-th round of the subroutine outputs the public information $\sigma_i$, the commitment randomness $\eta_{i,J_i}$ for the commitment to $m_i$, and the Merkle-tree commitment $\zeta_{i,J_i}$. The user keeps the private state $\boldsymbol{R}_{i,J_i}$, which is the randomness used to generate $\zeta_{i,J_i}$, to use later in the showing procedure.

If the concluded subsession is not the final subsession $i = K$, then the message $m_i$ along with $(\sigma_i, \eta_i, \zeta_i)$ are hashed together as explained earlier to generate the message $m_{i+1}$ to be signed by the next subsession $i + 1$.

After this subroutine has run for all $K$ key pairs, the user assembles the final signature out of the public components generated in all $K$ subsessions.

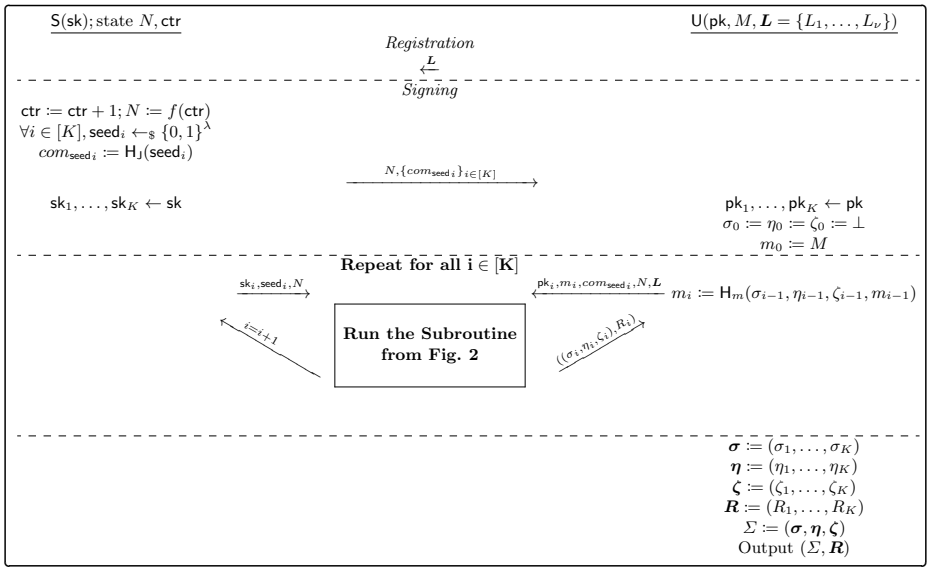


Fig. 1: The transform performs nested signing by applying the signing subroutine from 2 under keys $i = 1, \ldots, K$, such that the message in the $i$-th signing step is a hash of the signature and message from the previous signing step $i - 1$.

**Theorem 1** ($\ell$-OMUF). *Let* $\mathsf{H}, \mathsf{H}_{\mathsf{ro}} : \{0,1\}^* \to \{0,1\}^\lambda$ *for* $\mathsf{ro} \in RO := \{\Upsilon, m, \mu, rnd, \mathsf{MT}\}$ *be hash functions treated as random oracles. Let* $\mathsf{BS}^{\mathsf{H}}$ *be an arbitrary three-move simulatable blind signature scheme with the properties described in Definition 4, and let* $\mathbf{P}[\mathsf{BS}^{\mathsf{H}}]$ *be the scheme resulting from applying our transform on* $\mathsf{BS}^{\mathsf{H}}$*. Assume there exists* $\vartheta > 0$ *and* $f : \mathbb{N} \to \mathbb{N}$ *with* $f(\mathsf{ctr}) = \lceil 3 \cdot \vartheta \cdot \ln(\ell+1) \cdot \mathsf{ctr} \rceil$*. Then for all adversaries* $\mathcal{A}$ *making at most* $\mathsf{Q}_{\mathsf{H}}, \mathsf{Q}_{\mathsf{ro}}$ *queries to the oracles* $\mathsf{H}$ *and* $\mathsf{H}_{\mathsf{ro}}$*, respectively, and* $\mathsf{Q}_{\mathcal{S}_2}$ *and* $\mathsf{Q}_{\mathcal{S}_3}$ *queries to* $\mathbf{P}[\mathsf{BS}^{\mathsf{H}}].\mathcal{S}_2$ *and* $\mathbf{P}[\mathsf{BS}^{\mathsf{H}}].\mathcal{S}_3$*, respectively, and for all public parameters* $\mathsf{pp}$ *with a security parameter* $\lambda$*, and all non-negative integers* $\ell$*, there exists* $\delta > 0$ *with* $(1-\delta)\cdot\vartheta > 1$*, and a reduction* $\mathcal{R}_1$ *such that if* $\mathsf{BS}$ *is* $(t, \varepsilon)$*-0-*$\mathsf{OMUF}$*, then* $\mathbf{P}[\mathsf{BS}]$ *is* $(t', \varepsilon')$*-*$\ell$*-*$\mathsf{OMUF}$ *with* $t' \approx t$ *and*

$$\varepsilon' \leq K \cdot \mathsf{Q}_{\mathsf{H}} \cdot \varepsilon + \epsilon_{cheat} + \epsilon_{ro} + \epsilon_{bs}.$$

*where*

$$\epsilon_{ro} := \frac{\sum_{\mathsf{ro} \in RO} \mathsf{Q}_{\mathsf{ro}} \cdot (\mathsf{Q}_{\mathsf{ro}} - 1)}{2^{\lambda+1}} + \frac{N \cdot \mathsf{Q}_{\mathcal{S}_2} \cdot \mathsf{Q}_{rnd}}{2^\lambda} + \frac{N \cdot \mathsf{Q}_{\mathcal{S}_3} \cdot \mathsf{Q}_\Upsilon}{2^\lambda},$$

$$\epsilon_{cheat} := \exp(-\delta \cdot K),$$

$$\epsilon_{bs} := \varepsilon_{\mathsf{SHL}} \cdot K \cdot (\ell+1) + \ell \cdot K \cdot (N-1) \cdot (\varepsilon_{\mathsf{UUF}} + \varepsilon_{\mathsf{UUO}}) + K \cdot \mathsf{Q}_{\mathsf{H}} \cdot (\varepsilon_{\mathsf{UL}} + \varepsilon_{\mathsf{DS}}).$$

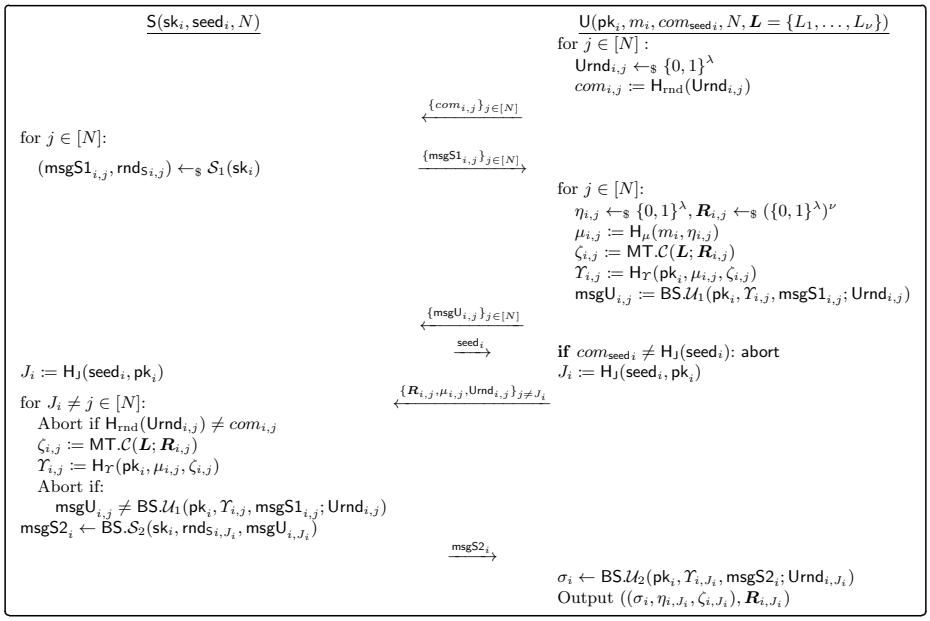


Fig. 2: This is the subroutine used in Figure 1, which shows the execution of the $i$-th subsession between the signer (the owner of $\mathsf{pk}_i$) and the user. $\mathsf{MT}$ is the commitment scheme described in Section 3.1, which relies internally on a hash function $\mathsf{H}_{\mathsf{MT}}$ for hashing. $\mathsf{H}, \mathsf{H}_{\mathrm{rnd}}, \mathsf{H}_{\mathsf{MT}}, \mathsf{H}_\mathsf{J}, \mathsf{H}_\Upsilon, \mathsf{H}_\mu : \{0,1\}^* \to \{0,1\}^\lambda$ are random oracles. $f : \mathbb{N} \to \mathbb{N}$ is an arbitrary function that determines how fast the value $N$ increases.

*Proof Overview.* We prove that the key-only security 0-OMUF of the underlying three-move blind signature scheme BS implies the one-more unforgeability $\ell$-OMUF of $\mathbf{P}[\mathsf{BS}]$, assuming BS satisfies the properties in Section 4. To this end, we provide a reduction $\mathcal{R}_1$ that runs game $\text{0-OMUF}^{\mathcal{R}_1}_{\mathsf{BS}}$ with access to an adversary $\mathcal{A}$ running game $\ell\text{-OMUF}^{\mathcal{A}}_{\mathbf{P}[\mathsf{BS}]}$. The goal of $\mathcal{R}_1$ is to produce a single forgery $(\Upsilon, \sigma)$ that is valid under the challenger's public key $\mathsf{pk}$ and its random oracle $\mathsf{H}_{\mathbf{ch}}$, i.e., $\mathsf{BS}.\mathcal{V}^{\mathsf{H}_{\mathbf{ch}}}(\mathsf{pk}, \Upsilon, \sigma) = 1$.

The reduction $\mathcal{R}_1$ receives the challenger's public key $\mathsf{pk}$ as input. It uses $\mathsf{pk}$ as one of the $K$ public keys it provides $\mathcal{A}$ with and generates the remaining public keys honestly.

Crucially, since $\mathcal{R}_1$ is running Game 0-OMUF with the challenger, it may not make any signing query, while it must answer $\mathcal{A}$'s signing queries ($\mathcal{A}$ may complete up to $\ell$ signing sessions with $\mathcal{R}_1$), including those queries targeting the challenger's key $\mathsf{pk}$ without knowing the corresponding secret key.

During a signing session for $\mathbf{P}[\mathsf{BS}]$, the reduction has to answer signing queries for sub-sessions w.r.t. the challenge public key, as well as the remaining honest keys. For the honest keys, it answers signing queries using the secret key. For the

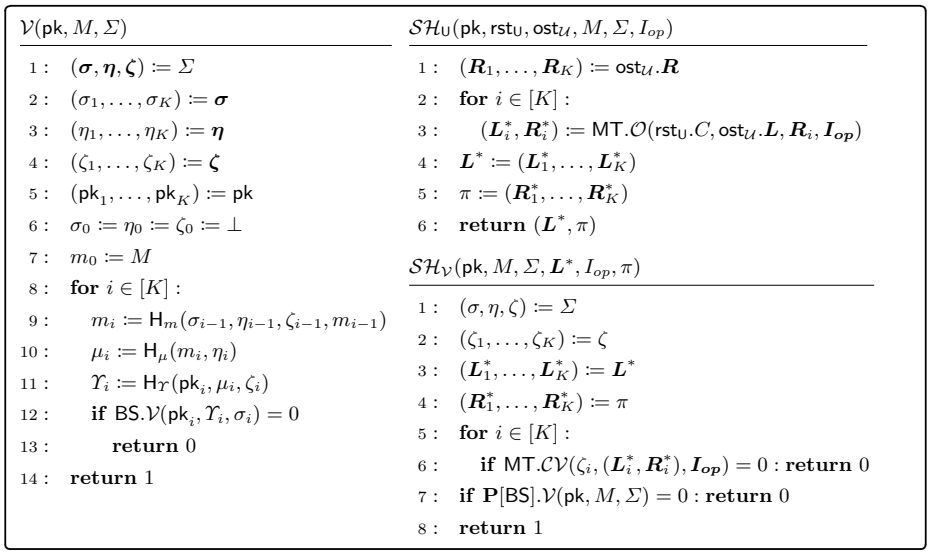


Fig. 3: For a three-move simulatable blind signature scheme $\mathsf{BS}$, this figure shows the signature verification procedure $\mathbf{P}[\mathsf{BS}].\mathcal{V}$, the showing generation algorithm $\mathbf{P}[\mathsf{BS}].\mathcal{SH}_\mathsf{U}$, and the showing verification algorithm $\mathbf{P}[\mathsf{BS}].\mathcal{SH}_\mathcal{V}$ of the blind signature with attributes scheme $\mathbf{P}[\mathsf{BS}]$ resulting from applying our transform Pachirisu on $\mathsf{BS}$.

challenge public key, the reduction's strategy is to use the simulation algorithm $\mathsf{BS}.\mathcal{SIM}$ to answer signing queries. We need to ensure the following things for the reduction to be able to do this:

1. The reduction needs to be able to extract the user randomness $\mathsf{Urnd}$ and the corresponding sub-message $\Upsilon$ that was used to generate the input to the unique hash query (see properties 1 and 2). We argue below how this can be achieved using the cut-and-choose technique.
2. The reduction needs to be able to link the hash input to the signing session at the time of the hash query so that $\mathsf{BS}.\mathcal{SIM}_\mathsf{H}(\mathsf{pk}, \mathsf{msgS1}, \mathsf{simSt}, \mathsf{Urnd}, \Upsilon)$ can be run to generate a hash output. This is achieved using property 5.
3. To simulate the signing response using $\mathsf{BS}.\mathcal{SIM}_{\mathcal{S}_2}(\mathsf{pk}, \mathsf{msgS1}, \mathsf{simSt}, \mathsf{Urnd}, \Upsilon)$, the reduction needs to be able to identify the correct tuple $(\mathsf{simSt}, \mathsf{Urnd}, \Upsilon)$ that was used to generate the user message $\mathsf{msgU}$ that it is trying to respond to. Due to property 3 it holds that when the adversary submits its user message $\mathsf{msgU}$, it must have already made a corresponding hash query to $\mathsf{H}$, and due to property 1, this hash query uniquely determines the tuple $(\mathsf{pk}, \Upsilon, \mathsf{msgS1}, \mathsf{Urnd})$.

We now explain how the cut-and-choose technique ensures that the inputs to the simulation algorithm can be extracted. In the signing (sub)sessions corresponding to the challenger's public key, we need to ensure that $\mathcal{A}$ generates all

$com_{i,j}$ honestly by making random oracle queries to $\mathsf{H}_{\mathrm{rnd}}$. Similarly, we ensure that it generates all $\mathsf{msgU}_{i,j}$ honestly by invoking $\mathsf{BS}.\mathcal{U}_1$ with $\Upsilon_{i,j}$ as input. In this overview we focus on extracting $\mathsf{Urnd}_{i,j}$ since the same arguments apply to $\Upsilon_{i,j}$.

Recall that for all $j \neq J_i$, where $J_i$ is the cut-and-choose index, $\mathcal{A}$ must open the commitment $com_{i,j}$. Consequently, the signer detects and aborts if $\mathcal{A}$ does not generate the commitment honestly. However, for $com_{i,J_i}$, $\mathcal{A}$ may cheat without being detected since it is not required to open this commitment. If $\mathcal{A}$ successfully guesses the cut-and-choose index for a subsession corresponding to the challenger's public key, $\mathcal{R}_1$ fails.

Following the observation from [18], we use the Chernoff bound to show that there is at least one public key for which $\mathcal{A}$ does not cheat (i.e., generates all $com_{i,j}$ honestly for that key's subsessions). If this no-cheat key is the challenger's public key, $\mathcal{R}_1$ can use $\mathsf{H}_{\mathrm{rnd}}$ hash queries to extract all the information it needs for simulating the signer oracles. To leverage this fact, $\mathcal{R}_1$ guesses this no-cheat key and sets it to be the challenger's key.

Once $\mathcal{A}$ terminates and outputs $\ell + 1$ forgeries for $\mathbf{P}[\mathsf{BS}]$, $\mathcal{R}_1$ can extract a forgery for $\mathsf{BS}$ from $\mathcal{A}$'s forgeries. This is possible since a $\mathbf{P}[\mathsf{BS}]$ signature is in fact a nested $\mathsf{BS}$ signature, i.e., a forgery $(M, \Sigma)$ for a public key $\mathbf{pk} \coloneqq \{\mathsf{pk}_1, \ldots, \mathsf{pk}_K\}$ is computed as

$$\Sigma \coloneqq \mathsf{BS}.\mathcal{S}(\mathsf{sk}_k, \mathsf{H}_m(\mathsf{BS}.\mathcal{S}(\mathsf{sk}_{k-1}, \mathsf{H}_m(\ldots \mathsf{BS}.\mathcal{S}(\mathsf{sk}_1, M))))),$$

where $\mathsf{BS}.\mathcal{S}$ denotes the execution of the interactive signing protocol of $\mathsf{BS}$ between the signer and the user. By choosing $\mathbf{pk}$ such that the challenger's key $\mathsf{pk} \in \mathbf{pk}$, $\mathcal{R}_1$ effectively receives a valid message-subsignature pair $(\Upsilon, \sigma)$ for $\mathsf{BS}.\mathcal{V}(\mathsf{pk}, \Upsilon, \sigma)$ if $\mathcal{A}$ outputs a $\mathbf{P}[\mathsf{BS}]$ signature that is valid under $\mathbf{pk}$.

It remains to ensure that at least one of these subsignatures is valid with respect to the challenger's random oracle, i.e. the hash query that must have been made by property 2 was forwarded to the challenger's random oracle for at least one signature. By property 6, no two signatures can be linked to the same hash query, and hence there must be a hash query for which later on no signing session was closed. Let $\mathsf{Q}_{\mathsf{H}}$ be an upper bound for the number of hash queries $\mathcal{A}$ makes to $\mathsf{H}$. $\mathcal{R}_1$ chooses a random $\kappa \in [\mathsf{Q}_{\mathsf{H}}]$ and answers the $\kappa$-th query by forwarding it to the challenger's oracle $\mathsf{H}_{\mathbf{ch}}$. In a *lucky* simulation, $\mathcal{A}$ does not use this query in any signing session, while one of the forgeries relies on it. This effectively provides $\mathcal{R}_1$ with a winning forgery.

The full proof is deferred to Section C.1 .

## 6 Stronger Security Definitions

### 6.1 Selective Secure Showing

The secure showing property as defined in [41] does not cover all possible attacks that one might consider. In particular, the adversary is required to always reveal all attributes. However, as we discussed in the introduction, in the real

world, users typically do not reveal all their attributes, but instead, they selectively disclose only those necessary to access a specific service. Thus, it is imperative that the security notion also captures attacks where adversaries might attempt to alter the attributes contained in their signatures before revealing them.

We model this criterion through the existence of an injective mapping from shown signatures to signing sessions; i.e., each shown partial vector is explained by one signing session's committed attribute vector, and no signing session is being used to "explain" two or more opened signatures. The adversary wins if no such mapping exists.

One concern raised in [41] was that the winning criterion might be inefficient to check because the challenger would have to check that there exists no mapping from the "shown" partial vectors to the full vectors that the adversary requested signatures for. In particular, we cannot trust the adversary to output a "witness" for winning the game (i.e., a bad mapping that leaves at least one vector unmapped), as a better mapping might exist.

We resolve this through the observation that this mapping can be expressed as a matching in a bipartite graph, and we describe a graph-based approach for efficiently checking the winning condition below. See also fig. 4.

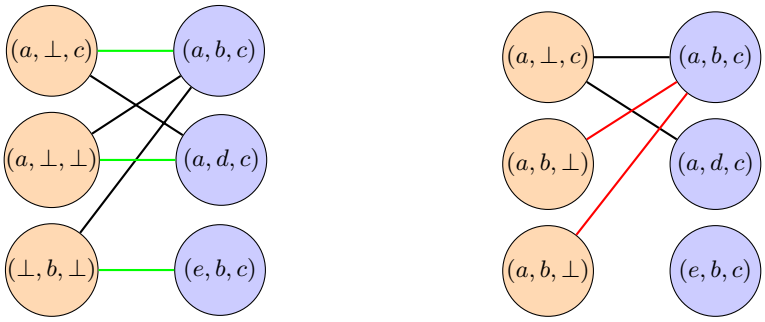


Fig. 4: Left: Opened attribute vectors with possible explanations and a matching where no signing session explains two vectors. Right: Opened attribute vectors where no such explanation exists. The second and third opened vector can only be explained by the first session.

**Definition 5 (Secure Showing).** *For a blind signature with attributes* $\mathsf{BSA} = (\mathcal{G}, \mathcal{RG}, \mathsf{U}, \mathsf{S}, \mathcal{V}, \mathcal{SH})$*, a positive integer* $\ell \in \mathbb{Z}^+$*, and an adversary* $\mathcal{A}$*, we define the game* $\ell$-$\mathsf{SH}$ *(with unique messages/with arbitrary messages) (requiring full opening/with selective reveal) as follows.*

1. *Generate a key pair* $(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathcal{G}(\mathsf{pp})$.
2. *Invoke* $\mathcal{A}^{\mathcal{RG}_{\mathsf{S}}(\cdot), \mathsf{S}(\mathsf{sk}, \cdot)}$*. Oracles* $\mathcal{RG}(\cdot)$ *and* $\mathsf{S}(\mathsf{sk}, \cdot)$ *share state, and we require that session* $i$ *with* $\mathsf{S}(\mathsf{sk}, \cdot)$ *uniquely identifies some prior session* $j$ *with* $\mathcal{RG}(\cdot)$*.*

*The adversary $\mathcal{A}$ may query its oracles arbitrarily and in an interleaved fashion as long as it completes at most $\ell$ sessions with $\mathsf{S}$. For $i \in [\ell]$, let $\mathsf{rst}_{\mathsf{U}}^i.C$ denote the registered commitment corresponding to the $i$-th session with $\mathsf{S}(\mathsf{sk}, \cdot)$. We define $\mathsf{rst}_{\mathsf{U}}^i.C = \perp$ if there is no session $i$ (this occurs if $\mathcal{A}$ makes fewer than $i$ signing queries).*

3. *$\mathcal{A}$ outputs $q \in [\ell]$ tuples of the form $(m_i, \sigma_i, \boldsymbol{L}_i, I_{op,i}, \pi_i)$ as well as $\ell$ pairs $(\boldsymbol{L}'_i, \mathsf{rst}_{\mathsf{U}}^i)$.*
4. *Output 1 if all of the following hold and 0 otherwise:*
   a. *For all $i \in [\ell]$, either $\mathsf{rst}_{\mathsf{U}}^i$ contains a valid opening of $\mathsf{rst}_{\mathsf{U}}^i.C$ to $\boldsymbol{L}'_i$ or $\mathsf{rst}_{\mathsf{U}}^i.C = \perp$ (recall that $\mathsf{rst}_{\mathsf{U}}^i.C = \perp$ when the adversary chooses to open fewer than $i$ signing sessions).*
   b. *For all $(m_i, \sigma_i, \boldsymbol{L}_i, I_{op,i}, \pi_i)$ in $\mathcal{A}$'s output, it holds that $\mathcal{SH}_{\mathcal{V}}(\mathsf{pk}, m_i, \sigma_i, \boldsymbol{L}_i, I_{op,i}, \pi_i) = 1$ and $\mathcal{V}(\mathsf{pk}, m_i, \sigma_i) = 1$.*
   c. *One of the following two requirements on uniqueness of messages holds:*
      i. ***EITHER with unique messages:*** *The messages $m_1, \ldots, m_q$ are pairwise distinct.*
      ii. ***OR with arbitrary messages:*** *The tuples $(m_1, \sigma_1), \ldots, (m_q, \sigma_q)$ are pairwise distinct.*
   d. *One of the following mapping requirements holds:*
      i. ***EITHER Requiring Full Opening:***
         A. *All $\boldsymbol{L}_i, \boldsymbol{L}'_i$ contain an entry for each attribute, i.e., they do not contain the $\perp$ symbol anywhere, and all $I_{op,i} = [\nu]$, i.e., all attributes are opened.*
         B. *The multiset of all attribute vectors $\{\boldsymbol{L}_i\} \not\subseteq \{\boldsymbol{L}'_i \colon \mathsf{rst}_{\mathsf{U}}^i.C \neq \perp\}$.*
      ii. ***OR With Selective Showing:*** *There exists no injective mapping*

$$\mu : \{\boldsymbol{L}_i\}_{i=1}^q \to \{\boldsymbol{L}'_i\}_{i=1}^{\ell}$$

*such that for all $i$, it holds that $\boldsymbol{L}_i \subseteq \mu(\boldsymbol{L}_i)$.*

*We define $\mathcal{A}$'s advantage in winning the game $\ell$-$\mathsf{SH}$ against a blind signature scheme with attributes $\mathsf{BSA}$ as*

$$\mathbf{Adv}_{\mathsf{BSA}}^{\ell\text{-}\mathsf{SH}}(\mathcal{A}) \coloneqq \Pr[\ell\text{-}\mathsf{SH}_{\mathsf{BSA}}^{\mathcal{A}} = 1].$$

*We say that $\mathsf{BSA}$ is $(t, \varepsilon, \ell)$-$\mathsf{SH}$ secure if, for all adversaries that run in time at most $t$, $\mathbf{Adv}_{\mathsf{BSA}}^{\ell\text{-}\mathsf{SH}}(\mathcal{A}) \leq \varepsilon$.*

We give a brief discussion of our new security notion.

**On the Efficiency of the Game.** It is possible to check the winning condition of the above game in time $O(\ell^{2.5})$ by using the Hopcroft-Karp [39] algorithm to find a maximum cardinality matching. To apply the algorithm, we briefly explain how to view the sets of vectors $\{\boldsymbol{L}_i\}_{i=1}^{\ell}$ and $\{\boldsymbol{L}'_i\}_{i=1}^{\ell}$ as a graph: Each vector from the multiset $\{\boldsymbol{L}_i\}_{i=1}^{\ell}$ corresponds to a vertex $v_i$, and each vector from the multiset $\{\boldsymbol{L}'_i\}_{i=1}^{\ell}$ corresponds to a vertex $w_i$ in the graph. There exists an edge between $v_i$ and $w_j$ if and only if $\boldsymbol{L}_i \subset \boldsymbol{L}'_j$.

It is easy to see that this graph is bipartite, as there are only edges between $v_i$s and $w_j$s but not within the set of the $v_i$s or within the set of the $w_j$s.

The running time of the Hopcroft-Karp algorithm is $n^{2.5}$, where $n$ is the number of vertices—in our case, the number of vertices is $2\ell$; thus, the running time of the challenger lies in $O(\ell^{2.5})$.

**On the Relation Between Definition 5 and that of [41].** The definition from [41] is achieved by choosing conditions item 4.c.ii and item 4.d.i above.

It is easy to see that replacing item 4.d.i with item 4.d.ii, one gets a strictly stronger definition. This is because item 4.d.i.B directly implies item 4.d.ii in the case that all attributes are opened (which is ensured by item 4.d.i.A). The check in item 4.d.i.A falls away in item 4.d.ii.

**On the Relation to Non-Transferability and Unforgeability.** Our Strong Secure Showing property might resemble the notions of *non-transferability* of single-use (attribute-less) tokens [30] and *unforgeability* defined for multi-use ABCs [21,36]. Under the unforgeability definition, the adversary would win if it presented a token with respect to a partial vector that is not a subvector of a vector for which it previously obtained a signature, and under non-transferability, the adversary would win if it presented a token that was not issued to it (where users are identified through key pairs). Under Definition 5, the adversary would also win if it presented partial vectors that are subvectors of vectors it previously requested, but with the wrong multiplicity. The unforgeability notion for multi-use ABCs does not capture this scenario, as it allows the user to show the same token several times.

### 6.2 Blindness with Partial Attributes

In previous blindness definitions, only signature verification $\mathcal{V}$ is taken into account, which does not give any privacy or untraceability guarantees to the user once they show their signature and reveal a subset of their attributes using $\mathcal{SH}_U$. Thus, in theory, a blind signature scheme that reveals the user's randomness during $\mathcal{SH}_U$, and thus makes the signatures linkable to signing sessions, could fulfill blindness. We address this issue with the following blindness definition, which allows the malicious signer to also specify the users' attribute vectors and to ask for openings of the signatures on a subset of attributes where the specified attribute vectors coincide, in case both signing sessions succeed. This new blindness notion guarantees users that they are anonymous within a set of users that share the same attributes at the coordinates where they opened their attribute commitment.

**Definition 6 (Blindness with Partial Showing).** *For a blind signature with attributes* $\mathsf{BSA} = (\mathcal{G}, \mathcal{RG}, \mathsf{U}, \mathsf{S}, \mathcal{V}, \mathcal{SH})$ *and an adversary* $\mathcal{A}$*, we define the game* BLIND *as follows:*

1. *Invoke the first signer stage* $\mathcal{A}$*, which outputs a public key* pk*, two messages* $m_0, m_1$*, and two attribute vectors* $\boldsymbol{L}_0, \boldsymbol{L}_1$*.*

2. *Sample a bit* $b \leftarrow \{0,1\}$ *uniformly at random.*
3. *Invoke* $\mathcal{A}^{\mathcal{RG}_{\mathcal{U}}(\boldsymbol{L}_0),\mathcal{RG}_{\mathcal{U}}(\boldsymbol{L}_1),\mathsf{U}^0(\mathsf{pk},m_b,\mathsf{rst}_{\mathsf{U}}^0),\mathsf{U}^1(\mathsf{pk},m_{1-b},\mathsf{rst}_{\mathsf{U}}^1)}$*, where* $\mathcal{A}$ *may query its oracles in an interleaved fashion as long as it queries each oracle once. The oracles behave as follows:*
   a. $\mathcal{RG}_{\mathcal{U}}(\boldsymbol{L}_0), \mathcal{RG}_{\mathcal{U}}(\boldsymbol{L}_1)$*: Register the attribute vectors and locally save* $\mathsf{rst}_{\mathsf{U}}^0$ *(corresponding to* $\boldsymbol{L}_0$*) and* $\mathsf{rst}_{\mathsf{U}}^1$ *(corresponding to* $\boldsymbol{L}_1$*).*
   b. $\mathsf{U}^0(\mathsf{pk}, m_b, \mathsf{rst}_{\mathsf{U}}^0), \mathsf{U}^1(\mathsf{pk}, m_{1-b}, \mathsf{rst}_{\mathsf{U}}^1)$*: If the corresponding registration has been called, run the user side of the signing protocol. Internally use the registration state* $\mathsf{rst}_{\mathsf{U}}^0$ *(for* $\mathsf{U}^0$*) or* $\mathsf{rst}_{\mathsf{U}}^1$ *(for* $\mathsf{U}^1$*). When a user algorithm terminates, locally store the resulting tuple* $(m_b, \sigma_b)$ *or* $(m_{1-b}, \sigma_{1-b})$*, as well as the opening states* $\mathsf{ost}_{\mathcal{U}}^0, \mathsf{ost}_{\mathcal{U}}^1$*.*
4. *Once both* $\mathsf{U}^0$ *and* $\mathsf{U}^1$ *have terminated: if both sessions resulted in a signature, output the two message-signature pairs* $(m_0, \sigma_0)$ *and* $(m_1, \sigma_1)$ *to the adversary. Otherwise, output* $(\perp, \perp)$ *to the adversary.*
5. $\mathcal{A}$ *may output two index sets* $I_0$ *and* $I_1$*. If, for all* $i \in I_0 \cup I_1$*, it holds that* $L_{0,i} = L_{1,i}$ *and the adversary was given two message-signature pairs in the previous phase, run* $(\boldsymbol{L}'_0, \pi_0) \leftarrow_{\$} \mathcal{SH}_U(\mathsf{pk}, \mathsf{rst}_{\mathsf{U}}^b, m_0, \sigma_0, \mathsf{ost}_{\mathcal{U}}^b, I_0)$ *and* $(\boldsymbol{L}'_1, \pi_1) \leftarrow_{\$} \mathcal{SH}_U(\mathsf{pk}, \mathsf{rst}_{\mathsf{U}}^{1-b}, m_1, \sigma_1, \mathsf{ost}_{\mathcal{U}}^{1-b}, I_1)$*, and output* $(\boldsymbol{L}'_0, \pi_0)$ *and* $(\boldsymbol{L}'_1, \pi_1)$ *to the adversary. Otherwise, output* $(\perp, \perp)$ *again to the adversary.*
6. *The adversary outputs a bit* $b'$*.*

*We define the advantage of* $\mathcal{A}$ *in the game* BLIND *as*

$$\mathbf{Adv}_{\mathsf{BSA}}^{\mathsf{BLIND}}(\mathcal{A}) \coloneqq 2 \cdot \left| \Pr[b = b'] - \frac{1}{2} \right|.$$

*We say that* BSA *is* $(t, \varepsilon)$*-blind if, for all adversaries that run in time at most* $t$*,* $\mathbf{Adv}_{\mathsf{BSA}}^{\mathsf{BLIND}}(\mathcal{A}) \leq \varepsilon$*.*

It is easy to see that the new notion of blindness implies the old notions used in [7,41], as any adversary that wins in the old notions can be used to also win in this new notion by merely skipping the showing step (the reduction can request a showing of the empty attribute vector).

## 7 Proving Secure Showing for Our Transformation

**Theorem 2.** *Let* BS *be a blind signature scheme satisfying the properties of Definition 4. Let* $\mathbf{P}[\mathsf{BS}]$ *be the scheme resulting from applying our transform on* BS*. If* $\mathbf{P}[\mathsf{BS}]$ *is* $(t, \varepsilon)$*-*$\ell$*-*OMUF*, then* $\mathbf{P}[\mathsf{BS}]$ *is* $(t', \varepsilon')$*-*$\ell$*-*SH *secure with unique messages (item 4.c.i) and selective reveal (item 4.d.ii), with* $t' \approx t$ *and*

$$\varepsilon' \leq K \cdot \mathbf{Adv}_{\mathsf{MT}}^{\mathsf{vcBINDING}}(\mathcal{R}_2) + \varepsilon.$$

*Simplifying Notation.* A message-signature pair $(M, \Sigma)$ consists of $K$ message-subsignature pairs $(\Upsilon_1, \sigma_1), \ldots, (\Upsilon_K, \sigma_K)$. Similarly, each signing session $\mathsf{ctr}$ consists of $K$ signing subsessions $(\mathsf{ctr}, 1), \ldots, (\mathsf{ctr}, K)$. We say that $(\Upsilon_i, \sigma_i)$ is linked

to a closed signing subsession $(\mathsf{ctr}, i)$ if, for the cut-and-choose index $J_i$ of that subsession, $\mathcal{A}$ submits

$$\mathsf{msgU}_{i,J_i} \coloneqq \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_i, \Upsilon_{i,J_i}, \mathsf{msgS1}_{i,J_i}; \mathsf{Urnd}_{i,J_i})$$

in its $\mathcal{S}_3$ query, subsequently closes the subsession, and the hash query made by $\mathsf{BS}.\mathcal{V}(\mathsf{pk}_i, \Upsilon_i, \sigma_i)$ is the same one made by $\mathsf{BS}.\mathcal{U}_1$ while generating $\mathsf{msgU}_{i,J_i}$, i.e., $Q = \mathsf{hq}(\mathsf{pk}_i, \Upsilon_i, \mathsf{msgS1}_{i,J_i}, \mathsf{Urnd}_{i,J_i})$ (see Property 1).

*Proof Overview.* Recall that winning Game $\ell$-$\mathsf{SH}$ with unique messages and selective reveal requires the adversary $\mathcal{A}$ to output $\ell$ pairs of the form $(\mathsf{rst}^i_{\mathsf{U}}, \boldsymbol{L}'_i)$ along with $q \in [\ell]$ tuples of the form $(M_i, \Sigma_i, \boldsymbol{L}_i, I_{op,i}, \pi_i)$ after completing at most $\ell$ signing sessions with the signing oracle $\mathbf{P}[\mathsf{BS}].\mathcal{S}$, while all the following conditions hold:

1. For all $i \in [\ell]$, either $\mathsf{rst}^i_{\mathsf{U}}$ contains a valid opening of $\mathsf{rst}^i_{\mathsf{U}}.C$ to $\boldsymbol{L}'_i$ or $\mathsf{rst}^i_{\mathsf{U}}.C = \perp$ (recall that $\mathsf{rst}^i_{\mathsf{U}}.C = \perp$ when the adversary chooses to open less than $i$ signing sessions).
2. For all $(M_i, \Sigma_i, \boldsymbol{L}_i, I_{op,i}, \pi_i)$ in $\mathcal{A}$'s output, it holds that $\mathcal{SH}_{\mathcal{V}}(\mathsf{pk}, M_i, \Sigma_i, \boldsymbol{L}_i, I_{op,i}, \pi_i) = 1$ and $\mathcal{V}(\mathsf{pk}, M_i, \Sigma_i) = 1$.
3. The messages $M_1, \ldots, M_q$ are pairwise distinct.
4. There exists no injective mapping

$$\mu : \{\boldsymbol{L}_i\}_{i=1}^{q} \to \{\boldsymbol{L}'_i\}_{i=1}^{\ell}$$

such that for all $i$, it holds that $\boldsymbol{L}_i \subseteq \mu(\boldsymbol{L}_i)$.

Akin to the one-more unforgeability proof, we first obtain a public key $\mathsf{pk}_\iota$ for which $\mathcal{A}$ does not cheat in any signing subsession. We then distinguish two cases regarding the message-subsignature pairs corresponding to $\mathsf{pk}_\iota$.

– Suppose that every message-subsignature pair $(\Upsilon_\iota^{(\mathsf{ctr})}, \sigma_\iota^{(\mathsf{ctr})})$ output by $\mathcal{A}$ is linked to a closed signing subsession $(\mathsf{ctr}', \iota)$. These links induce a mapping from the output message-signature pairs to the signing sessions. We show that this mapping is injective: if two different outputs were linked to the same signing subsession, then their $\iota$-th message-subsignature pairs would be linked to the same hash query. After ruling out collisions in the random oracles used in the construction, this implies that the two outer messages are equal, contradicting the requirement that the messages output by $\mathcal{A}$ are pairwise distinct.
Since $\mathcal{A}$ wins Game $\ell$-$\mathsf{SH}$, this injective mapping cannot match every shown vector to a corresponding attribute vector used during signing. Hence, there is an output tuple with shown vector $\boldsymbol{L}_{\mathsf{ctr}^*}$ that is linked to a signing session $\mathsf{ctr}^{**}$ with $\boldsymbol{L}_{\mathsf{ctr}^*} \not\subseteq \boldsymbol{L}'_{\mathsf{ctr}^{**}}$. We rule out this case via a reduction to the binding property of the underlying commitment scheme: the showing provides an opening of the attribute commitment to $\boldsymbol{L}_{\mathsf{ctr}^*}$, while the signing session provides an opening of the same commitment to $\boldsymbol{L}'_{\mathsf{ctr}^{**}}$.

– Otherwise, there exists a message-subsignature pair corresponding to $\mathsf{pk}_\iota$ that is not linked to any closed signing subsession. Using the same strategy as in the OMUF proof, we reduce this case to the 0-OMUF security of the underlying scheme BS. In particular, we use the same reduction from the proof of theorem 1.

*Proof.* Assume there is an adversary $\mathcal{A}$ winning Game $\ell$-SH by outputting $\ell$ pairs $(\mathsf{rst}_{\mathsf{U}}^{\mathsf{ctr}}, \boldsymbol{L}'_{\mathsf{ctr}})$ for $\mathsf{ctr} \in [\ell]$, along with $q \in [\ell]$ tuples $(M_{\mathsf{ctr}}, \Sigma_{\mathsf{ctr}}, \boldsymbol{L}_{\mathsf{ctr}}, I_{op,\mathsf{ctr}}, \pi_{\mathsf{ctr}})$ for $\mathsf{ctr} \in [q]$, satisfying the constraints defined in Game $\ell$-SH.

Recall that in our transform, the user does not send a commitment of its attributes to the signer in the registration phase, instead it sends the attribute vector in plain $\boldsymbol{L}$. Therefore, whenever $\mathsf{rst}_{\mathsf{U}}^{\mathsf{ctr}}.C \neq \perp$, the winning condition requires $\boldsymbol{L}'_{\mathsf{ctr}}$ to be the attribute vector corresponding to the $\mathsf{ctr}$-th signing session.

Akin to the one-more unforgeability proof, we want to show that the adversary cannot cheat for all keys $\mathsf{pk}_i$ for $i \in [K]$. That is, there must exist some public key $\mathsf{pk}_i$, for which $\mathcal{A}$ does not cheat at all. We note that the simulation of both games $\ell$-OMUF and $\ell$-SH is identical, since the simulator must simulate at most $\ell$ signing sessions for $\mathcal{A}$, in addition to the signer registration oracle and the random oracles. Thus, we use the same game hops we used in the OMUF proof to rule out the same bad events ruled out in the OMUF proof (see section C.1 ). Since most of these games rule out basic bad events (such as finding hash collisions or computing hash preimages), we defer these game hops here and provide them in section C.2 . We end up in $\mathbf{G}_{10}$.

Per this game, we know there is a uniformly chosen $\iota \in [K]$ such that $\mathcal{A}$ does not cheat for $\mathsf{pk}_\iota$. Each signature-message pair $(M_j, \Sigma_j)$ $\mathcal{A}$ outputs consists of $K$ message-subsignature pairs $(\Upsilon_i, \sigma_i)$ for $i \in [K]$. Additionally, recall that $\mathcal{A}$ closes at most $\ell$ signing sessions and each session consists of $K$ subsessions, a subsession per key $\mathsf{pk}_i$. Now, we want to rule out that all message-subsignature pairs corresponding to the $\iota$-th key $\mathsf{pk}_\iota$ are linked to some closed signing subsession corresponding to that key. That is, we want a game ensuring that there exists a message-subsignature pair $(\Upsilon_\iota^{(\mathsf{ctr}^*)}, \sigma_\iota^{(\mathsf{ctr}^*)})$ that is not linked to any closed signing subsession $(\mathsf{ctr}, \iota)$.

*Game* $\mathbf{G}_{11}$. This game is identical to $\mathbf{G}_{10}$, except that it aborts and outputs 0 if Event $\mathbf{BAD}_1$ occurs, which we define as follows:

$\mathbf{BAD}_1$: $\forall \mathsf{ctr} \in [q]\, \exists \mathsf{ctr}' \in [\ell]$ *such that the message-subsignature pair* $(\Upsilon_\iota^{(\mathsf{ctr})}, \sigma_\iota^{(\mathsf{ctr})})$ *is linked to the closed signing subsession* $(\mathsf{ctr}', \iota)$.

*Claim 1.*

$$\mathbf{Adv}^{\mathbf{G}_{11}}(\mathcal{A}) \geq \mathbf{Adv}^{\mathbf{G}_{10}}(\mathcal{A}) - \mathbf{Adv}^{\mathsf{vcBINDING}}_{\mathsf{MT}}(\mathcal{R}_2).$$

*Proof.* Assume $\mathcal{A}$ provokes Event $\mathbf{BAD}_1$, hence each message-subsignature pair corresponding to $\mathsf{pk}_\iota$ is linked to some closed signing subsession. We provide a reduction $\mathcal{R}_2$ that uses $\mathcal{A}$ to break the binding property of the commitment scheme MT (see lemma 7 ).

$\mathcal{R}_2$ simulates $\mathbf{G}_{11}$ for $\mathcal{A}$ perfectly. Once $\mathcal{A}$ terminates, it outputs $\boldsymbol{L}'_j$ for $j \in [\ell]$ and $(M_i, \Sigma_i, \boldsymbol{L}_i, I_{op,i}, \pi_i)$ for $i \in [q]$.

For every shown vector $\boldsymbol{L}_{\mathsf{ctr}}$, $\mathsf{ctr} \in [q]$, let $\boldsymbol{L}'_{\mathsf{ctr}'}$ be the attribute vector corresponding to a signing session $\mathsf{ctr}' \in [\ell]$ such that the message-subsignature pair $(\Upsilon_\iota^{(\mathsf{ctr})}, \sigma_\iota^{(\mathsf{ctr})})$ is linked to the signing subsession $(\mathsf{ctr}', \iota)$, and define $\lambda(\boldsymbol{L}_{\mathsf{ctr}}) \coloneqq \boldsymbol{L}'_{\mathsf{ctr}'}$. We first show that $\lambda$ is injective. Assume towards a contradiction that there exist two distinct shown-vector occurrences $\boldsymbol{L}_{\mathsf{ctr}_1}$ and $\boldsymbol{L}_{\mathsf{ctr}_2}$ such that $\lambda(\boldsymbol{L}_{\mathsf{ctr}_1}) = \lambda(\boldsymbol{L}_{\mathsf{ctr}_2}) = \boldsymbol{L}'_{\mathsf{ctr}'}$. By definition of $\lambda$, the message-subsignature pairs $(\Upsilon_\iota^{(\mathsf{ctr}_1)}, \sigma_\iota^{(\mathsf{ctr}_1)})$ and $(\Upsilon_\iota^{(\mathsf{ctr}_2)}, \sigma_\iota^{(\mathsf{ctr}_2)})$ are both linked to the same closed signing subsession $(\mathsf{ctr}', \iota)$. By definition of linking, $\mathsf{BS}.\mathcal{V}(\mathsf{pk}_\iota, \Upsilon_\iota^{(\mathsf{ctr}_1)}, \sigma_\iota^{(\mathsf{ctr}_1)})$ and $\mathsf{BS}.\mathcal{V}(\mathsf{pk}_\iota, \Upsilon_\iota^{(\mathsf{ctr}_2)}, \sigma_\iota^{(\mathsf{ctr}_2)})$ make the same hash query as $\mathsf{BS}.\mathcal{U}_1$ in the closed signing subsession $(\mathsf{ctr}', \iota)$. Since the signed message is contained in this query (property 1), it follows that $\Upsilon_\iota^{(\mathsf{ctr}_1)} = \Upsilon_\iota^{(\mathsf{ctr}_2)}$. Recall that $\Upsilon_\iota = \mathsf{H}_\Upsilon(\mathsf{pk}_\iota, \mu_\iota, \zeta_\iota)$ and $\mu_\iota = \mathsf{H}_\mu(m_\iota, \eta_\iota)$. Since collisions in $\mathsf{H}_\Upsilon$ and $\mathsf{H}_\mu$ have been ruled out, we obtain $m_\iota^{(\mathsf{ctr}_1)} = m_\iota^{(\mathsf{ctr}_2)}$. For $i \in [K]$, recall that $m_i = \mathsf{H}_m(\sigma_{i-1}, \eta_{i-1}, \zeta_{i-1}, m_{i-1})$. Since collisions in $\mathsf{H}_m$ have also been ruled out, applying the same argument recursively for $i = \iota, \ldots, 1$ yields $M_{\mathsf{ctr}_1} = m_0^{(\mathsf{ctr}_1)} = m_0^{(\mathsf{ctr}_2)} = M_{\mathsf{ctr}_2}$. This contradicts the requirement that the messages output by $\mathcal{A}$ are pairwise distinct. Consequently, $\lambda$ is injective.

Since $\mathcal{A}$ wins Game $\ell$-SH, there exists no injective mapping from the multiset $\{\boldsymbol{L}_{\mathsf{ctr}}\}_{\mathsf{ctr}=1}^{q}$ to the multiset $\{\boldsymbol{L}'_{\mathsf{ctr}}\}_{\mathsf{ctr}=1}^{\ell}$ that maps every shown vector to a vector containing it. Since $\lambda$ is injective, there must therefore exist a shown-vector occurrence $\boldsymbol{L}_{\mathsf{ctr}^*}$ such that $\boldsymbol{L}_{\mathsf{ctr}^*} \not\subseteq \lambda(\boldsymbol{L}_{\mathsf{ctr}^*})$. Let $\lambda(\boldsymbol{L}_{\mathsf{ctr}^*}) \coloneqq \boldsymbol{L}'_{\mathsf{ctr}^{**}}$. Then it holds $\boldsymbol{L}_{\mathsf{ctr}^*} \not\subseteq \boldsymbol{L}'_{\mathsf{ctr}^{**}}$. Let $(\mathsf{ctr}^{**}, \iota)$ be the signing subsession linked to $(\Upsilon_\iota^{(\mathsf{ctr}^*)}, \sigma_\iota^{(\mathsf{ctr}^*)})$, and $(\zeta_1^{(\mathsf{ctr}^*)}, \ldots, \zeta_K^{(\mathsf{ctr}^*)}) \coloneqq \boldsymbol{\zeta}_{\mathsf{ctr}^*}$. $\mathcal{R}_2$ gets the commitment and the first opening $(\zeta_\iota^{(\mathsf{ctr}^*)}, (\boldsymbol{L}_{\mathsf{ctr}^*}, \pi_{\mathsf{ctr}^*}))$ directly from $\mathcal{A}$'s output. Now, $\mathcal{R}_2$ has to find another opening, which it can extract from the tables of the random oracles from the entries corresponding to the subsession $(\mathsf{ctr}^{**}, \iota)$.

Let $\mathsf{msgU}_{\iota,J_\iota}^{(\mathsf{ctr}^{**})}$ be the user message corresponding to the index $J_\iota$ for the subsession $(\mathsf{ctr}^{**}, \iota)$, i.e.,

$$\mathsf{msgU}_{\iota,J_\iota}^{(\mathsf{ctr}^{**})} \coloneqq \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_\iota, \Upsilon_{\iota,J_\iota}^{(\mathsf{ctr}^{**})}, \mathsf{msgS1}_{\iota,J_\iota}^{(\mathsf{ctr}^{**})}; \mathsf{Urnd}_{\iota,J_\iota}^{(\mathsf{ctr}^{**})}).$$

Due to Games $\mathbf{G}_7$ and $\mathbf{G}_9$, $\mathcal{R}_2$ can extract the message $\Upsilon_{\iota,J_\iota}^{(\mathsf{ctr}^{**})}$ used to generate $\mathsf{msgU}_{\iota,J_\iota}^{(\mathsf{ctr}^{**})}$. Additionally, due to Game $\mathbf{G}_8$, $\mathcal{R}_2$ can look up the message $\Upsilon_{\iota,J_\iota}^{(\mathsf{ctr}^{**})}$ in the table of $\mathsf{H}_\Upsilon$ to extract the query $(\mathsf{pk}_\iota, \mu_{\iota,J_\iota}^{(\mathsf{ctr}^{**})}, \zeta_{\iota,J_\iota}^{(\mathsf{ctr}^{**})})$ used to generate $\Upsilon_{\iota,J_\iota}^{(\mathsf{ctr}^{**})}$. Since $\mathcal{A}$ does not cheat for $\mathsf{pk}_\iota$, the commitment $\zeta_{\iota,J_\iota}^{(\mathsf{ctr}^{**})}$ is a commitment to the attribute vector $\boldsymbol{L}'_{\mathsf{ctr}^{**}}$ used in signing session $\mathsf{ctr}^{**}$. Thus, by looking up the corresponding entries in the table of $\mathsf{H}_{\mathsf{MT}}$, $\mathcal{R}_2$ obtains randomness $\boldsymbol{R}_{\iota,J_\iota}^{(\mathsf{ctr}^{**})}$ such that

$$\zeta_{\iota,J_\iota}^{(\mathsf{ctr}^{**})} = \mathsf{MT}.\mathcal{C}(\boldsymbol{L}'_{\mathsf{ctr}^{**}}; \boldsymbol{R}_{\iota,J_\iota}^{(\mathsf{ctr}^{**})}).$$

It remains to show that $\zeta_{\iota,J_\iota}^{(\mathsf{ctr}^{**})} = \zeta_\iota^{(\mathsf{ctr}^*)}$. Recall that the linking between the message-subsignature pair $(\Upsilon_\iota^{(\mathsf{ctr}^*)}, \sigma_\iota^{(\mathsf{ctr}^*)})$ and the signing subsession $(\mathsf{ctr}^{**}, \iota)$ implies that the queries $\mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_\iota, \Upsilon_{\iota,J_\iota}^{(\mathsf{ctr}^{**})}, \mathsf{msgS1}_{\iota,J_\iota}^{(\mathsf{ctr}^{**})}; \mathsf{Urnd}_{\iota,J_\iota}^{(\mathsf{ctr}^{**})})$ and $\mathsf{BS}.\mathcal{V}(\mathsf{pk}_\iota, \Upsilon_\iota^{(\mathsf{ctr}^*)}, \sigma_\iota^{(\mathsf{ctr}^*)})$ make the same hash query. Since the message is part of this query (property 1), it holds that $\Upsilon_\iota^{(\mathsf{ctr}^*)} = \Upsilon_{\iota,J_\iota}^{(\mathsf{ctr}^{**})}$. Since collisions in $\mathsf{H}_\Upsilon$ have been ruled out, the corresponding inputs to $\mathsf{H}_\Upsilon$ are identical and, consequently, $\zeta_\iota^{(\mathsf{ctr}^*)} = \zeta_{\iota,J_\iota}^{(\mathsf{ctr}^{**})}$.

Hence, $\mathcal{R}_2$ obtains the two valid openings $(\boldsymbol{L}_{\mathsf{ctr}^*}, \pi_{\mathsf{ctr}^*})$ and $(\boldsymbol{L}'_{\mathsf{ctr}^{**}}, \boldsymbol{R}_{\iota,J_\iota}^{(\mathsf{ctr}^{**})})$ of the same commitment $\zeta_\iota^{(\mathsf{ctr}^*)}$. The first opening is valid since

$$\mathcal{SH}_\mathcal{V}(\mathsf{pk}, M_{\mathsf{ctr}^*}, \Sigma_{\mathsf{ctr}^*}, \boldsymbol{L}_{\mathsf{ctr}^*}, I_{op,\mathsf{ctr}^*}, \pi_{\mathsf{ctr}^*}) = 1,$$

and the second opening is valid by the argument above. Finally, the two openings are inconsistent since $\boldsymbol{L}_{\mathsf{ctr}^*} \not\subseteq \boldsymbol{L}'_{\mathsf{ctr}^{**}}$. Thus, $\mathcal{R}_2$ breaks the binding property of $\mathsf{MT}$.

By this game, $\mathcal{A}$ outputs a message-subsignature pair corresponding to $\mathsf{pk}_\iota$ that is not linked to any closed signing subsession. We reduce this to $\mathsf{0\text{-}OMUF}$ using the same reduction from the proof of theorem 1. Note that all games from that proof are adopted for this proof, hence none of the abort conditions makes the reduction abort, which concludes the proof. Since the only difference compared to the proof of theorem 1 is the additional game $\mathbf{G}_{11}$, we compute $\mathcal{A}$' advantage as

$$\mathbf{Adv}^{\ell\text{-}\mathsf{SH}}_{\mathbf{P}[\mathsf{BS}]}(\mathcal{A}) \leq K \cdot \mathbf{Adv}^{\mathsf{vcBINDING}}_{\mathsf{MT}}(\mathcal{R}_2) + \mathbf{Adv}^{\ell\text{-}\mathsf{OMUF}}_{\mathbf{P}[\mathsf{BS}]}(\mathcal{R}_1).$$

## Acknowledgments

Foteini Baldimtsi was funded through NSF #2143287. Julia Kastner is supported by the Dutch Research Agenda (NWA) project HAPKIDO (Project No. NWA.1215.18.002), which is financed by the Dutch Research Council (NWO). Julian Loss was funded by the European Union, ERC-2023-STG, Project ID: 101116713 and Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. Omar Renawi was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 50723758.

## 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, Berlin, Heidelberg (May 2001). https://doi.org/10.1007/3-540-44987-6_9

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, Berlin, Heidelberg (Nov 1996). https://doi.org/10.1007/BFb0034851
3. Abe, M., Okamoto, T.: Provably secure partially blind signatures. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 271–286. Springer, Berlin, Heidelberg (Aug 2000). https://doi.org/10.1007/3-540-44598-6_17
4. Apple: icloud private relay overview. https://www.apple.com/icloud/docs/iCloud_Private_Relay_Overview_Dec2021.pdf (2021), accessed: 2026-04-29
5. Baldimtsi, F., Hanzlik, L., Nguyen, Q., Yadav, A.: Non-interactive anonymous tokens with private metadata bit. Cryptology ePrint Archive, Report 2025/430 (2025), https://eprint.iacr.org/2025/430
6. Baldimtsi, F., Kastner, J., Loss, J., Renawi, O.: Secure showing of partial attributes. Cryptology ePrint Archive, Report 2025/161 (2025), https://eprint.iacr.org/2025/161
7. 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). https://doi.org/10.1145/2508859.2516687
8. Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable proofs and delegatable anonymous credentials. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer, Berlin, Heidelberg (Aug 2009). https://doi.org/10.1007/978-3-642-03356-8_7
9. Benhamouda, F., Lepoint, T., Orrù, M., Raykova, M.: Publicly verifiable anonymous tokens with private metadata bit. Cryptology ePrint Archive, Report 2022/004 (2022), https://eprint.iacr.org/2022/004
10. Bichsel, P., Camenisch, J., Groß, T., Shoup, V.: Anonymous credentials on a standard java card. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) ACM CCS 2009. pp. 600–610. ACM Press (Nov 2009). https://doi.org/10.1145/1653662.1653734
11. Blömer, J., Bobolz, J., Diemert, D., Eidens, F.: Updatable anonymous credentials and applications to incentive systems. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 1671–1685. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3354223
12. 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, Berlin, Heidelberg (Jan 2003). https://doi.org/10.1007/3-540-36288-6_3
13. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Berlin, Heidelberg (Aug 2004). https://doi.org/10.1007/978-3-540-28628-8_3
14. Bootle, J., Lyubashevsky, V., Nguyen, N.K., Sorniotti, A.: A framework for practical anonymous credentials from lattices. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part II. LNCS, vol. 14082, pp. 384–417. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38545-2_13
15. Camenisch, J., Hohenberger, S., Kohlweiss, M., Lysyanskaya, A., Meyerovich, M.: How to win the clonewars: Efficient periodic n-times anonymous authentication. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM CCS 2006. pp. 201–210. ACM Press (Oct / Nov 2006). https://doi.org/10.1145/1180405.1180431
16. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EU-

ROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Berlin, Heidelberg (May 2001). https://doi.org/10.1007/3-540-44987-6_7
17. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Berlin, Heidelberg (Aug 2004). https://doi.org/10.1007/978-3-540-28628-8_4
18. Chairattana-Apirom, R., Hanzlik, L., Loss, J., Lysyanskaya, A., Wagner, B.: PI-cut-choo and friends: Compact blind signatures via parallel instance cut-and-choose and more. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part III. LNCS, vol. 13509, pp. 3–31. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15982-4_1
19. Chase, M., Durak, F.B., Vaudenay, S.: Anonymous tokens with stronger metadata bit hiding from algebraic MACs. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part II. LNCS, vol. 14082, pp. 418–449. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38545-2_14
20. Chase, M., Kohlweiss, M., Lysyanskaya, A., Meiklejohn, S.: Malleable signatures: New definitions and delegatable anonymous credentials. In: Datta, A., Fournet, C. (eds.) CSF 2014 Computer Security Foundations Symposium. pp. 199–213. IEEE Computer Society Press (2014). https://doi.org/10.1109/CSF.2014.22
21. Chase, M., Meiklejohn, S., Zaverucha, G.: Algebraic MACs and keyed-verification anonymous credentials. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 2014. pp. 1205–1216. ACM Press (Nov 2014). https://doi.org/10.1145/2660267.2660328
22. 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). https://doi.org/10.1007/978-1-4757-0602-4_18
23. Chaum, D.: Blind signature system. In: Chaum, D. (ed.) CRYPTO'83. p. 153. Plenum Press, New York, USA (1983). https://doi.org/10.1007/978-1-4684-4730-9_14
24. Commission, E.: European digital identity (eudi) regulation. https://digital-strategy.ec.europa.eu/en/policies/eudi-regulation (2026), accessed: 2026-04-29
25. Cramer, R.: Modular Design of Secure yet Practical Cryptographic Protocols. Ph.D. thesis, Universiteit van Amsterdam (Jan 1997)
26. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y. (ed.) CRYPTO'94. LNCS, vol. 839, pp. 174–187. Springer, Berlin, Heidelberg (Aug 1994). https://doi.org/10.1007/3-540-48658-5_19
27. Crites, E.C., Komlo, C., Maller, M., Tessaro, S., Zhu, C.: Snowblind: A threshold blind signature in pairing-free groups. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part I. LNCS, vol. 14081, pp. 710–742. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38557-5_23
28. Crites, E.C., Lysyanskaya, A.: Delegatable anonymous credentials from mercurial signatures. In: Matsui, M. (ed.) CT-RSA 2019. LNCS, vol. 11405, pp. 535–555. Springer, Cham (Mar 2019). https://doi.org/10.1007/978-3-030-12612-4_27
29. Davidson, A., Goldberg, I., Sullivan, N., Tankersley, G., Valsorda, F.: Privacy pass: Bypassing internet challenges anonymously. PoPETs **2018**(3), 164–180 (Jul 2018). https://doi.org/10.1515/popets-2018-0026
30. Durak, F.B., Marco, L., Talayhan, A., Vaudenay, S.: Non-transferable anonymous tokens by secret binding. In: Luo, B., Liao, X., Xu, J., Kirda, E., Lie, D. (eds.) ACM CCS 2024. pp. 2460–2474. ACM Press (Oct 2024). https://doi.org/10.1145/3658644.3670338

31. Facebook: Facebook private stats. https://research.fb.com/privatestats (2024), accessed: 2026-01-29
32. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Berlin, Heidelberg (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12
33. 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, Cham (Aug 2018). https://doi.org/10.1007/978-3-319-96881-0_2
34. Google: Private state tokens. https://developers.google.com/privacy-sandbox/protections/private-state-tokens, accessed: 2026-01-29
35. Hanzlik, L., Lai, Y.F., Mula, M., Paracucchi, E., Slamanig, D., Tang, G.: Tanuki: New frameworks for (concurrently secure) blind signatures from post-quantum group actions. In: Hanaoka, G., Yang, B.Y. (eds.) ASIACRYPT 2025, Part IV. LNCS, vol. 16248, pp. 35–69. Springer, Singapore (Dec 2025). https://doi.org/10.1007/978-981-95-5113-2_2
36. Hanzlik, L., Slamanig, D.: With a little help from my friends: Constructing practical anonymous credentials. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021. pp. 2004–2023. ACM Press (Nov 2021). https://doi.org/10.1145/3460120.3484582
37. 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, Cham (May 2019). https://doi.org/10.1007/978-3-030-17659-4_12
38. Hinterwälder, G., Zenger, C.T., Baldimtsi, F., Lysyanskaya, A., Paar, C., Burleson, W.P.: Efficient E-cash in practice: NFC-based payments for public transportation systems. In: De Cristofaro, E., Wright, M.K. (eds.) PETS 2013. LNCS, vol. 7981, pp. 40–59. Springer, Berlin, Heidelberg (Jul 2013). https://doi.org/10.1007/978-3-642-39077-7_3
39. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM Journal on Computing **2**(4), 225–231 (1973). https://doi.org/10.1137/0202019, https://doi.org/10.1137/0202019
40. IETF: Private pass. https://datatracker.ietf.org/wg/privacypass/about/ (2020), accessed: 2026-01-29
41. Kastner, J., Loss, J., Renawi, O.: Concurrent security of anonymous credentials light, revisited. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) ACM CCS 2023. pp. 45–59. ACM Press (Nov 2023). https://doi.org/10.1145/3576915.3623184
42. Kastner, J., Loss, J., Renawi, O.: Concurrent security of anonymous credentials light, revisited. Cryptology ePrint Archive, Report 2023/707 (2023), https://eprint.iacr.org/2023/707
43. Kastner, J., Loss, J., Xu, J.: The Abe-Okamoto partially blind signature scheme revisited. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part IV. LNCS, vol. 13794, pp. 279–309. Springer, Cham (Dec 2022). https://doi.org/10.1007/978-3-031-22972-5_10
44. Kastner, J., Loss, J., Xu, J.: On pairing-free blind signature schemes in the algebraic group model. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) PKC 2022, Part II. LNCS, vol. 13178, pp. 468–497. Springer, Cham (Mar 2022). https://doi.org/10.1007/978-3-030-97131-1_16

45. Katsumata, S., Lai, Y.F., LeGrow, J.T., Qin, L.: CSI-Otter: Isogeny-based (partially) blind signatures from the class group action with a twist. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part III. LNCS, vol. 14083, pp. 729–761. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38548-3_24
46. Katz, J., Loss, J., Rosenberg, M.: Boosting the security of blind signature schemes. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part IV. LNCS, vol. 13093, pp. 468–492. Springer, Cham (Dec 2021). https://doi.org/10.1007/978-3-030-92068-5_16
47. Kreuter, B., Lepoint, T., Orrù, M., Raykova, M.: Anonymous tokens with private metadata bit. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 308–336. Springer, Cham (Aug 2020). https://doi.org/10.1007/978-3-030-56784-2_11
48. Lysyanskaya, A., Nolan, E.: $\mathsf{SMA}^2\mathsf{RT}$ : Secret-metadata attribute-based anonymous rate-limited tokens. Cryptology ePrint Archive, Paper 2026/518 (2026), https://eprint.iacr.org/2026/518
49. McConvey, J.R.: Breaking down the european commission's white label age verification app. https://www.biometricupdate.com/202604/breaking-down-the-european-commissions-white-label-age-verification-app, accessed: 2026-05-19
50. Meunier, T., Rubin, C.D., Faz-Hernández, A.: Privacy pass: upgrading to the latest protocol version. https://blog.cloudflare.com/privacy-pass-standard (2024), accessed: 2026-01-29
51. 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, Berlin, Heidelberg (Aug 1993). https://doi.org/10.1007/3-540-48071-4_3
52. Orrù, M., Tessaro, S., Zaverucha, G., Zhu, C.: Oblivious issuance of proofs. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part IX. LNCS, vol. 14928, pp. 254–287. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68400-5_8
53. Paquin, C., Zaverucha, G.: U-prove technology overview v1.1. In: Microsoft Technical Report, https://www.microsoft.com/en-us/research/publication/uprove-cryptographic-specification-v1-1-revision-3/ (2013)
54. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 129–140. Springer, Berlin, Heidelberg (Aug 1992). https://doi.org/10.1007/3-540-46766-1_9
55. Sanders, O.: Efficient redactable signature and application to anonymous credentials. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part II. LNCS, vol. 12111, pp. 628–656. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45388-6_22
56. 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, Berlin, Heidelberg (Nov 2001). https://doi.org/10.1007/3-540-45600-7_1
57. Silde, T., Strand, M.: Anonymous tokens with public metadata and applications to private contact tracing. In: Eyal, I., Garay, J.A. (eds.) FC 2022. LNCS, vol. 13411, pp. 179–199. Springer, Cham (May 2022). https://doi.org/10.1007/978-3-031-18283-9_9

58. Tessaro, S., Zhu, C.: Short pairing-free blind signatures with exponential security. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 782–811. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07085-3_27

# Supplementary Material

## A Additional Preliminaries

### A.1 Groups and Hardness Assumptions

**Definition 7 (AGM/Algebraic Algorithms [33]).** *We say that an algorithm is* algebraic *with respect to a group $\mathbb{G}$ if it takes as input a set of group elements $g_1, \ldots, g_n$ (w.l.o.g. we assume that one of them is the generator $g$ and in case of interactive algorithms group elements in the output of any potential oracles are added to the set) along with a bit string input $s$ and for every group element $y$ that it outputs it also outputs an* explanation vector *(sometimes also called the* representation*) $\boldsymbol{z}$ such that*

$$y = \prod_{i=1}^{n} g_i^{z_i}.$$

*In the* algebraic group model (AGM) *it is assumed that every algorithm is algebraic.*

**Definition 8 (Discrete Logarithm Problem (DLP)).** *We define the discrete logarithm game $\mathsf{DLP}_{\mathbb{G}}$ for the group $\mathbb{G}$ with generator $g$ and prime order $p$ as follows:*

1. *Sample $x \leftarrow_{\$} \mathbb{Z}_p$. Invoke the adversary on input $g, g^x$.*
2. *The adversary outputs a candidate solution $x' \in \mathbb{Z}_p$, the game outputs 1 if $g^{x'} = y$, 0 otherwise.*

*We define the advantage of an adversary $\mathcal{A}$ in the game $\mathsf{DLP}$ as*

$$\mathbf{Adv}_{\mathbb{G}}^{\mathsf{DLP}}(\mathcal{A}) \coloneqq \Pr[\mathsf{DLP}_{\mathbb{G}}^{\mathcal{A}} = 1]$$

*and say that $\mathsf{DLP}$ is $(t, \varepsilon)$-hard in the group $\mathbb{G}$ if for any adversary $\mathcal{A}$ that runs in time at most $t$, it holds that $\mathbf{Adv}_{\mathbb{G}}^{\mathsf{DLP}}(\mathcal{A}) \leq \varepsilon$.*

**Definition 9 (Decisional Diffie-Hellman Problem (DDH)).** *We define the Decisional Diffie-Hellman game $\mathsf{DDH}_{\mathbb{G}}$ for the group $\mathbb{G}$ with generator $g$ and prime order $p$ as follows:*

1. *Sample $x, y, z \leftarrow_{\$} \mathbb{Z}_p$ and a bit $b \leftarrow_{\$} \{0, 1\}$.*
2. *Invoke the adversary on input $g, X = g^x, Y = g^y, Z = g^{xy+bz}$.*
3. *The adversary outputs a candidate solution $b' \in \mathbb{Z}_p$.*

*We define the advantage of an adversary $\mathcal{A}$ in the game $\mathsf{DDH}_{\mathbb{G}}$ as*

$$\mathbf{Adv}_{\mathbb{G}}^{\mathsf{DDH}}(\mathcal{A}) \coloneqq 2 \cdot \left| \Pr[b = b'] - \frac{1}{2} \right|$$

*and say that $\mathsf{DDH}$ is $(t, \varepsilon)$-hard in the group $\mathbb{G}$ if for any adversary $\mathcal{A}$ that runs in time at most $t$, it holds that $\mathbf{Adv}_{\mathbb{G}}^{\mathsf{DDH}}(\mathcal{A}) \leq \varepsilon$.*

### A.2 Chernoff Bound

We recall a helpful lemma which we use later in this work.

**Lemma 1 (Chernoff Bound).** *Let $X$ be the sum of independent $\{0,1\}$-random variables. Let $s \in \mathbb{N}$ with $s > \mathbb{E}[X]$. It holds that*

$$\Pr[X \geq s] \leq \exp(3 \cdot \mathbb{E}[X] - s).$$

For the proof, see Lemma E.1. in [18].

### A.3 Definitions for Three-Move Blind Signatures

We recall the standard definitions for blind signatures.

**Definition 10 (Correctness).** *We say that a blind signature scheme $\mathsf{BS} = (\mathcal{G}, \mathsf{S} = (\mathcal{S}_1, \mathcal{S}_2), \mathsf{U} = (\mathcal{U}_1, \mathcal{U}_2, \mathcal{V})$ is perfectly correct if the following holds: for all public parameters $\mathsf{pp}$ and all messages $m$,*

1. *Generate a pair of keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathcal{G}(\mathsf{pp})$.*
2. *Run the signing protocol honestly by the signer $\mathsf{S}(\mathsf{sk})$ and the user $\mathsf{U}(\mathsf{pk}, m)$ resulting in the user obtaining a signature $\sigma$*
3. *Verification $\mathcal{V}(\mathsf{pk}, m, \sigma)$ outputs 1.*

**Definition 11 (Blindness).** *For a blind signature $\mathsf{BS} = (\mathcal{G}, \mathsf{S} = (\mathcal{S}_1, \mathcal{S}_2), \mathsf{U} = (\mathcal{U}_1, \mathcal{U}_2), \mathcal{V})$ and an adversary $\mathcal{A}$ we define the game $\mathsf{BLIND}$ as follows:*

1. *Invoke the first signer stage $\mathcal{A}$ that outputs a public key $\mathsf{pk}$, two messages $m_0, m_1$.*
2. *Sample a bit $b \leftarrow \{0,1\}$ uniformly at random.*
3. *Invoke $\mathcal{A}^{\mathsf{U}^0(\mathsf{pk}, m_b), \mathsf{U}^1(\mathsf{pk}, m_{1-b})}$ where $\mathcal{A}$ may query its oracles in an interleaved fashion as long as it queries each oracle once. The oracles behave as follows:*
   a. *$\mathsf{U}^0(\mathsf{pk}, m_b), \mathsf{U}^1(\mathsf{pk}, m_{1-b})$: The oracles run the user-side of the signing protocol. When a user algorithm terminates, the game locally stores the tuple $(m_b, \sigma_b)$ resp. $(m_{1-b}, \sigma_{1-b})$*
4. *Once both $\mathsf{U}^0$ and $\mathsf{U}^1$ have terminated, if both sessions resulted in a signature, output the two message-signature pairs $(m_0, \sigma_0)$ and $(m_1, \sigma_1)$ to the adversary. Otherwise, output $(\bot, \bot)$ to the adversary.*
5. *The adversary outputs a bit $b'$.*

*We define the advantage of $\mathcal{A}$ in the game $\mathsf{BLIND}$ as*

$$\mathbf{Adv}^{\mathsf{BLIND}}_{\mathsf{BSA}}(\mathcal{A}) \coloneqq 2 \cdot \left| \Pr[b = b'] - \frac{1}{2} \right|.$$

*We say that $\mathsf{BSA}$ is $(t, \varepsilon)$-blind if for all adversaries that run in time at most $t$, $\mathbf{Adv}^{\mathsf{BLIND}}_{\mathsf{BSA}}(\mathcal{A}) \leq \varepsilon$.*

**Definition 12 ((strong) OMUF).** *For a blind signature with attributes* $\mathsf{BS} = (\mathcal{G}, \mathsf{S} = (\mathcal{S}_1, \mathcal{S}_2), \mathsf{U} = (\mathcal{U}_1, \mathcal{U}_2), \mathcal{V})$*, public parameters* $\mathsf{pp}$*, a positive integer* $\ell \in \mathbb{Z}^+$*, and an adversary* $\mathcal{A}$*, we define Game (strong)* $\ell$-$\mathsf{OMUF}$ *as follows:*

1. *Generate key pair* $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathcal{G}(\mathsf{pp})$.
2. *Invoke* $\mathcal{A}^{\mathsf{S}(\mathsf{sk},.)}(\mathsf{pk})$. $\mathcal{A}$ *may query its oracle* $\mathsf{S}(\mathsf{sk},.)$ *arbitrarily and in an interleaved fashion as long as it completes at most* $\ell$ *sessions with* $\mathsf{S}$.
3. $\mathcal{A}$ *outputs* $\ell+1$ *pairwise-distinct tuples* $(\sigma_1, m_1), \ldots, (\sigma_{\ell+1}, m_{\ell+1})$ *(in the case of strong OMUF). In the case of weak or unique-message OMUF we require additionally that* $m_i \neq m_j$ *for* $i \neq j$.
4. *Output 1 if for all* $i \in [\ell + 1]$, $\mathcal{V}(\mathsf{pk}, m_i, \sigma_i) = 1$.

*We define* $\mathcal{A}$*'s advantage in winning the game* $\ell$-$\mathsf{OMUF}$ *against a blind signature scheme with attributes* $\mathsf{BSA}$ *as*

$$\mathbf{Adv}^{\ell\text{-}\mathsf{OMUF}}_{\mathsf{BSA}}(\mathcal{A}) \coloneqq \Pr[\ell\text{-}\mathsf{OMUF}^{\mathcal{A}}_{\mathsf{BSA}} = 1].$$

*We say that* $\mathsf{BSA}$ *is* $(t, \varepsilon, \ell)$*-(strong-)*$\mathsf{OMUF}$*-secure if for all adversaries* $\mathcal{A}$ *that run in time at most* $t$*,* $\mathbf{Adv}^{\ell\text{-}\mathsf{OMUF}}_{\mathsf{BSA}}(\mathcal{A}) \leq \varepsilon$.

### A.4 Commitments

**Definition 13 (Commitment Scheme).** *Commitment schemes* $\mathsf{CS} = (\mathcal{PG}, \mathcal{C}, \mathcal{V})$ *consist of the following algorithms:*

- *The* parameter generation algorithm $\mathcal{PG}$ *takes as input public parameters* $\mathsf{pp}$ *and outputs commitment parameters* $\mathsf{pp}_{\mathsf{CS}}$.
- *The* commit algorithm $\mathcal{C}$ *takes as input commitment parameters* $\mathsf{pp}_{\mathsf{CS}}$ *and a value* $v$ *outputs a commitment* $C$ *and an opening* $r$.
- *The* verification algorithm $\mathcal{CV}$ *takes as input commitment parameters* $\mathsf{pp}_{\mathsf{CS}}$*, a commitment* $C$*, a value* $v$ *and an opening* $r$ *outputs either* 1 *(*Accept*) or* 0 *(*Reject*).*

**Definition 14 (Hiding).** *We define Game* $\mathsf{HIDING}$ *for a commitment scheme* $\mathsf{CS}$ *and an adversary* $\mathcal{A}$ *as follows:*

1. *Run* $\mathsf{CS}.\mathcal{PG}(\mathsf{pp})$ *to obtain* $\mathsf{pp}_{\mathsf{CS}}$ *and output* $\mathsf{pp}_{\mathsf{CS}}$ *to the adversary.*
2. *The adversary outputs two values* $v_0, v_1$.
3. *Sample a bit* $b$ *uniformly at random.*
4. *Compute* $(C, r) \leftarrow_\$ \mathsf{CS}.\mathcal{C}(\mathsf{pp}_{\mathsf{CS}}, v_b)$ *and output* $C$ *to the adversary.*
5. *The adversary outputs a bit* $b'$.

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

$$\mathbf{Adv}^{\mathsf{HIDING}}_{\mathsf{CS}}(\mathcal{A}) \coloneqq 2 \cdot \left| \Pr[b = b'] - \frac{1}{2} \right|$$

*We say that* $\mathsf{CS}$ *is* $(\varepsilon, t)$-hiding *if for any adversary* $\mathcal{A}$ *that runs in time at most* $t$*, it holds that* $\mathbf{Adv}^{\mathsf{HIDING}}_{\mathsf{CS}}(\mathcal{A}) \leq \varepsilon$*. We say that* $\mathsf{CS}$ *is* perfectly hiding *if for any adversary the advantage is* 0.

**Definition 15 (Binding).** *We define the* BINDING *game for a commitment scheme* CS *and an adversary* $\mathcal{A}$ *as follows:*

1. *Run* $\mathsf{CS}.\mathcal{PG}$ *to obtain* $\mathsf{pp}_{\mathsf{CS}}$ *and output* $\mathsf{pp}_{\mathsf{CS}}$ *to the adversary.*
2. *The adversary outputs a commitment* $C$*, two values* $v_1, v_2$ *and two openings* $r_1, r_2$*.*
3. *Output* 1 *if* $v_1 \neq v_2$ *and* $\mathcal{CV}(\mathsf{pp}_{\mathsf{CS}}, v_1, r_1) = \mathcal{CV}(\mathsf{pp}_{\mathsf{CS}}, v_2, r_2) = 1$*, 0 otherwise.*

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

$$\mathbf{Adv}^{\mathsf{BINDING}}_{\mathsf{CS}}(\mathcal{A}) \coloneqq \Pr\left[\mathsf{BINDING}^{\mathcal{A}}_{\mathsf{CS}} = 1\right]$$

*and we say that* CS *is* $(t, \varepsilon)$-binding *if for any adversary* $\mathcal{A}$ *that runs in time at most* $t$ *it holds that* $\mathbf{Adv}^{\mathsf{BINDING}}_{\mathsf{CS}}(\mathcal{A}) \leq \varepsilon$*. We say that* CS *is* perfectly binding *if for any adversary the advantage is* 0*.*

### A.5 Multi-index Vector Commitment

**Definition 16 (Multi-index Vector Commitment).** *We define a Multi-index Vector Commitment scheme* $\mathsf{MiVC} \coloneqq (\mathcal{C}, \mathcal{O}, \mathcal{CV})$ *as follows.*

- *The* commit algorithm $\mathcal{C}$ *takes as input a randomness* $\boldsymbol{R}$ *and a messages vector* $\boldsymbol{L}$*, and it outputs a commitment* $C$*.*
- *The* opening algorithm $\mathcal{O}$ *takes as input a commitment* $C$*, a messages vector* $\boldsymbol{L}$*, a randomness* $\boldsymbol{R}$*, and an opening index vector* $\boldsymbol{I_{op}}$*, and it outputs an opening* $(\boldsymbol{L'}, \boldsymbol{R'})$*.*
- *The* verification algorithm $\mathcal{CV}$ *takes as input a commitment* $C$*, an opening* $(\boldsymbol{L'}, \boldsymbol{R'})$*, and an opening index vector* $\boldsymbol{I_{op}}$*, and it outputs either* 1 (Accept) *or* 0 (Reject)*.*

**Definition 17 (Correctness).** *We say that a Multi-index Vector Commitment scheme* MiVC *is* correct *if for all message vectors* $\boldsymbol{L}$*, all randomness vectors* $\boldsymbol{R}$*, and all index sets* $\boldsymbol{I_{op}} \subseteq [\nu]$*:*

$$\Pr\left[\mathcal{CV}(C, \boldsymbol{L'}, \boldsymbol{R'}, \boldsymbol{I_{op}}) = 1 \;\middle|\; \begin{array}{c} C \leftarrow \mathcal{C}(\boldsymbol{L}, \boldsymbol{R}) \\ (\boldsymbol{L'}, \boldsymbol{R'}) \leftarrow \mathcal{O}(C, \boldsymbol{L}, \boldsymbol{R}, \boldsymbol{I_{op}}) \end{array}\right] = 1.$$

**Definition 18 (Hiding).** *We define Game* HIDING *for a Multi-index Vector Commitment scheme* $\mathsf{MiVC} \coloneqq (\mathcal{C}, \mathcal{O}, \mathcal{CV})$ *and an adversary* $\mathcal{A}$ *as follows:*

1. *The adversary outputs two message vectors* $\boldsymbol{L}^{(0)}, \boldsymbol{L}^{(1)}$ *and a target set of opening indices* $\boldsymbol{I_{op}}$*.*
2. *If there exists* $j \in \boldsymbol{I_{op}}$ *such that* $L^{(0)}_j \neq L^{(1)}_j$*, abort and output* 0*.*
3. *Sample a bit* $b$ *uniformly at random and a randomness vector* $\boldsymbol{R}$*.*
4. *Compute* $C \leftarrow \mathcal{C}(\boldsymbol{L}^{(b)}, \boldsymbol{R})$ *and the opening* $(\boldsymbol{L'}, \boldsymbol{R'}) \leftarrow \mathcal{O}(C, \boldsymbol{L}^{(b)}, \boldsymbol{R}, \boldsymbol{I_{op}})$*.*
5. *Output* $(C, \boldsymbol{L'}, \boldsymbol{R'})$ *to the adversary.*
6. *The adversary outputs a bit* $b'$*.*

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

$$\mathbf{Adv}^{\mathsf{HIDING}}_{\mathsf{MiVC}}(\mathcal{A}) \coloneqq 2 \cdot \left| \Pr[b = b'] - \frac{1}{2} \right|$$

*We say that* MiVC *is* $(\varepsilon, t)$-hiding *if for any adversary $\mathcal{A}$ that runs in time at most $t$, it holds that* $\mathbf{Adv}^{\mathsf{HIDING}}_{\mathsf{MiVC}}(\mathcal{A}) \leq \varepsilon$.

**Definition 19 (Position Binding).** *We define Game* vcBINDING *for a Multi-index Vector Commitment scheme* $\mathsf{MiVC} \coloneqq (\mathcal{C}, \mathcal{O}, \mathcal{CV})$ *and an adversary $\mathcal{A}$ as follows:*

1. *The adversary outputs a commitment $C$, two opening pairs $(\boldsymbol{L}^{(0)}, \boldsymbol{R}^{(0)})$ and $(\boldsymbol{L}^{(1)}, \boldsymbol{R}^{(1)})$, an opening index vector $\boldsymbol{I_{op}}$, and a target index $i \in \boldsymbol{I_{op}}$.*
2. *Output* 1 *if:*
   a. $L_i^{(0)} \neq L_i^{(1)}$,
   b. $\mathcal{CV}(C, \boldsymbol{L}^{(0)}, \boldsymbol{R}^{(0)}, \boldsymbol{I_{op}}) = 1$, *and*
   c. $\mathcal{CV}(C, \boldsymbol{L}^{(1)}, \boldsymbol{R}^{(1)}, \boldsymbol{I_{op}}) = 1$.

   *Otherwise, output* 0.

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

$$\mathbf{Adv}^{\mathsf{vcBINDING}}_{\mathsf{MiVC}}(\mathcal{A}) \coloneqq \Pr\left[\mathsf{vcBINDING}^{\mathcal{A}}_{\mathsf{MiVC}} = 1\right]$$

*We say that* MiVC *is* $(t, \varepsilon)$-position binding *if for any adversary $\mathcal{A}$ that runs in time at most $t$ it holds that* $\mathbf{Adv}^{\mathsf{vcBINDING}}_{\mathsf{MiVC}}(\mathcal{A}) \leq \varepsilon$.

### A.6 Properties of Pedersen Commitments

**Construction 2 (Generalized Pedersen Commitment [54]).** We describe the generalized Pedersen Commitment Scheme GPC over a group $\mathbb{G}$ described by group parameters $\mathsf{pp}_{\mathbb{G}}$ (which take the role of the public parameters $\mathsf{pp}$) for vectors $\boldsymbol{v}$ of length $\nu$ below:

$\mathcal{PG}(\mathsf{pp}_{\mathbb{G}})$ : sample $\nu + 1$ group elements $h_0, \ldots, h_n \leftarrow_{\$} \mathbb{G}$. Output those group elements as the commitment parameters $\mathsf{pp}_{\mathsf{GPC}} = (h_0, \ldots, h_n)$.

$\mathcal{C}(\mathsf{pp}_{\mathsf{GPC}}, \boldsymbol{v})$: Given commitment parameters as above and a vector $\boldsymbol{v} \in \mathbb{Z}_p^{\nu}$ with $\nu$ entries, sample an exponent $r \leftarrow_{\$} \mathbb{Z}_p$ and output $C = h_0^r \prod_{i=1}^{\nu} h_i^{v_i}$ and $r$.

$\mathcal{CV}(\mathsf{pp}_{\mathsf{GPC}}, v, C, r)$: Given the commitment parameters as above, a value $v$, a commitment $C$, and an opening $r$, output 1 iff $C = h_0^r \prod_{i=1}^{\nu} h_i^{v_i}$.

**Construction 3 (PoK for Opening of a Generalized Pedersen Commitment).** We describe a protocol for proving knowledge of a full opening of GPC as described in construction 2, i.e. for the relation $\mathcal{R}_{\boldsymbol{h}} = \{(C, (r, \boldsymbol{v})) \colon h_0^r \cdot \prod_{i=1}^{\nu} h_i^{v_i} = C\}$,

– The *prover algorithm* $\mathcal{P}$ is split into two algorithms $\mathcal{P}_1$ and $\mathcal{P}_2$.

- $\mathcal{P}_1(\boldsymbol{h}, C, (r, \boldsymbol{v}))$: Sample $r_0, \ldots, r_n \leftarrow_\$ \mathbb{Z}_p$. Compute $R = \prod_{i=0}^{n} h_i^{r_i}$. Output the first flow $R$ and keep the local state $r_0, \ldots, r_n$.
- $\mathcal{P}_2(r_0, \ldots, r_n, c, \boldsymbol{v})$: Compute $s_0 = r_0 - c \cdot r$, for $i = 1, \ldots, n$ compute $s_i = r_i - c \cdot v_i$. Output $s_0, \ldots, s_n$.

– The *verifier algorithm* $\mathcal{V}$ is split into two algorithms $\mathcal{V}_1$ and $\mathcal{V}_2$:
- $\mathcal{V}_1(\boldsymbol{h}, C, R)$: Sample a random challenge $c \leftarrow_\$ \mathbb{Z}_p$, output $c$, keep local state $\boldsymbol{h}, C, R$.
- $\mathcal{V}_2(\boldsymbol{h}, C, R, c, \boldsymbol{s})$: Output 1 if $R = C^c \cdot \prod_{i=0}^{n} h_i^{s_i}$, 0 otherwise.

**Lemma 2 (Pedersen Commitments are perfectly hiding).** *The commitment scheme* GPC *from construction 2 is perfectly hiding.*

*Proof.* Information-theoretically, for each vector of commitment parameters $\boldsymbol{h} \in \mathbb{G}^{\nu+1}$, each commitment $C \in \mathbb{G}$ and each value vector $v \in \mathbb{Z}_p^\nu$, there exists exactly one opening $r = \mathrm{dlog}_{h_0} \frac{C}{\prod_{i=1}^{\nu} h_i^{v_i}}$. The commitment thus perfectly hides the vector $v$ and even an unbounded adversary cannot do better than guessing.

**Lemma 3 (Pedersen Commitments are binding under DL).** *If the* DLP *is $(t, \varepsilon)$-hard in the group $\mathbb{G}$,* GPC *from construction 2 is $(t', \varepsilon')$-binding with $t' \approx t$ and $\varepsilon' \leq \nu \cdot \varepsilon$.*

*Proof.* We provide a reduction $\mathcal{B}$ against DLP that uses a binding adversary $\mathcal{A}$ to break DLP. The reduction takes as input the discrete logarithm challenge $g, Y$. It samples $j \leftarrow_\$ [\nu]$ and sets $h_j := Y$. For each $i \neq j \in [\nu]_0$ it samples $a_i \leftarrow_\$ \mathbb{Z}_p$ and sets $h_i := g^{a_i}$. It outputs the commitment parameters $\mathsf{pp}_{\mathsf{GPC}} = (h_0, \ldots, h_\nu)$ to the adversary $\mathcal{A}$. When the adversary outputs a commitment $C$ and two vectors $\boldsymbol{v}_1, \boldsymbol{v}_2$ along with openings $r_1, r_2$, the reduction aborts if $v_{1,j} = v_{2,j}$. As $\boldsymbol{v}_1 \neq \boldsymbol{v}_2$, there must be at least one index $i^*$ where $v_{1,i^*} \neq v_{2,i^*}$ and $j = i^*$ happens with probability $\frac{1}{\nu}$. Thus, the reduction aborts with probability at most $1 - \frac{1}{\nu}$. If the reduction did not abort in the previous step, it computes $\mathrm{dlog} Y = \frac{r_1 \cdot a_0 + \sum_{i \neq j} a_i \cdot v_{1,i} - (r_2 \cdot a_0 + \sum_{i \neq j} a_i \cdot v_{2,i})}{v_{2,j} - v_{1,j}}$ and outputs the solution.

### A.7 Sigma-Protocols and Proofs

We recall definition of $\Sigma$-protocols and some of their security properties, see also for example [25]

**Construction 4 ($\Sigma$-Protocol).** A $\Sigma$-Protocol for a relation $\mathcal{R}$ consists of the following algorithms $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2), \mathcal{V} = (\mathcal{V}_1, \mathcal{V}_2)$.

$\mathcal{P}_1(x, w)$ Outputs a message $R$ and saves an internal state $s_\mathcal{P}$
$\mathcal{V}_1(x, R)$ Takes as input the statement and the prover's first message and outputs a challenge $c \leftarrow_\$ \mathcal{C}$.
$\mathcal{P}_2(x, w, s_\mathcal{P}, c)$ Outputs a response $S$.
$\mathcal{V}_2(x, R, c, S)$ Outputs a bit $b$ indicating acceptance or rejection.

**Definition 20 ($\Sigma$-Protocol Correctness).** *A $\Sigma$-Protocol is* correct *if for all $(x, w) \in \mathcal{R}$, all $(R, s_{\mathcal{P}}) \leftarrow_{\$} \mathcal{P}_1(x, w)$, all $c \in \mathcal{C}$, and all $S \leftarrow_{\$} \mathcal{P}_2(x, w, s_{\mathcal{P}}, c)$ , it holds that $\mathcal{V}_2(x, R, c, S) = 1$*

**Definition 21.** *(HVZK) A $\Sigma$-Protocol is* honest-verifier zero-knowledge *(HVZK) if there exists a simulator $\mathcal{SIM}$ such that for all $x \in \mathcal{L}$ and all $c \in \mathcal{C}$, the distributions of $(R, c, S) \leftarrow_{\$} \mathcal{SIM}(x, c)$ are indistinguishable from $R' \leftarrow_{\$} \mathcal{P}_1(x, w), c, S \leftarrow_{\$} \mathcal{P}_2(x, w, s_{\mathcal{P}}, c)$ conditioned on $c$ being used as the challenge.*

**Definition 22 (k-Special-Soundness).** *A $\Sigma$-Protocol is* k-special sound *if there exists a deterministic extractor $\mathcal{E}$ such that given $k$ valid transcripts $\{(R, c_i, S_i)\}_{i \in [k]}$ for statement $x$ with pairwise distinct challenges $c_i$, outputs a witness $w$ such that $(x, w) \in \mathcal{R}$.*

**Definition 23 ($\varepsilon$-Soundness).** *A $\Sigma$-Protocol is* $\varepsilon$-sound *if for any $x \notin \mathcal{L}$, any adversary $\mathcal{P}^*$ it holds that the probability that $\mathcal{V}_2(x, R, c, S) = 1$ is at most $\varepsilon$. Where $(R, s_{\mathcal{P}}) \leftarrow_{\$} \mathcal{P}_1^*(x)$, $c \leftarrow_{\$} \mathcal{V}_1(x, R)$, and $S \leftarrow_{\$} \mathcal{P}_2(x, w, s_{\mathcal{P}}, c)$ and the probability is taken over the random coins of the $\mathcal{P}^*$ and $\mathcal{V}_1(x, R)$.*

**Construction 5 (Fiat Shamir Transform [32]).** The Fiat-Shamir transform of a $\Sigma$-protocol $\Sigma = (\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2), \mathcal{V} = (\mathcal{V}_1, \mathcal{V}_2))$ with challenge space $\mathcal{C}$ and a hash function $\mathsf{H}\colon \{0,1\}^* \to \mathcal{C}$ and some auxiliary information that might be used in surrounding protocols is described below.

$\mathcal{P}(x, w)$ Compute $(R, s_{\mathcal{P}}) \leftarrow_{\$} \mathcal{P}_1(x, w)$. Compute $c \coloneqq \mathsf{H}(x, R, aux)$ for potential auxiliary information $aux$. Compute $S \leftarrow_{\$} \mathcal{P}_2(x, w, s_{\mathcal{P}}, c)$. Output $\pi = (R, S)$

$\mathcal{V}(x, \pi = (R, S), aux)$ Compute $c = \mathsf{H}(x, R, aux)$, output $\mathcal{V}_2(x, R, c, S)$.

**Lemma 4.** *Let $\Sigma_1$ be a $\Sigma$-protocol for language $\mathcal{L}_1$ with relation $\mathcal{R}_1$ and $\Sigma_2$ be a $\Sigma$-protocol for language $\mathcal{L}_2$ with relation $\mathcal{R}_2$ where $\Sigma_1$ and $\Sigma_2$ have the same challenge space $\mathcal{C}$ and both fulfill k-Special-Soundness and HVZK as above. Then, the following protocol is a $\Sigma$-protocol for the language $\mathcal{L}_1 \times \mathcal{L}_2$ with relation $\mathcal{R} = \{((x_1, x_2, ), (w_1, w_2))\colon (x_1, w_1) \in \mathcal{R}_1, (x_2, w_2) \in \mathcal{R}_2\}$ which fulfills k-Special-Soundness and HVZK.*

$\mathcal{P}_1$ *compute* $(R_1, s_1) \leftarrow_{\$} \Sigma_1.\mathcal{P}_1(x_1, w_1), (R_2, s_2) \leftarrow_{\$} \Sigma_2.\mathcal{P}_2$
$\mathcal{V}_1$ *sample* $c \leftarrow_{\$} \mathcal{C}$
$\mathcal{P}_2$ *compute* $S_1 \leftarrow_{\$} \Sigma_1.\mathcal{P}_2(x_1, w_1, s_1, c), S_2 \leftarrow_{\$} \Sigma_2.\mathcal{P}_2(x_2, w_2, s_2, c)$
$\mathcal{V}_2(x_1, x_2, R_1, R_2, c, S_1, S_2)$ *run* $\Sigma_1.\mathcal{V}_2(x_1, R_1, c, S_1)$ *and* $\Sigma_2.\mathcal{V}_2(x_2, R_2, c, S_2)$ *and output* 1 *iff both output* 1.

*Proof.* For HVZK, given $c, x_1, x_2$ run $(R_1, S_1) \leftarrow_{\$} \mathcal{SIM}_1(x_1, c)$ and $(R_2, S_2) \leftarrow_{\$} \mathcal{SIM}_2(x_2, c)$ where $\mathcal{SIM}_1$ and $\mathcal{SIM}_2$ are the HVZK simulators for $\Sigma_1$ and $\Sigma_2$, respectively.

For $k$-special soundness, given $k$ transcripts with the same first flow $R_1, R_2$ and $k$ different challenges $c_i$ and corresponding responses $S_{1,i}, S_{2,i}$, run $\mathcal{E}_1$ on $\{(R_1, c_i, S_{1,i}) : i \in [k]\}$ and $\mathcal{E}_2$ on $\{(R_2, c_i, S_{2,i}) : i \in [k]\}$ to obtain the witnesses $w_1$ and $w_2$ respectively.

### A.8 Identification Scheme from ACL ($\mathsf{ID}_{\mathsf{ACL}}$)

In the following, we describe the identification scheme $\mathsf{ID}_{\mathsf{ACL}}$, which was proposed by [41]. The scheme is defined as tuple $(\mathcal{G}, \mathcal{P}, \mathcal{V})$, where

- $\mathcal{G}(\mathsf{pp})$: Parse $\mathsf{pp}$ to obtain $\mathbb{G}, p, g$, sample $h \leftarrow_\$ \mathbb{G}$ and $x, v_{0,1}, v_{0,2} \leftarrow_\$ \mathbb{Z}_p$, compute $y = g^x, z \coloneqq g^{v_{0,1}} \cdot h^{v_{0,2}}$, set $\mathsf{sk} = x$ and $\mathsf{pk} = (h, z, y)$, and output $(\mathsf{pk}, \mathsf{sk})$.
- The prover algorithms $\mathcal{P}_=(\mathcal{P}_1, \mathcal{P}_2)$ are defined as follows:
  - $\mathcal{P}_1(\mathsf{sk})$: Sample $d, s_1, s_2, u, \mathrm{rnd} \leftarrow_\$ \mathbb{Z}_p$, compute $z_1 \coloneqq g^{\mathrm{rnd}}, z_2 \coloneqq z/z_1, a \coloneqq g^u, b_1 \coloneqq g^{s_1} \cdot z_1^d, b_2 \coloneqq h^{s_2} \cdot z_2^d$, set $R \coloneqq (a, b_1, b_2, \mathrm{rnd}), st_p \coloneqq (d, s_1, s_2, u)$ and output $(R, st_p)$.
  - $\mathcal{P}_2(\mathsf{sk}, st_p, R, \varepsilon)$: Parse $st_p$ as $(d, s_1, s_2, u)$, compute $c \coloneqq \varepsilon - d, r \coloneqq u - c \cdot \mathsf{sk}$, set and output $S \coloneqq (c, d, r, s_1, s_2)$.
- The verifier algorithms $\mathcal{V}=(\mathcal{V}_1, \mathcal{V}_2)$ are defined as follows:
  - $\mathcal{V}_1(\mathsf{pk}, R)$: Parse $\mathsf{pk}$, and $R = (a, b_1, b_2, \mathrm{rnd})$. Output $\perp$ if $\mathrm{rnd} = 0$. Sample $\varepsilon \leftarrow_\$ \mathbb{Z}_p$, set $st_v \coloneqq (a, b_1, b_2, z_1 \coloneqq g^{\mathrm{rnd}}, z_2 \coloneqq z/z_1)$, and output $(\varepsilon, st_v)$.
  - $\mathcal{V}_2(\mathsf{pk}, st_v, R, \varepsilon, S)$: Parse $S$ and $st_v$, and output 1 if the following condition holds: $\varepsilon = c + d \wedge a = g^r \cdot y^c \wedge b_1 = g^{s_1} \cdot z_1^d \wedge b_2 = h^{s_2} \cdot z_2^d$. Otherwise, output 0.

### A.9 Merkle Tree-based Multi-index Vector Commitment

#### Correctness of Merkle Tree-based Multi-index Vector Commitment

**Lemma 5 (Correctness).** *Our construction in 3.1 is correct.*

*Proof.* We need to show that $\mathcal{C}(\boldsymbol{L}; \boldsymbol{R}) = \mathcal{C}(\boldsymbol{L'}; \boldsymbol{R'})$.

First, the verification checks that for all $i \in \boldsymbol{I_{op}}$, $R'_i \neq \perp$. Since the opening algorithm sets $R'_i = R_i$ for all $i \in \boldsymbol{I_{op}}$, and valid randomness implies $R_i \neq \perp$, this check always passes.

We note that the commitment function attempts to reconstruct the leaves set $S = \{s_1, \ldots, s_\nu\}$, and constructs the Merkle tree of these leaves. Therefore, it is sufficient to show that the execution of $\mathcal{C}(\boldsymbol{L}; \boldsymbol{R}) = \mathcal{C}(\boldsymbol{L'}; \boldsymbol{R'})$ results in the same leaves set $S$.

For all $i \in [\nu]$, we note that $s_i = \mathsf{H}_{\mathsf{MT}}(L_i; R_i)$ in the execution of $\mathcal{C}(\boldsymbol{L}; \boldsymbol{R})$. But we distinguish two cases in the execution of $\mathcal{C}(\boldsymbol{L'}; \boldsymbol{R'})$:

1. $i \in \boldsymbol{I_{op}}$. In this case it holds that $L'_i = L_i$ and $R_i = R_i$. Consequently, we have $s_i = \mathsf{H}_{\mathsf{MT}}(L'_i; R'_i)$.
2. $i \notin \boldsymbol{I_{op}}$. In this case, we have $R'_i = \perp$ and $L'_i = \mathsf{H}_{\mathsf{MT}}(L_i; R_i)$ (see $\mathcal{O}$). Again, we have $s_i = L'_i = \mathsf{H}_{\mathsf{MT}}(L_i; R_i)$.

In both cases, the leaves set $S$ is identical in the execution of $\mathcal{C}(\boldsymbol{L}; \boldsymbol{R})$ and $\mathcal{C}(\boldsymbol{L'}; \boldsymbol{R'})$, hence it holds $\mathcal{C}(\boldsymbol{L'}; \boldsymbol{R'}) = C$, hence $\mathcal{CV}(C, \boldsymbol{L'}, \boldsymbol{R'}, \boldsymbol{I_{op}}) = 1$.

**Lemma 6 (Hiding).** *In the random oracle model, the commitment scheme* $\mathsf{MT}$ *described in Section 3.1 with a random oracle* $\mathsf{H}_{\mathsf{MT}}$ *is Hiding. Concretely, for all positive integers* $\nu$ *and all adversaries* $\mathcal{A}$ *making at most* $\mathsf{Q}_{\mathsf{MT}}$ *queries to* $\mathsf{H}_{\mathsf{MT}}$*, it holds*

$$\mathbf{Adv}^{\mathsf{HIDING}}_{\mathsf{MT}}(\mathcal{A}) \leq \frac{2 \cdot \mathsf{Q}_{\mathsf{MT}}}{2^{\lambda}}.$$

*Proof.* Let $\mathcal{A}$ be an adversary running Game $\mathsf{HIDING}^{\mathcal{A}}_{\mathsf{MT}}$. $\mathcal{A}$ submits two vectors $\boldsymbol{L}^{(0)}$ and $\boldsymbol{L}^{(1)}$ of length $\nu$ along with a vector $\boldsymbol{I_{op}}$ of indices. Then $\mathcal{A}$ is given $C \leftarrow_{\$} \mathcal{C}(\boldsymbol{L}^{(b)}, \boldsymbol{R})$ along with an opening $(\boldsymbol{L'}, \boldsymbol{R'}) \leftarrow_{\$} \mathcal{O}(C, \boldsymbol{L}^{(b)}, \boldsymbol{R}, \boldsymbol{I_{op}})$ for uniformly random $b \leftarrow_{\$} \{0,1\}$ and $\boldsymbol{R} \leftarrow_{\$} (\{0,1\}^{\lambda})^{\nu}$ and it outputs $b'$.

We use a simple hybrid argument.

*Game* $\mathbf{G}_1$. This is Game $\mathsf{HIDING}^{\mathcal{A}}_{\mathsf{MT}}$.

*Game* $\mathbf{G}_2$. This game is identical to $\mathbf{G}_1$, except that it aborts if $\mathcal{A}$ queries $\mathsf{H}_{\mathsf{MT}}$ on $(L_i^{(b)}, R_i)$ for some $i \notin \boldsymbol{I_{op}}$.

*Claim 2.*

$$|\Pr[\mathbf{G}_2 = 1] - \Pr[\mathbf{G}_1 = 1]| \leq \mathsf{Q}_{\mathsf{MT}} \cdot 2^{-\lambda}.$$

*Proof.* Since $R_i$ is chosen uniformly at random and is hidden from $\mathcal{A}$, the probability of $\mathcal{A}$ hitting the query $(L_i^{(b)}, R_i)$ is at most $2^{-\lambda}$ for a fixed query. Taking the union bound over all $\mathsf{Q}_{\mathsf{MT}}$ queries $\mathcal{A}$ makes to $\mathsf{H}_{\mathsf{MT}}$, the probability that $\mathbf{G}_2$ aborts is at most $\mathsf{Q}_{\mathsf{MT}} \cdot 2^{-\lambda}$.

Now we analyze the advantage of $\mathcal{A}$ in winning $\mathbf{G}_2$.

We distinguish two types of vector positions:

*Case 1.* Opened positions $i \in \boldsymbol{I_{op}}$: These positions are visible in the opening $(\boldsymbol{L'}, \boldsymbol{R'})$ $\mathcal{A}$ receives. However, since the winning conditions enforce both vectors to be identical at the opened positions, these positions do not help the adversary in guessing $b$.

*Case 2.* Unopened positions $i \notin \boldsymbol{I_{op}}$: For each position, $\mathcal{A}$ receives $L'_i = \mathsf{H}_{\mathsf{MT}}(L_i^{(b)}, R_i)$ and $R'_i = \perp$. In this case, the value $L'_i$ is uniformly random in $\mathcal{A}$'s view, as long as it does not query $\mathsf{H}_{\mathsf{MT}}$ on $(L_i^{(b)}, R_i)$. Due to the abort condition in $\mathbf{G}_2$, $\mathcal{A}$ does not query $\mathsf{H}_{\mathsf{MT}}(L_i^{(b)}, R_i)$. Consequently, as $L'_i$ is uniformly random and independent of $b$ in $\mathcal{A}$'s view, it outputs the correct $b$ with probability $\frac{1}{2}$.

Overall, we have:

$$\mathbf{Adv}^{\mathsf{HIDING}}_{\mathsf{MT}}(\mathcal{A}) = 2 \cdot \left| \Pr[\mathbf{G}_1 = 1] - \frac{1}{2} \right| \leq 2 \cdot \left( \left| \frac{1}{2} - \frac{1}{2} \right| + \mathsf{Q}_{\mathsf{MT}} \cdot 2^{-\lambda} \right) = \frac{2 \cdot \mathsf{Q}_{\mathsf{MT}}}{2^{\lambda}}.$$

**Binding of Merkle Tree-based Multi-index Vector Commitment**

**Lemma 7 (Binding).** *In the random oracle model, the commitment scheme* $\mathsf{MT}$ *described in Section 3.1 with a random oracle* $\mathsf{H}_{\mathsf{MT}}$ *is position binding. Let* $\mathsf{Q}_{\mathsf{MT}}$ *be an upper bound for the number of queries* $\mathcal{A}$ *makes to* $\mathsf{H}_{\mathsf{MT}}$*. Then it holds*

$$\mathbf{Adv}^{\mathsf{BINDING}}_{\mathsf{MT}}(\mathcal{A}) \leq \frac{\mathsf{Q}_{\mathsf{MT}} \cdot (\mathsf{Q}_{\mathsf{MT}} - 1)}{2^{\lambda+1}}.$$

*Proof.* Let $\mathcal{A}$ be an adversary running game $\mathsf{vcBINDING}^{\mathcal{A}}_{\mathsf{MT}}$ that outputs a commitment $C$, a target index $i \in [\nu]$, and two pairs $(\boldsymbol{L}^{(0)}, \boldsymbol{R}^{(0)})$ and $(\boldsymbol{L}^{(1)}, \boldsymbol{R}^{(1)})$ such that $L^{(0)}_i \neq L^{(1)}_i$, $\mathcal{CV}(C, \boldsymbol{L}^{(0)}, \boldsymbol{R}^{(0)}, \boldsymbol{I_{op}}) = \mathcal{CV}(C, \boldsymbol{L}^{(1)}, \boldsymbol{R}^{(1)}, \boldsymbol{I_{op}}) = 1$.

Since both openings open $C$ correctly, it holds $C = C_0 = C_1$, where $C_0 \coloneqq \mathcal{C}(\boldsymbol{L}^{(0)}; \boldsymbol{R}^{(0)})$ and $C_1 \coloneqq \mathcal{C}(\boldsymbol{L}^{(1)}; \boldsymbol{R}^{(1)})$. However, the roots $C_0$ and $C_1$ are identical iff the leaves are identical or a collision occurs inside the tree.

We proceed via a series of games.

*Game* $\mathbf{G}_1$. This is Game $\mathsf{vcBINDING}^{\mathcal{A}}_{\mathsf{MT}}$.

*Game* $\mathbf{G}_2$. This game is identical to $\mathbf{G}_1$, except that it aborts if $\mathcal{A}$ finds a collisions $(Q_1, Q_2)$ for $\mathsf{H}_{\mathsf{MT}}$, i.e. $Q_1 \neq Q_2$ with $\mathsf{H}_{\mathsf{MT}}(Q_1) = \mathsf{H}_{\mathsf{MT}}(Q_2)$.

*Claim 3.*

$$\mathbf{Adv}^{\mathbf{G}_2}(\mathcal{A}) \geq \mathbf{Adv}^{\mathbf{G}_1}(\mathcal{A}) - \frac{\mathsf{Q}_{\mathsf{MT}} \cdot (\mathsf{Q}_{\mathsf{MT}} - 1)}{2^{\lambda+1}}.$$

*Proof.* The probability that any specific pair $(Q_1, Q_2)$ collide is $\frac{1}{2^\lambda}$. Since $\mathcal{A}$ makes up to $\mathsf{Q}_{\mathsf{MT}}$ queries, the number of possible distinct pairs is $\mathsf{Q}_{\mathsf{MT}} \cdot (\mathsf{Q}_{\mathsf{MT}} - 1)/2$, taking the union bound yields that a collision occurs with probability at most $\frac{\mathsf{Q}_{\mathsf{MT}} \cdot (\mathsf{Q}_{\mathsf{MT}} - 1)}{2^{\lambda+1}}$.

Let $MT_0$ and $MT_1$ be the Merkle trees constructed from the openings $(\boldsymbol{L}^{(0)}, \boldsymbol{R}^{(0)})$ and $(\boldsymbol{L}^{(1)}, \boldsymbol{R}^{(1)})$, respectively. Let $MT_b[j]$ denote the set of the nodes at the $j$-th level of the tree $MT_b$ for $j \in [\log(\nu)]_0$. The tree levels are indexed top to bottom (the root is at $j = 0$ and the leaves are at $j = \log(\nu)$.) In the following game, we want to rule out that the trees are different on some level $j$, yet they are identical on the level above.

Let $j \in [\log(\nu)]$ be a fixed index satisfying $MT_0[j] \neq MT_1[j]$ but $MT_0[j-1] = MT_1[j-1]$. Let $S_{b,j} \coloneqq (s_{b,1}, \ldots, s_{b,t})$ be the nodes of $MT_b[j]$ for $t = 2^j$. Assume without loss of generality $S_{0,j}$ and $S_{1,j}$ differ in a single pair $(s_{b,2r-1}, s_{b,2r})$ for $r \in [t/2]$.

*Game* $\mathbf{G}_3$. The game is identical to $\mathbf{G}_2$, except that it aborts and outputs 0 if $MT_0[j] \neq MT_1[j]$ but $MT_0[j-1] = MT_1[j-1]$.

*Claim 4.*

$$|\Pr[\mathbf{G}_3 = 1] - \Pr[\mathbf{G}_2 = 1]| = 0.$$

*Proof.* Since $MT_0[j-1]$ and $MT_1[j-1]$ are identical, it follows per construction that $\mathsf{H}_{\mathsf{MT}}(s_{0,2r-1}, s_{0,2r}) = \mathsf{H}_{\mathsf{MT}}(s_{1,2r-1}, s_{1,2r})$. This, however, does not occur; otherwise, $\mathbf{G}_2$ would have aborted.

The only remaining strategy for $\mathcal{A}$ is to ensure that the leaves of both trees are identical $MT_0[\log(\nu)] = MT_1[\log(\nu)]$. However, since we know that $L_i^{(0)} \neq L_i^{(1)}$ and $R_i^{(0)} \neq \bot \wedge R_i^{(1)} \neq \bot$, it must hold due to Game $\mathbf{G}_2$ that $\mathsf{H}_{\mathsf{MT}}(L_i^{(0)}; R_i^{(0)}) \neq \mathsf{H}_{\mathsf{MT}}(L_i^{(1)}; R_i^{(1)})$.

Overall, we have $\mathbf{Adv}^{\mathsf{vcBINDING}}_{\mathsf{MT}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathbf{G}_2} + \frac{\mathsf{Q}_{\mathsf{MT}} \cdot (\mathsf{Q}_{\mathsf{MT}}-1)}{2^{\lambda+1}} = \frac{\mathsf{Q}_{\mathsf{MT}} \cdot (\mathsf{Q}_{\mathsf{MT}}-1)}{2^{\lambda+1}}$.

## B Our Transformation: Pachirisu

**Construction 6 (Pachirisu).** Let $\mathsf{BS}^{\mathsf{H}} := (\mathcal{G}^{\mathsf{H}}, \mathcal{S}, \mathcal{SIM}_{\mathcal{S}}, \mathcal{U}^{\mathsf{H}}, \mathcal{V}^{\mathsf{H}})$ be a three-move simulatable blind signature scheme with respect to a random oracle $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$, where $\lambda$ is a security parameter specified by public parameters $\mathsf{pp}$. Let $\mathsf{H}_{\mathsf{J}}, \mathsf{H}_m, \mathsf{H}_{\mathrm{rnd}}, \mathsf{H}_\mu, \mathsf{H}_{\Upsilon} : \{0,1\}^* \to \{0,1\}^\lambda$ be hash functions treated as random oracles, and let $\mathsf{MT} := (\mathcal{C}, \mathcal{O}, \mathcal{CV})$ be the commitment scheme described in Construction 1. For a positive integer $K \geq 1$, a function $f : \mathbb{N} \to \mathbb{N}$, we define $\mathsf{BS}$ as follows:

- $\mathcal{G}(\mathsf{pp})$: Generate $K$ pairs of keys $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow_{\$} \mathsf{BS}.\mathcal{G}(\lambda)$ for $i \in [K]$, and return $\{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i\in[K]}$.
- $\mathcal{RG}_{\mathcal{U}}(\boldsymbol{L})$: Set $C := \boldsymbol{L}$, $\pi := \bot$, and $\mathsf{rst}_{\mathsf{U}} := (\boldsymbol{L}, C)$, and return $(C, \mathsf{rst}_{\mathsf{U}}, \pi)$. The user sends the attribute vector $\boldsymbol{L}$ in plain to the signer instead of sending a commitment of the attributes; therefore, the proof is not required ($\pi = \bot$).
- $\mathcal{RG}_{\mathcal{S}}(C, \pi, \mathsf{rst}'_{\mathsf{S}})$: Parse $\boldsymbol{L} := C$. Update and output the state $\mathsf{rst}_{\mathsf{S}} := \mathsf{rst}'_{\mathsf{S}} \cup \boldsymbol{L}$.
- $\mathsf{S}$: The signer algorithm is split into four algorithms $\mathcal{S}_{1-4}$. $\mathcal{S}_1$ acts as stateful initialization phase. After a session is initialized via an $\mathcal{S}_1$ query, the algorithms $\mathcal{S}_{2-4}$ are executed $K$ times to conclude the session.
  - $\mathcal{S}_1(\mathsf{sk}, \mathsf{rst}_{\mathsf{S}}, C)$: If $C \notin \mathsf{rst}_{\mathsf{S}}$, abort and output $\bot$. Otherwise, increment $\mathsf{ctr} := \mathsf{ctr} + 1$, and compute $N := f(\mathsf{ctr})$. Sample $\mathsf{seed}_i \leftarrow_{\$} \{0,1\}^\lambda$ and compute $com_{\mathsf{seed}\,i} := \mathsf{H}_{\mathsf{J}}(\mathsf{seed}_i)$ for all $i \in [K]$. Define a signer state $st_{\mathcal{S}} := (\mathsf{ctr}, N, \{\mathsf{seed}_i, com_{\mathsf{seed}\,i}\}_{i\in[K]})$. Return $(st_{\mathcal{S}}, N, com_{\mathsf{seed}\,i})$.
  - $\mathcal{S}_2(\mathsf{sk}_i, st_{\mathcal{S}}, \{com_j\}_{j\in[N]})$: Parse $N$ from $st_{\mathcal{S}}$. For all $j \in [N]$, compute $(\mathsf{msgS1}_j, \mathsf{rnd}_{\mathsf{S}\,j}) \leftarrow_{\$} \mathsf{BS}.\mathcal{S}_1(\mathsf{sk}_i)$. Compute $J_i := \mathsf{H}_{\mathsf{J}}(\mathsf{seed}_i, \mathsf{pk}_i)$, update the signer state $st_{\mathcal{S}} := st_{\mathcal{S}} \cup \{J_i, \{\mathsf{rnd}_{\mathsf{S}\,j}, \mathsf{msgS1}_j, com_j\}_{j\in[N]}\}$ and return $(st_{\mathcal{S}}, \{\mathsf{msgS1}_j\}_{j\in[N]})$.
  - $\mathcal{S}_3(\mathsf{sk}_i, st_{\mathcal{S}}, \{\mathsf{msgU}_j\}_{j\in[N]})$: Parse $\mathsf{seed}_i$ from $st_{\mathcal{S}}$. Update the signer state $st_{\mathcal{S}} := st_{\mathcal{S}} \cup \{\mathsf{msgU}_j\}_{j\in[N]}$. Return $(st_{\mathcal{S}}, \mathsf{seed}_i)$.
  - $\mathcal{S}_4(\mathsf{sk}_i, st_{\mathcal{S}}, \{\boldsymbol{R}_j, \mu_j, \mathsf{Urnd}_j\}_{J_i \neq j\in[N]})$: Parse $(\{com_j, \mathsf{msgU}_j, \mathsf{rnd}_{\mathsf{S}\,j}, \mathsf{msgS1}_j\}_{j\in[N]}, J_i, C)$ from $st_{\mathcal{S}}$. For all $J_i \neq j \in [N]$, compute $\zeta_j := \mathsf{MT}.\mathcal{C}(\boldsymbol{L}; \boldsymbol{R}_j)$, $\Upsilon_j := \mathsf{H}_{\Upsilon}(\mathsf{pk}_i, \mu_j, \zeta_j)$, and abort and output $\bot$ if $com_j \neq \mathsf{H}_{\mathrm{rnd}}(\mathsf{Urnd}_j)$ or $\mathsf{msgU}_j \neq \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_i, \Upsilon_j, \mathsf{msgS1}_j; \mathsf{Urnd}_j)$. Compute and output $\mathsf{msgS2}_{J_i} := \mathsf{BS}.\mathcal{S}_2(\mathsf{sk}_i, \mathsf{rnd}_{\mathsf{S}\,J_i}, \mathsf{msgU}_{J_i})$.

  $\mathsf{S}$ outputs 1 (COMPLETE) if $\mathcal{S}_4(\mathsf{sk}_K, .)$ terminates successfully.

- $\mathsf{U}$: The user algorithm consists of four algorithms $\mathcal{U}_1$-$\mathcal{U}_4$. Upon receiving $(N, \{com_{\mathsf{seed}i}\}_{i\in[K]})$ from the signer. Let $M$ be the message being signed. The user initializes the session by setting $m_0 := M$, $\sigma_0 := \zeta_0 := \eta_0 := \bot$ and the state $st_{\mathcal{U}} := (m_0, \sigma_0, \zeta_0, \eta_0)$. Then, algorithms $\mathcal{U}_{1-4}$ are executed $K$ times sequentially.
  - $\mathcal{U}_1(\mathsf{pk}_i, st_{\mathcal{U}}, \boldsymbol{L}, N, \{com_{\mathsf{seed}i}\}_{i\in[K]})$: Parse $st_{\mathcal{U}}$ to extract $(m_{i-1}, \sigma_{i-1}, \zeta_{i-1}, \eta_{i-1})$. Compute $m_i := \mathsf{H}_m(\sigma_{i-1}, \eta_{i-1}, \zeta_{i-1}, m_{i-1})$. For $j \in [N]$, sample $\mathsf{Urnd}_j \leftarrow_{\$} \{0,1\}^\lambda$, compute $com_j := \mathsf{H}_{\mathrm{rnd}}(\mathsf{Urnd}_j)$. Update the state $st_{\mathcal{U}} := st_{\mathcal{U}} \cup \{m_i, \boldsymbol{L}, N, \{com_{\mathsf{seed}i}\}_{i\in[K]}, \{\mathsf{Urnd}_j\}_{j\in[N]}\}$. Return $(st_{\mathcal{U}}, \{com_j\}_{j\in[N]})$.
  - $\mathcal{U}_2(\mathsf{pk}_i, st_{\mathcal{U}}, \{\mathsf{msgS1}_j\}_{j\in[N]})$: Parse $(m_i, \boldsymbol{L})$ from $st_{\mathcal{U}}$. For all $j \in [N]$, do the following: Sample $\eta_j \leftarrow_{\$} \{0,1\}^\lambda, \boldsymbol{R}_j \leftarrow_{\$} (\{0,1\}^\lambda)^\nu$, compute $\mu_j := \mathsf{H}_\mu(m_i, \eta_j)$, $\zeta_j := \mathsf{MT}.\mathcal{C}(\boldsymbol{L}; \boldsymbol{R}_j)$, $\Upsilon_j := \mathsf{H}_\Upsilon(\mathsf{pk}_i, \mu_j, \zeta_j)$, and $\mathsf{msgU}_j := \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_i, \Upsilon_j, \mathsf{msgS1}_j; \mathsf{Urnd}_j)$. Update $st_{\mathcal{U}} := st_{\mathcal{U}} \cup \{\eta_j, \boldsymbol{R}_j, \mu_j, \zeta_j, \Upsilon_j, \mathsf{msgU}_j\}_{j\in[N]}$. Return $(st_{\mathcal{U}}, \{\mathsf{msgU}_j\}_{j\in[N]})$.
  - $\mathcal{U}_3(\mathsf{pk}_i, st_{\mathcal{U}}, \mathsf{seed}_i)$: Parse $(\{com_{\mathsf{seed}i}\}_{i\in[K]}, \{\mathsf{Urnd}_j, \boldsymbol{R}_j, \mu_j\}_{j\in[N]})$ from the state $st_{\mathcal{U}}$. Abort and output $\bot$ if $com_{\mathsf{seed}i} \neq \mathsf{H}_{\mathsf{J}}(\mathsf{seed}_i)$. Compute $J_i := \mathsf{H}_{\mathsf{J}}(\mathsf{seed}_i, \mathsf{pk}_i)$. Return $(st_{\mathcal{U}}, \{\boldsymbol{R}_j, \mu_j, \mathsf{Urnd}_j\}_{j\neq J_i})$.
  - $\mathcal{U}_4(\mathsf{pk}_i, st_{\mathcal{U}}, \mathsf{msgS2}_{J_i})$: Parse $(m_i, \{\mathsf{Urnd}_j, \eta_j, \zeta_j, \boldsymbol{R}_j, \Upsilon_j\}_{j\in[N]})$ from $st_{\mathcal{U}}$. Compute $\sigma_i := \mathsf{BS}.\mathcal{U}_2(\mathsf{pk}_i, \Upsilon_{J_i}, \mathsf{msgS2}_{J_i}; \mathsf{Urnd}_{J_i})$. Update $st_{\mathcal{U}} := st_{\mathcal{U}} \cup \{\sigma_i, \eta_{J_i}, \zeta_{J_i}\}$. Return a signature $(\sigma_i, \eta_{J_i}, \zeta_{J_i})$ and an opening parameter $\mathsf{ost}_{\mathcal{U}i} := \boldsymbol{R}_{J_i}$.
- $\mathcal{SH}_{\mathcal{U}}(\mathsf{pk}, \mathsf{rst}_{\mathsf{U}}, \mathsf{ost}_{\mathcal{U}}, M, \Sigma, \boldsymbol{I_{op}})$: Parse $(\boldsymbol{R}_1, \ldots, \boldsymbol{R}_K) := \mathsf{ost}_{\mathcal{U}}.\boldsymbol{R}$. For all $i \in [K]$, compute $(\boldsymbol{L}_i^*, \boldsymbol{R}_i^*) := \mathsf{MT}.\mathcal{O}(\mathsf{rst}_{\mathsf{U}}.C, \mathsf{ost}_{\mathcal{U}}.\boldsymbol{L}, \boldsymbol{R}_i, \boldsymbol{I_{op}})$. Set $\boldsymbol{L}^* := (\boldsymbol{L}_1^*, \ldots, \boldsymbol{L}_K^*)$ and $\pi := (\boldsymbol{R}_1^*, \ldots, \boldsymbol{R}_K^*)$. Return $(\boldsymbol{L}^*, \pi)$.
- $\mathcal{SH}_{\mathcal{V}}(\mathsf{pk}, M, \Sigma, \boldsymbol{L}^*, \boldsymbol{I_{op}}, \pi)$: Parse $(\sigma, \eta, \zeta) := \Sigma$, $(\zeta_1, \ldots, \zeta_K) := \zeta$, $(\boldsymbol{L}_1^*, \ldots, \boldsymbol{L}_K^*) := \boldsymbol{L}^*$, and $(\boldsymbol{R}_1^*, \ldots, \boldsymbol{R}_K^*) := \pi$. Output 0 if $\mathsf{MT}.\mathcal{CV}(\zeta_i, (\boldsymbol{L}_i^*, \boldsymbol{R}_i^*), \boldsymbol{I_{op}}) = 0$ for an $i \in [K]$. If $\mathcal{V}(\mathsf{pk}, M, \Sigma) = 0$ return 0, and 1 otherwise.
- $\mathcal{V}(\mathsf{pk}, M, \Sigma)$: Parse $(\boldsymbol{\sigma}, \boldsymbol{\eta}, \boldsymbol{\zeta}) := \Sigma$, $(\sigma_1, \ldots, \sigma_K) := \boldsymbol{\sigma}$, $(\eta_1, \ldots, \eta_K) := \boldsymbol{\eta}$, $(\zeta_1, \ldots, \zeta_K) := \boldsymbol{\zeta}$, and $(\mathsf{pk}_1, \ldots, \mathsf{pk}_K) := \mathsf{pk}$. Set $\sigma_0 := \eta_0 := \zeta_0 := \bot$ and $m_0 := M$. For $i \in [K]$, compute $m_i := \mathsf{H}_m(\sigma_{i-1}, \eta_{i-1}, \zeta_{i-1}, m_{i-1})$, $\mu_i := \mathsf{H}_\mu(m_i, \eta_i)$, and $\Upsilon_i := \mathsf{H}_\Upsilon(\mathsf{pk}_i, \mu_i, \zeta_i)$. If $\mathsf{BS}.\mathcal{V}(\mathsf{pk}_i, \Upsilon_i, \sigma_i) = 1$ for all $i \in [K]$, return 1, and 0 otherwise.

# C Deferred Full Proofs for Pachirisu

## C.1 OMUF

**Theorem 1** ($\ell$-OMUF). *Let $\mathsf{H}, \mathsf{H}_{\mathsf{ro}} : \{0,1\}^* \to \{0,1\}^\lambda$ for $\mathsf{ro} \in RO := \{\Upsilon, m, \mu, rnd, \mathsf{MT}\}$ be hash functions treated as random oracles. Let $\mathsf{BS}^{\mathsf{H}}$ be an arbitrary three-move simulatable blind signature scheme with the properties described in Definition 4, and let $\mathbf{P}[\mathsf{BS}^{\mathsf{H}}]$ be the scheme resulting from applying our transform on $\mathsf{BS}^{\mathsf{H}}$. Assume there exists $\vartheta > 0$ and $f : \mathbb{N} \to \mathbb{N}$ with $f(\mathsf{ctr}) = \lceil 3 \cdot \vartheta \cdot \ln(\ell+1) \cdot \mathsf{ctr} \rceil$. Then for all adversaries $\mathcal{A}$ making at most $\mathsf{Q}_{\mathsf{H}}, \mathsf{Q}_{\mathsf{ro}}$ queries to the oracles $\mathsf{H}$ and $\mathsf{H}_{\mathsf{ro}}$, respectively, and $\mathsf{Q}_{\mathcal{S}_2}$ and $\mathsf{Q}_{\mathcal{S}_3}$ queries to $\mathbf{P}[\mathsf{BS}^{\mathsf{H}}].\mathcal{S}_2$ and $\mathbf{P}[\mathsf{BS}^{\mathsf{H}}].\mathcal{S}_3$,*

*respectively, and for all public parameters* $\mathsf{pp}$ *with a security parameter* $\lambda$*, and all non-negative integers* $\ell$*, there exists* $\delta > 0$ *with* $(1-\delta)\cdot\vartheta > 1$*, and a reduction* $\mathcal{R}_3$ *such that if* $\mathsf{BS}$ *is* $(t,\varepsilon)$*-0-*$\mathsf{OMUF}$*, then* $\mathbf{P}[\mathsf{BS}]$ *is* $(t',\varepsilon')$*-*$\ell$*-*$\mathsf{OMUF}$ *with* $t' \approx t$ *and*

$$\varepsilon' \leq K \cdot \mathsf{Q}_{\mathsf{H}} \cdot \varepsilon + \epsilon_{cheat} + \epsilon_{ro} + \epsilon_{bs}.$$

*where*

$$\epsilon_{ro} \coloneqq \frac{\sum_{\mathsf{ro}\in RO} \mathsf{Q}_{\mathsf{ro}} \cdot (\mathsf{Q}_{\mathsf{ro}} - 1)}{2^{\lambda+1}} + \frac{N \cdot \mathsf{Q}_{\mathcal{S}_2} \cdot \mathsf{Q}_{rnd}}{2^{\lambda}} + \frac{N \cdot \mathsf{Q}_{\mathcal{S}_3} \cdot \mathsf{Q}_{\Upsilon}}{2^{\lambda}},$$
$$\epsilon_{cheat} \coloneqq \exp(-\delta \cdot K),$$
$$\epsilon_{bs} \coloneqq \varepsilon_{\mathsf{SHL}} \cdot K \cdot (\ell+1) + \ell \cdot K \cdot (N-1) \cdot (\varepsilon_{\mathsf{UUF}} + \varepsilon_{\mathsf{UUO}}) + K \cdot \mathsf{Q}_{\mathsf{H}} \cdot (\varepsilon_{\mathsf{UL}} + \varepsilon_{\mathsf{DS}}).$$

*Simplifying Notation.* Once $\mathcal{A}$ queries oracle $\mathcal{S}_1$, it starts a new signing session $\mathsf{ctr}$. A session $\mathsf{ctr}$ consists of $K$ sequential subsessions $\mathsf{ctr}_1, \ldots, \mathsf{ctr}_K$, where each subsession $(\mathsf{ctr}, i)$ is a sequence of $\mathcal{S}_2 \to \mathcal{S}_3 \to \mathcal{S}_4$ calls with respect to the secret key $\mathsf{sk}_i$. A session $\mathsf{ctr}$ is successfully closed after all subsessions are completed successfully, that is, a $\mathcal{S}_4$ query for the $K$-th subsession closes the respective session. Each valid message-signature pair $(M, \Sigma)$ consists of $K$ nested message-subsignature pairs $(\Upsilon_1, \sigma_1), \ldots, (\Upsilon_K, \sigma_K)$ (see $\mathcal{V}$ of $\mathbf{P}[\mathsf{BS}]$), which are valid $\mathsf{BS}$ message-signature pairs.

We say that a $\mathsf{BS}$ signature $\Sigma$ on a message $\Upsilon$ is linked to a hash query $\mathsf{H}(Q)$ if the verification procedure $\mathsf{BS}.\mathcal{V}$ makes the same query $Q$ to $\mathsf{H}$ (in the sense of Property 2).

*Proof.* We proceed via a sequence of games to rule out some bad events.

*Game* $\mathbf{G}_1$*.* This is Game $\ell$-$\mathsf{OMUF}^{\mathcal{A}}_{\mathbf{P}[\mathsf{BS}]}$.

*Game* $\mathbf{G}_2$*.* This game is identical to $\mathbf{G}_1$, except that it aborts and outputs 0 if $\mathcal{A}$ provokes Event $\mathbf{BAD}_1$, which is defined as follows:

$\mathbf{BAD}_1$: *$\mathcal{A}$ makes a $\mathcal{S}_2$ query for some signing session* $\mathsf{ctr}$ *with a value* $com_j$ *while* $\mathsf{T}_{rnd}[com_j] = \perp$*, and later $\mathcal{A}$ finds a preimage* $\mathsf{Urnd}_j$ *for* $com_j$ *satisfying* $\mathsf{H}_{rnd}(\mathsf{Urnd}_j) = com_j$*. We modify the way the random oracle* $\mathsf{H}_{rnd}$ *is simulated to abort if the adversary submits a fresh query* $Q$*,* $\mathsf{H}_{rnd}(Q) = com_j$*, and* $com_j$ *was received by the simulator previously in an* $\mathcal{S}_2$ *query made by* $\mathcal{A}$*.*

*Claim 5.* Let $\mathsf{Q}_{\mathrm{rnd}}$ be the maximum number of hash queries $\mathcal{A}$ may make to the oracle $\mathsf{H}_{\mathrm{rnd}}$, and let $\mathsf{Q}_{\mathcal{S}_2}$ be the maximum number of $\mathcal{S}_2$ queries $\mathcal{A}$ may make. Then it holds that

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_2} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_1} - \frac{N \cdot \mathsf{Q}_{\mathcal{S}_2} \cdot \mathsf{Q}_{\mathrm{rnd}}}{2^{\lambda}}.$$

*Proof.* Since the output of the random oracle is uniformly random, the probability that a fresh hash query $Q$ to $\mathsf{H}_{\mathrm{rnd}}$ results in a desired fixed value $com_j \in \{0,1\}^\lambda$ is at most $\frac{1}{2^\lambda}$. In the worst case, $\mathcal{A}$ may submit up to $N \cdot \mathsf{Q}_{\mathcal{S}_2}$ commitments since it makes up to $\mathsf{Q}_{\mathcal{S}_2}$ queries and each query contains commitments $com_1, \ldots, com_N$. Thus, for each $\mathsf{H}$ query $\mathcal{A}$ makes, the probability of finding the preimage of one of the submitted commitments is at most $\frac{N \cdot \mathsf{Q}_{\mathcal{S}_2}}{2^\lambda}$. As $\mathcal{A}$ makes up to $\mathsf{Q}_{\mathrm{rnd}}$ queries, the probability of finding a preimage of some commitment $com$ is at most $\frac{N \cdot \mathsf{Q}_{\mathcal{S}_2} \cdot Q_{\mathrm{rnd}}}{2^\lambda}$.

*Game* $\mathbf{G}_3$. This game is identical to $\mathbf{G}_2$, except that it aborts and outputs 0 if $\mathcal{A}$ provokes Event $\mathbf{BAD}_2$:

$\mathbf{BAD}_2$: *$\mathcal{A}$ causes a collision in one of its queries to the random oracles $\mathsf{H}_m, \mathsf{H}_\mu, \mathsf{H}_{rnd}, \mathsf{H}_\Upsilon$, and $\mathsf{H}_{\mathsf{MT}}$.*[8] *That is, for $\mathsf{ro} \in RO \coloneqq \{m, \mu, rnd, \Upsilon, \mathsf{MT}\}$, $\mathcal{A}$ sends a fresh query $Q$, such that $\mathsf{H}_{\mathsf{ro}}(Q) = \mathsf{H}_{\mathsf{ro}}(Q')$ for some previous query $Q'$ with $Q \neq Q'$.*

*Claim 6.* Let $\mathsf{Q}_{\mathsf{ro}}$ be the maximum number of hash queries $\mathcal{A}$ may make to the oracle $\mathsf{H}_{\mathsf{ro}}$ for $\mathsf{ro} \in RO$. Then it holds

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_3} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_2} - \frac{\sum_{\mathsf{ro} \in RO} \mathsf{Q}_{\mathsf{ro}} \cdot (\mathsf{Q}_{\mathsf{ro}} - 1)}{2^{\lambda+1}}.$$

*Proof.* As the output of $\mathsf{H}_{\mathsf{ro}}$ is uniformly random, it holds for any pair $(Q_1, Q_2)$ that $\Pr[\mathsf{H}_{\mathsf{ro}}(Q_1) = \mathsf{H}_{\mathsf{ro}}(Q_2) \mid Q_1 \neq Q_2] \leq \frac{1}{2^\lambda}$. Assuming $\mathcal{A}$ makes at most $\mathsf{Q}_{\mathsf{ro}}$ queries to the oracle $\mathsf{H}_{\mathsf{ro}}$, the number of pairs of inputs is $\frac{\mathsf{Q}_{\mathsf{ro}} \cdot (\mathsf{Q}_{\mathsf{ro}} - 1)}{2}$. Consequently, the probability that a pair causes a collision is at most $\frac{\mathsf{Q}_{\mathsf{ro}} \cdot (\mathsf{Q}_{\mathsf{ro}} - 1)}{2^{\lambda+1}}$. It follows that the probability of causing a collision in some $\mathsf{ro} \in RO$ is at most $\frac{\sum_{\mathsf{ro} \in RO} \mathsf{Q}_{\mathsf{ro}} \cdot (\mathsf{Q}_{\mathsf{ro}} - 1)}{2^{\lambda+1}}$, hence the claim.

Per this game, for $\mathsf{ro} \in \{m, \mu, \mathsf{MT}, \Upsilon, \mathrm{rnd}\}$, and every hash output $y$, if $\mathsf{T}^{-1}_{\mathsf{ro}}[y] \neq \bot$, then there exists a unique query $Q$ such that $\mathsf{H}_{\mathsf{ro}}(Q) = y$.

*Game* $\mathbf{G}_4$. This game is identical to $\mathbf{G}_3$, except that it aborts and outputs 0 if $\mathcal{A}$ provokes Event $\mathbf{BAD}_3$, which is defined as follows:

$\mathbf{BAD}_3$: *$\mathcal{A}$ terminates and outputs a valid message-signature pair $(M, \Sigma)$ for $\mathbf{P}[\mathsf{BS}]$, which consists of $K$ valid message-signature pairs $(\Upsilon_1, \sigma_1, \ldots, \Upsilon_K, \sigma_K)$ for $\mathsf{BS}$, and $\exists i \in [K]$ such that $\mathsf{BS}.\mathcal{V}(\mathsf{pk}_i, \Upsilon_i, \sigma_i)$ does not query the random oracle $\mathsf{H}$ with a query $Q$ that was queried before to $\mathsf{H}$, i.e. $\mathsf{T}_{\mathsf{H}}[Q] \neq \bot$, where $\mathsf{T}_{\mathsf{H}}$ is the hash table of the random oracle $\mathsf{H}$.*

[8] Collisions in $\mathsf{H}_m$ and $\mathsf{H}_\mu$ do not need to be ruled out for the one-more unforgeability proof itself. We include them here because the same game sequence is later reused in the proof of secure showing.

*Claim 7.*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_4} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_3} - \varepsilon_{\mathsf{SHL}} \cdot K \cdot (\ell + 1).$$

*Proof.* The verification of a valid $\mathsf{BS}$ signature using $\mathsf{BS}.\mathcal{V}(\mathsf{pk}_i, \Upsilon_i, \sigma_i)$ makes a singular hash query $Q$ to $\mathsf{H}$, and the same hash query was made previously. Per Property 2, $\mathcal{A}$ may provoke $\mathbf{BAD}_3$ for a single $\mathsf{BS}$ message-signature pair $(\Upsilon_i, \sigma_i)$ with probability at most $\varepsilon_{\mathsf{SHL}}$. Since $\mathcal{A}$ outputs up to $\ell+1$ $\mathbf{P}[\mathsf{BS}]$ signatures, and each signature consists of $K$ $\mathsf{BS}$ signatures, the probability that $\mathbf{BAD}_3$ occurs is at most $\varepsilon_{\mathsf{SHL}} \cdot K \cdot (\ell + 1)$, hence the claim.

*Game* $\mathbf{G}_5$. This game is identical to $\mathbf{G}_4$, except that it aborts and outputs 0 if $\mathcal{A}$ provokes Event $\mathbf{BAD}_4$, which is defined as follows:

$\mathbf{BAD}_4$: *After $\mathcal{A}$ terminates and outputs a valid message-signature pair $(M, \Sigma)$ for $\mathbf{P}[\mathsf{BS}]$, which consists of $K$ valid message-signature pairs $(\Upsilon_1, \sigma_1, \ldots, \Upsilon_K, \sigma_K)$ for $\mathsf{BS}$, and for all $i \in [K]$, $\mathsf{BS}.\mathcal{V}(\mathsf{pk}_i, \Upsilon_i, \sigma_i)$ makes an $\mathsf{H}$ query $Q_i$ with $\mathsf{T}_{\mathsf{H}}[Q_i] \neq \perp$ ($Q_i$ has been queried previously), but $\exists j \in [K]$ such that $Q_j$ is not of the form $(\ldots, \Upsilon_j, \ldots)$ (i.e. the signed message is not in the hash query).*

*Claim 8.*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_5} = \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_4}.$$

*Proof.* Per Property 1, running $\mathcal{V}(\mathsf{pk}, \Upsilon, \sigma)$ makes a hash query to $\mathsf{H}$, and this query contains the signed message $\Upsilon$, i.e. the query is of the form $(\ldots, \Upsilon, \ldots)$.

*Game* $\mathbf{G}_6$. This game is identical to $\mathbf{G}_5$, except that it aborts and outputs 0 if $\mathcal{A}$ provokes Event $\mathbf{BAD}_5$, which occurs if $\mathcal{A}$ generates a valid user message $\mathsf{msgU}$ without relying on the random oracle $\mathsf{H}$.

$\mathbf{BAD}_5$: *For a subsession $(\mathsf{ctr}, i)$ for $i \in [K]$, $\mathcal{A}$ makes an $\mathcal{S}_3$ query with user message $\{\mathsf{msgU}_j\}_{j \in [N]}$, and then it makes an $\mathcal{S}_4$ query with input $\{\boldsymbol{R}_j, \mu_j, \mathsf{Urnd}_j\}_{J_i \neq j \in [N]}$ for $J_i \coloneqq \mathsf{H}_{\mathsf{J}}(\mathsf{seed}, \mathsf{pk}_i)$. The event occurs if, after $\mathcal{A}$ terminates, there exists $J_i \neq j' \in [N]$ satisfying the following:*

1. *Let $\zeta_{j'} \coloneqq \mathsf{MT}.\mathcal{C}(\boldsymbol{L}; R_{j'}), \Upsilon_{j'} \coloneqq \mathsf{H}_{\Upsilon}(\mathsf{pk}_i, \mu_{j'}, \zeta_{j'})$, where $\boldsymbol{L}$ is the attribute vector used in the session $\mathsf{ctr}$.*
2. *Let $\widehat{\mathsf{T}}$ be a snapshot of the hash table of $\mathsf{H}$ by the time $\mathcal{A}$ makes the $\mathcal{S}_4$ query.*
3. *Let $Q$ be the unique hash query made by the execution of $\mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_i, \Upsilon_{j'}, \mathsf{msgS1}_{j'}; \mathsf{Urnd}_{j'})$.*
4. *It holds that $\mathsf{msgU}_{j'} = \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_i, \Upsilon_{j'}, \mathsf{msgS1}_{j'}; \mathsf{Urnd}_{j'})$ and $\widehat{\mathsf{T}}[Q] = \perp$.*

*Claim 9.*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_6} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_5} - \varepsilon_{\mathsf{UUF}} \cdot \ell \cdot K \cdot (N - 1).$$

*Proof.* We note that the occurrence of $\mathbf{BAD}_5$ is a direct violation for the property described in Property 3, because $\mathcal{A}$'s output $(\mathsf{msgU}'_j, \boldsymbol{R}_{j'}, \mu_{j'}, \mathsf{Urnd}_{j'})$ can be directly transformed into an attack $(\mathsf{pk}_i, \Upsilon_{j'}, \mathsf{msgS1}_{j'}; \mathsf{Urnd}_{j'})$ against the said property. The probability of this happening for each tuple $\mathcal{A}$ submits is per Property 3 at most $\varepsilon_{\mathsf{UUF}}$. Since $\mathcal{A}$ makes up to $\ell \cdot K$ $\mathcal{S}_4$ queries, each containing $N-1$ tuples, we take the union bound over the $\ell \cdot K \cdot (N-1)$ tuples $\mathcal{A}$ outputs, which implies the claim.

*Game* $\mathbf{G}_7$. This game is identical to $\mathbf{G}_6$, except that it aborts and outputs 0 if $\mathcal{A}$ provokes Event $\mathbf{BAD}_6$, which we define as follows:

$\mathbf{BAD}_6$: *$\mathcal{A}$ makes an $\mathcal{S}_3$ query for a session* $\mathsf{ctr}$ *and a subsession $i \in [K]$ with user messages $\{\mathsf{msgU}_j\}_{j\in[N]}$. Then, for some $\mathsf{H}_\mathsf{J}(\mathsf{seed}, \mathsf{pk}_i) \neq j' \in [N]$, $\mathcal{A}$ makes an $\mathcal{S}_4$ query for the subsession ($\mathsf{ctr}$, $i$) with input $(R_{j'}, \mu_{j'}, \mathsf{Urnd}_{j'})$. Let $\zeta_{j'} \coloneqq \mathsf{MT}(\boldsymbol{L}; R_{j'}), \Upsilon_{j'} \coloneqq \mathsf{H}_\Upsilon(\mathsf{pk}_i, \mu_{j'}, \zeta_{j'})$, where $\boldsymbol{L}$ is the attribute vector used in the session $\mathsf{ctr}$, and it holds $\mathsf{msgU}_{j'} = \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_i, \Upsilon_{j'}, \mathsf{msgS1}_{j'}; \mathsf{Urnd}_{j'})$. Let $\widehat{\mathsf{T}_\mathsf{H}}$ be a snapshot of the hash table of $\mathsf{H}$ at the time the $\mathcal{S}_4$ query is made. The event occurs if*

1. *there is no query $Q' \in \widehat{\mathsf{T}_\mathsf{H}}$ such that $Q'$ is of the form $(\ldots, \Upsilon_{j'}, \ldots)$, or*
2. *there is a query $Q \in \widehat{\mathsf{T}_\mathsf{H}}$ such that $Q$ is of the form $(\ldots, \Upsilon, \ldots)$, $\Upsilon \neq \Upsilon_{j'}$, and $\mathsf{msgU}_{j'} = \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_i, \Upsilon, \mathsf{msgS1}_{j'}; \mathsf{Urnd}_{j'})$.*

*Claim 10.*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_7} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_6} - \ell \cdot K \cdot (N-1) \cdot \varepsilon_{\mathsf{UUO}}.$$

*Proof.* Since $\mathsf{msgU}_{j'} = \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_i, \Upsilon_{j'}, \mathsf{msgS1}_{j'}; \mathsf{Urnd}_{j'})$, $\mathcal{A}$ must have made a hash query to $\mathsf{H}$ identical to the hash query that would have been made by an honest $\mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_i, \Upsilon_{j'}, \mathsf{msgS1}_{j'}; \mathsf{Urnd}_{j'})$ execution; otherwise, Game $\mathbf{G}_6$ would have aborted. Additionally, this $\mathsf{H}$ query contains the message $\Upsilon_{j'}$ (due to Game $\mathbf{G}_5$). Furthermore, there cannot be another query with a different message $\Upsilon \neq \Upsilon_{j'}$ satisfying $\mathsf{msgU}_{j'} = \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_i, \Upsilon, \mathsf{msgS1}_{j'}; \mathsf{Urnd}_{j'}) = \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_i, \Upsilon_{j'}, \mathsf{msgS1}_{j'}; \mathsf{Urnd}_{j'})$ as this would violate the property described in Property 4, which occurs with probability at most $\varepsilon_{\mathsf{UUO}}$. Taking the union bound over $\ell$ $\mathbf{P}[\mathsf{BS}]$ sessions, each having $K$ subsessions, and $N-1$ user message per subsession (excluding the cut-and-choose user message), this event occurs with probability at most $\ell \cdot K \cdot (N-1) \cdot \varepsilon_{\mathsf{UUO}}$.

*Game* $\mathbf{G}_8$. This game is identical to $\mathbf{G}_7$, except that it aborts and outputs 0 if $\mathcal{A}$ provokes Event $\mathbf{BAD}_7$, which we define as follows:

$\mathbf{BAD}_7$: *$\mathcal{A}$ makes an $\mathcal{S}_3$ query for a session* $\mathsf{ctr}$ *with a user message* $\mathsf{msgU}_j$. *Let $\widehat{\mathsf{T}_\Upsilon}$ be a snapshot of the hash table of $\mathsf{H}_\Upsilon$ at the time this $\mathcal{S}_3$ query is made. Then, $\mathcal{A}$ makes an $\mathcal{S}_4$ query for this session with input $(R_j, \mu_j, \mathsf{Urnd}_j)$. Let*

$\zeta_j \coloneqq \mathsf{MT}.\mathcal{C}(\boldsymbol{L}; R_j), \Upsilon_j \coloneqq \mathsf{H}_{\Upsilon}(\mathsf{pk}, \mu_j, \zeta_j)$, *where* $\boldsymbol{L}$ *is the attribute vector used in this signing session. Then it holds* $\widehat{\mathsf{T}_{\Upsilon}}[\Upsilon_j] = \perp$ *and* $\mathsf{msgU}_j = \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}, \Upsilon_j, \mathsf{msgS1}_j; \mathsf{Urnd}_j)$.

*Claim 11.* Let $\mathsf{Q}_{\Upsilon}$ and $\mathsf{Q}_{\mathcal{S}_3}$ be the maximum number of hash queries $\mathcal{A}$ makes to $\mathsf{H}_{\Upsilon}$ and $\mathcal{S}_3$, respectively. Then it holds

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_8} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_7} - \frac{N \cdot \mathsf{Q}_{\mathcal{S}_3} \cdot \mathsf{Q}_{\Upsilon}}{2^{\lambda}}.$$

*Proof.* Due to Game $\mathbf{G}_7$, we know that the equality $\mathsf{msgU}_j = \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}, \Upsilon_j, \mathsf{msgS1}_j; \mathsf{Urnd}_j)$ implies that $\Upsilon_j$ is the only message existing in the table of the oracle $\mathsf{H}$, and no other message in that table satisfies this equality. Consequently, $\Upsilon_j$ is identical to the value $\mathcal{A}$ used to generate $\mathsf{msgU}_j$, because otherwise, $\mathcal{A}$ must have made an $\mathsf{H}$ with that message (due to Game $\mathbf{G}_6$). Since $\widehat{\mathsf{T}_{\Upsilon}}[\Upsilon_j] = \perp$, $\mathcal{A}$ managed to find the preimage $(\mathsf{pk}, \mu_j, \zeta_j)$ of $\Upsilon_j$ under $\mathsf{H}_{\Upsilon}$. Similar to Claim 5, this occurs with probability at most $\frac{N \cdot \mathsf{Q}_{\Upsilon} \cdot \mathsf{Q}_{\mathcal{S}_3}}{2^{\lambda}}$.

At this point, we know that, for each subsession $\mathsf{ctr}_i$ that will be closed via $\mathcal{S}_4$ successfully and for all $j \neq J = \mathsf{H}_{\mathsf{J}}(\mathsf{seed}, \mathsf{pk}_i)$:

- Upon $\mathcal{S}_2$ queries, all $com_j$ must have a corresponding entry $\mathsf{Urnd}_j$ in the table $\mathsf{T}_{\mathrm{rnd}}$ and this is extractable by $\mathcal{R}_3$; otherwise, if no entry in $\mathsf{T}_{\mathrm{rnd}}$, $\mathcal{A}$ will fail to find a preimage for $com_j$ (Game $\mathbf{G}_2$), hence it will fail to close $\mathcal{S}_4$.
- Upon $\mathcal{S}_4$ queries, $\mathcal{R}_3$ can obtain the message $\Upsilon_j$, and it is the same message $\mathcal{A}$ used to generate the user message $\mathsf{msgU}_j$ received previously in the $\mathcal{S}_3$ query (Game $\mathbf{G}_7$).

$\mathcal{R}_3$ requires this to hold even for $j = J$ because it needs $\mathsf{Urnd}_J$ and $\Upsilon_J$ to use $\mathcal{SIM}_{\mathcal{S}}$ to answer $\mathcal{A}$'s queries in the subsessions corresponding to the challenger's key. Since $\mathcal{A}$ does not send the openings for $\Upsilon_J$ and $com_J$ for any subsession, the argument above does not cover $j = J$. Concretely, $\mathcal{A}$ may guess the cut-and-choose index $J$ and behaves maliciously with the respective values (e.g., by computing $com_J$ and $\Upsilon_J$ without relying on $\mathsf{H}_{\mathrm{rnd}}$ and $\mathsf{H}_{\Upsilon}$). In the following, we rule out that $\mathcal{A}$ successfully computes $com_J$ or $\Upsilon_J$ in this manner.

Define Event $\mathbf{BAD}_8$ as follows:

$\mathbf{BAD}_8$: $\forall i \in [K], \exists \mathsf{ctr} \in \{2, \ldots, \ell + 1\}$, *such that in the subsession* $(\mathsf{ctr}, i)$, $\mathcal{A}$

- ***either*** *queries* $\mathcal{S}_2$ *with a commitment* $com_{J_i}$ *that was not generated using the random oracle* $\mathsf{H}_{rnd}$ *(i.e.,* $\mathsf{T}_{rnd}[com_{J_i}] = \perp$ *by the time of the* $\mathcal{S}_2$ *query),*
- ***or*** *queries* $\mathcal{S}_3$ *with a user message* $\mathsf{msgU}_{J_i}$ *for which, by the time the subsession closes, there does not exist a unique message* $\Upsilon_{J_i}$ *represented in the table of* $\mathsf{H}$ *such that*

$$\mathsf{msgU}_{J_i} = \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_i, \Upsilon_{J_i}, \mathsf{msgS1}_{J_i}; \mathsf{Urnd}_{J_i}),$$

– ***or*** *queries* $\mathcal{S}_3$ *with a user message* $\mathsf{msgU}_{J_i}$ *as above, while, by the time the subsession closes, there is no query to* $\mathsf{H}_{\Upsilon}$ *whose output is* $\Upsilon_{J_i}$*,*
– ***or*** *queries* $\mathcal{S}_3$ *with a user message* $\mathsf{msgU}_{J_i}$ *that was generated by* $\mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_i, \Upsilon_{J_i}, \mathsf{msgS1}_{J_i}; \mathsf{Urnd}_{J_i})$*, where*

$$\Upsilon_{J_i} = \mathsf{H}_{\Upsilon}(\mathsf{pk}_i, \mu_{J_i}, \zeta_{J_i}),$$

*and, by the time the subsession closes, there is no randomness* $R_{J_i}$ *extractable from the table of* $\mathsf{H}_{\mathsf{MT}}$ *such that*

$$\zeta_{J_i} = \mathsf{MT}.\mathcal{C}(\boldsymbol{L}_{\mathsf{ctr}}; R_{J_i}).$$

[9]

*and then closes this subsession* $(\mathsf{ctr}, i)$ *successfully by making a* $\mathcal{S}_4$ *query.*

Note that $\mathbf{BAD}_8$ can be efficiently checked upon successfully closing the subsession. The first case can be determined by checking the hash table of $\mathsf{H}_{\mathrm{rnd}}$ for every commitment submitted in a $\mathcal{S}_2$ query. For the second case, the game searches the table of $\mathsf{H}$ for all messages $\Upsilon$ satisfying

$$\mathsf{msgU}_{J_i} = \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_i, \Upsilon, \mathsf{msgS1}_{J_i}; \mathsf{Urnd}_{J_i}).$$

The second case occurs unless there is exactly one such message; denote it by $\Upsilon_{J_i}$. The third case can then be checked by searching the table of $\mathsf{H}_{\Upsilon}$ for a query whose output is $\Upsilon_{J_i}$. If such a query exists, the fourth case can be checked by extracting $\zeta_{J_i}$ from this query and reversing $\zeta_{J_i}$ through the table of $\mathsf{H}_{\mathsf{MT}}$ to check whether it commits to the attribute vector $\boldsymbol{L}_{\mathsf{ctr}}$ used in signing session $\mathsf{ctr}$.

*Game* $\mathbf{G}_9$. This game is the same as $\mathbf{G}_8$, except that it aborts and outputs 0 if Event $\mathbf{BAD}_8$ occurs.

*Claim 12.* Recall that $f(\mathsf{ctr}) = \lceil 3 \cdot \vartheta \cdot \ln(\ell+1) \cdot \mathsf{ctr} \rceil$ and $\delta > 0$ with $(1-\delta) \cdot \vartheta > 1$. It holds

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_9} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_8} - \exp(-\delta \cdot K).$$

*Proof.* Assume for the sake of contradiction that $\mathcal{A}$ provokes $\mathbf{BAD}_8$. This means that for each key $i$, there exists a signing session $\mathsf{ctr}$ and a subsession $(\mathsf{ctr}, i)$, in which $\mathcal{A}$ does the following: $\mathcal{A}$ chooses $j \in [N]$ arbitrarily and cheats in one of the ways specified in Event $\mathbf{BAD}_8$. We distinguish the following cases:

– **Case 1** $(j \neq J_i)$**.** $\mathcal{A}$ does not guess the cut-and-choose index successfully, hence it has to open the instance corresponding to $j$.

[9] This condition is not required for the one-more unforgeability proof itself. We include it here because the same game sequence is later used in the proof of secure showing.

If $\mathcal{A}$ generated $com_j$ without using the random oracle $\mathsf{H}_{\mathrm{rnd}}$, then successfully opening this instance corresponds to finding a preimage of $com_j$ under $\mathsf{H}_{\mathrm{rnd}}$. This is not possible because otherwise Game $\mathbf{G}_2$ would have aborted.
Next, suppose there does not exist a unique message $\Upsilon_j$ represented in the table of $\mathsf{H}$ satisfying

$$\mathsf{msgU}_j = \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_i, \Upsilon_j, \mathsf{msgS1}_j; \mathsf{Urnd}_j).$$

If no such message exists, Game $\mathbf{G}_6$ would have aborted. If two distinct messages satisfy this equality, Game $\mathbf{G}_7$ would have aborted.
Furthermore, if the unique message $\Upsilon_j$ was not generated using $\mathsf{H}_\Upsilon$, Game $\mathbf{G}_8$ would have aborted.
Finally, suppose that no opening of the corresponding commitment $\zeta_j$ to the attribute vector $\boldsymbol{L}_{\mathsf{ctr}}$ used in signing session $\mathsf{ctr}$ can be extracted from the table of $\mathsf{H}_{\mathsf{MT}}$. In the $\mathcal{S}_4$ query, the signer recomputes

$$\zeta'_j \coloneqq \mathsf{MT}.\mathcal{C}(\boldsymbol{L}_{\mathsf{ctr}}; R_j) \qquad \text{and} \qquad \Upsilon'_j \coloneqq \mathsf{H}_\Upsilon(\mathsf{pk}_i, \mu_j, \zeta'_j).$$

Since $R_j$ constitutes an opening of $\zeta'_j$ to $\boldsymbol{L}_{\mathsf{ctr}}$, it holds that $\zeta_j \neq \zeta'_j$. Hence, the corresponding check on $\mathsf{msgU}_j$ fails; otherwise, Game $\mathbf{G}_3$ or Game $\mathbf{G}_7$ would have aborted.

– **Case 2 ($j = J_i$).** $\mathcal{A}$ guesses successfully the cut-and-choose index $J_i$ for key $\mathsf{pk}_i$, hence it does not have to open the instance corresponding to $j$.

Since Case 1 does not occur, we focus on Case 2. More specifically, we want to analyze the probability that $\mathcal{A}$ cheats with respect to all keys $\mathsf{pk}_i$ by guessing the cut-and-choose index $J_i$.

We first compute the expected number of cheats per key $\mathsf{pk}_i$ for $i \in [K]$. Define independent Bernoulli random variables $\mathsf{cheat}_{i,\mathsf{ctr}}$ for $i = 1, \ldots, K$ and $\mathsf{ctr} = 2, \ldots, \ell + 1$, such that $\mathsf{cheat}_{i,\mathsf{ctr}} = 1$ if $\mathcal{A}$ cheats by successfully guessing $J_i$ in session $\mathsf{ctr}$, and $\mathsf{cheat}_{i,\mathsf{ctr}} = 0$ otherwise. Let

$$\mathsf{cheat}_i \coloneqq \sum_{\mathsf{ctr}=2}^{\ell+1} \mathsf{cheat}_{i,\mathsf{ctr}}.$$

Since $J_i$ is uniformly random in $[N]$, it holds that $\Pr[\mathsf{cheat}_{i,\mathsf{ctr}} = 1] \leq 1/N$. Consequently,

$$\mathbb{E}(\mathsf{cheat}_i) \leq \sum_{\mathsf{ctr}=2}^{\ell+1} \frac{1}{f(\mathsf{ctr})} = \frac{1}{3 \cdot \vartheta \cdot \ln(\ell+1)} \sum_{\mathsf{ctr}=2}^{\ell+1} \frac{1}{\mathsf{ctr}} \leq \frac{\ln(\ell+1)}{3 \cdot \vartheta \cdot \ln(\ell+1)} = \frac{1}{3 \cdot \vartheta}.$$

The second inequality follows from the fact that $\sum_{i=1}^{N} 1/i \leq \ln(N) + 1$. Substituting $X \coloneqq \mathsf{cheat}_i$ and $s \coloneqq 3 \cdot \mathbb{E}(\mathsf{cheat}_i) + \delta = \frac{1}{\vartheta} + \delta$ in Lemma 1 yields that for any $i \in [K]$

$$\Pr\left[\mathsf{cheat}_i \geq \frac{1}{\vartheta} + \delta\right] \leq \exp(3 \cdot \mathbb{E}(\mathsf{cheat}_i) - s) = \exp(-\delta). \tag{1}$$

Note that this equation states the probability of $\mathcal{A}$ successfully cheating at least once for $\mathsf{pk}_i$. That is,

$$\mathsf{cheat}_i < \frac{1}{\vartheta} + \delta \quad \Longrightarrow \quad \mathsf{cheat}_i < 1,$$

because we defined $\vartheta > 0$ and $\delta > 0$ as $(1-\delta)\cdot\vartheta > 1 \Leftrightarrow 1 > \delta + \frac{1}{\vartheta}$. This means that for any key $\mathsf{pk}_i$, the probability that $\mathcal{A}$ cheats successfully is at most $\exp(-\delta)$. Thus, the probability that $\mathcal{A}$ provokes Event $\mathbf{BAD}_8$ by cheating for all keys $\mathsf{pk}_i$ successfully is at most

$$\prod_{i\in[K]} \Pr\left[\mathsf{cheat}_i \geq \frac{1}{\vartheta} + \delta\right] \leq \exp(-\delta \cdot K),$$

hence the claim.

Now, we want to fix a uniformly random index $\iota \in [K]$. We want to rule out that $\mathcal{A}$ cheats for $\mathsf{pk}_\iota$ (this key will be the challenger's key in our reduction)

*Game* $\mathbf{G}_{10}$. Identical to $\mathbf{G}_9$, except that it aborts and outputs 0 if Even $\mathbf{BAD}_9$ occurs:

$\mathbf{BAD}_9$: *The event occurs if* $\mathcal{A}$ *cheats for* $\mathsf{pk}_\iota$ *(i.e.,* $\mathsf{cheat}_\iota \geq 1$*).*

*Claim 13.*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_{10}} \geq \frac{1}{K} \cdot \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_9}.$$

*Proof.* Due to Game $\mathbf{G}_9$, there exists $i^* \in [K]$ such that $\mathcal{A}$ does not cheat for $\mathsf{pk}_{i^*}$ ($\mathsf{cheat}_{i^*} < 1$). Since $\iota \in [K]$ is uniformly random and hidden from $\mathcal{A}$, the probability that $i^* = \iota$ is at least $\frac{1}{K}$.

*Game* $\mathbf{G}_{11}$. This game is identical to $\mathbf{G}_{10}$, except that it aborts if $\mathcal{A}$ provokes Event $\mathbf{BAD}_{10}$.

$\mathbf{BAD}_{10}$: *The game samples* $\kappa \leftarrow_\$ [\mathsf{Q}_\mathsf{H}]$*, where* $\mathsf{Q}_\mathsf{H}$ *is an upper bound for the number of hash queries* $\mathcal{A}$ *makes to the oracle* $\mathsf{H}$*. During the online phase,* $\mathcal{A}$ *makes an* $\mathcal{S}_4$ *query for the* $\iota$*-th subsession of a session* $\mathsf{ctr}$*, for which* $\mathcal{A}$ *made* $\mathcal{S}_3$ *query with input* $\{\mathsf{msgU}_j\}_{j\in[N]}$*, and the singular hash query made during the computation of* $\mathsf{msgU}_{J_\iota} = \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_\iota, \Upsilon_{J_\iota}, \mathsf{msgS1}_{J_\iota}; \mathsf{Urnd}_{J_\iota})$ *is the* $\kappa$*-th hash query made to* $\mathsf{H}$*.*

*Claim 14.*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_{11}} \geq \frac{\mathsf{Q}_\mathsf{H} - \ell}{\mathsf{Q}_\mathsf{H}} \cdot \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_{10}}.$$

*Proof.* $\mathcal{A}$ wins $\mathbf{G}_{11}$ iff it wins $\mathbf{G}_{10}$ and $\mathbf{BAD}_{10}$ does not occur. $\mathbf{BAD}_{10}$ occurs if $\mathcal{A}$ closes one signing session and this session is linked to the $\kappa$-th hash query. As $\mathcal{A}$ closes at most $\ell$ signing sessions and makes at most $\mathsf{Q}_\mathsf{H}$ queries, while $\kappa$ is uniformly random and hidden from $\mathcal{A}$, it holds that $\Pr[\mathbf{BAD}_{10}] \leq \frac{\ell}{\mathsf{Q}_\mathsf{H}}$. Consequently, we have

$$\begin{aligned}\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_{11}} &\geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_{10}} \cdot \Pr[\neg\mathbf{BAD}_{10}] \\ &\geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_{10}} \cdot (1 - \frac{\ell}{\mathsf{Q}_\mathsf{H}}) \\ &\geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_{10}} \cdot \frac{\mathsf{Q}_\mathsf{H} - \ell}{\mathsf{Q}_\mathsf{H}}.\end{aligned}$$

*Extracting the Keys from* $\mathsf{H}$ *Queries.* Note that the definition of three-move simulatable blind signatures (Section 4) does not enforce honest hash queries (e.g., those queries sent by $\mathsf{BS}.\mathcal{U}_1$ during signing) to include the public key of the signer. Without the ability to extract the key from the corresponding (honest) hash query and to program the oracle $\mathsf{H}$ respectively, it is not clear how to complete this proof. For this reason, our transform embeds the key inside the message being signed. Since the message must be included in the hash (otherwise Game $\mathbf{G}_5$ aborts), it remains to show how the oracle $\mathsf{H}$ exploits the message to extract the key from honest queries. Given a query $Q$, the simulator of $\mathsf{H}$ does the following:

- Parse $(\ldots, \Upsilon, \ldots) \leftarrow Q$, and abort if the parsing fails (recall that queries sent by $\mathsf{BS}.\mathcal{U}_1$ have this format due to Game $\mathbf{G}_5$.
- Look $\Upsilon$ up in Table $\mathsf{T}_\Upsilon$ and extract the corresponding query $Q_\Upsilon \coloneqq \mathsf{T}_\Upsilon^{-1}[\Upsilon]$. If $Q_\Upsilon$ is not of the form $(\mathsf{pk}_i, \mu, \zeta)$, abort.

Note that if the procedure aborts, the query is not honest (e.g., it corresponds to some arbitrary query made by $\mathcal{A}$) and does not contain a public key.

*Game* $\mathbf{G}_{12}$. This game is identical to $\mathbf{G}_{11}$, except that it aborts if $\mathcal{A}$ provokes Event $\mathbf{BAD}_{11}$, which we define as follows.

$\mathbf{BAD}_{11}$: *The event occurs if the following holds:*

1. *The game simulates everything as in* $\mathbf{G}_{11}$*, but it extracts the key from honest queries as described above and maintains a counter to track the number of queries per key.*
2. $\mathcal{A}$ *terminates and outputs* $\ell+1$ *valid* $\mathbf{P}[\mathsf{BS}]$ *message-signature pairs* $(M_1, \Sigma_1), \ldots, (M_{\ell+1}, \Sigma_{\ell+1})$.
3. *For all* $t \in [\ell + 1]$ :
   a. *Parse* $(\boldsymbol{\sigma}_t, \boldsymbol{\eta}_t, \boldsymbol{\zeta}_t) \coloneqq \Sigma_t$. *Then, parse* $(\sigma_1^{(t)}, \ldots, \sigma_K^{(t)}) \coloneqq \boldsymbol{\sigma}_t$, $(\eta_1^{(t)}, \ldots, \eta_K^{(t)}) \coloneqq \boldsymbol{\eta}_t$, *and* $(\zeta_1^{(t)}, \ldots, \zeta_K^{(t)}) \coloneqq \boldsymbol{\zeta}_t$.
   b. *Set* $\sigma_0^{(t)} \coloneqq \eta_0^{(t)} \coloneqq \zeta_0^{(t)} \coloneqq \bot$, $m_0^{(t)} \coloneqq M_t$.
   c. *For all* $i = 1 \ldots \iota$, *compute* $m_i^{(t)} \coloneqq \mathsf{H}_m(\sigma_{i-1}^{(t)}, \eta_{i-1}^{(t)}, \zeta_{i-1}^{(t)}, m_{i-1}^{(t)})$, $\mu_i^{(t)} \coloneqq \mathsf{H}_\mu(m_i^{(t)}, \eta_i^{(t)})$, $\Upsilon_i^{(t)} \coloneqq \mathsf{H}_\Upsilon(\mathsf{pk}_i, \mu_i^{(t)}, \zeta_i^{(t)})$.

*d. It holds* $\mathsf{BS}.\mathcal{V}^{\mathsf{H}}(\mathsf{pk}_\iota, \Upsilon_\iota^{(t)}, \sigma_\iota^{(t)}) = 1$*, and the singular hash query made by* $\mathsf{BS}.\mathcal{V}$ *(Game* $\mathbf{G}_4$*) is not the* $\kappa$*-th* $\mathsf{H}$ *query.*

*Claim 15.*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_{12}} \geq \frac{1}{\mathsf{Q}_{\mathsf{H}} - \ell} \cdot \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_{11}} - \varepsilon_{\mathsf{DS}}.$$

*Proof.* First, due to Game $\mathbf{G}_5$, the hash query must contain the signed message $\Upsilon$, hence $\mathbf{G}_{12}$ can identify the occurrence of $\mathbf{BAD}_{11}$. Second, we recall Property 6 that implies that each hash query may be linked to at most a single signature (except with probability at most $\varepsilon_{\mathsf{DS}}$). $\mathcal{A}$ can distinguish two types of queries:

- **Type 1** Queries used in honest $\mathsf{BS}.\mathcal{U}_1$ executions, and the resulting user messages $\mathsf{msgU}$ were sent to the signer via $\mathcal{S}_3$ queries, and these sessions were closed later via $\mathcal{S}_4$ queries. These are $\ell$ queries.
- **Type 2** The remainder of the queries $\mathcal{A}$ makes to $\mathsf{H}$. These are $\mathsf{Q}_{\mathsf{H}} - \ell$ queries.

Due to Game $\mathbf{G}_{11}$, the $\kappa$-th query belongs to the second type; otherwise, the game would have aborted. Thus, $\mathcal{A}$'s best strategy to provoke $\mathbf{BAD}_{11}$ is to link $\ell$ forgeries to the queries of the first type, and the remaining forgery to a randomly chosen query of the second type. In this case, $\mathbf{G}_{12}$ aborts with probability at most $1 - \frac{1}{\mathsf{Q}_{\mathsf{H}} - \ell}$, hence $\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_{12}} \geq \frac{1}{\mathsf{Q}_{\mathsf{H}} - \ell} \cdot \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_{11}} - \varepsilon_{\mathsf{DS}}$.

The following game hop happens only when the underlying scheme is a three-move scheme.

*Game* $\mathbf{G}_{13}$. This game is identical to $\mathbf{G}_{12}$, except that it aborts if $\mathcal{A}$ provokes Event $\mathbf{BAD}_{12}$, which we define as follows.

$\mathbf{BAD}_{12}$: *$\mathcal{A}$ makes a hash query to* $\mathsf{H}$ *that has the format of honest hash queries, and this query can be explained via two distinct tuples* $(\mathsf{msgS1}, \mathsf{Urnd})$ *and* $(\mathsf{msgS1}', \mathsf{Urnd}')$ *that were provided previously by $\mathcal{A}$ for the same message-key pair* $(\Upsilon, \mathsf{pk})$*, i.e.* $\mathsf{hq}(\mathsf{pk}, \Upsilon, \mathsf{msgS1}, \mathsf{Urnd})$ *and* $\mathsf{hq}(\mathsf{pk}, \Upsilon, \mathsf{msgS1}', \mathsf{Urnd}')$*, where* $\mathsf{hq}$ *is the function defined in Property 1.*

*Claim 16.*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_{13}} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_{12}} - \varepsilon_{\mathsf{UL}}.$$

*Proof.* First, we note that the occurrence of $\mathbf{BAD}_{12}$ can be checked efficiently, since for every hash query, we know that the query contains signed message $\Upsilon$, and the corresponding public key can be extracted from the table of $\mathsf{H}_\Upsilon$ as described above. Having $\mathsf{pk}$ and $\Upsilon$, $\mathbf{G}_{13}$ searches all $(\mathsf{msgS1}, \mathsf{Urnd})$ and checks if two combinations $(\mathsf{pk}, \Upsilon, \mathsf{msgS1}, \mathsf{Urnd})$ and $(\mathsf{pk}, \Upsilon, \mathsf{msgS1}', \mathsf{Urnd}')$ explain the incoming query. Provoking $\mathbf{BAD}_{12}$ requires $\mathcal{A}$ to open two signing subsessions $(\mathsf{ctr}, i)$ and $(\mathsf{ctr}', i)$ for $i \in [K]$, such that the signing algorithm generates two $\mathsf{BS}$ signer messages $\mathsf{msgS1}$ and $\mathsf{msgS1}'$ for a public key $\mathsf{pk}_i$ such that these signer messages along with the user randomness $\mathsf{Urnd}$ and $\mathsf{Urnd}'$ satisfy $\mathsf{hq}(\mathsf{pk}_i, \Upsilon, \mathsf{msgS1}, \mathsf{Urnd}) = \mathsf{hq}(\mathsf{pk}_i, \Upsilon, \mathsf{msgS1}', \mathsf{Urnd}')$. This, however, directly violates Property 5, which must hold except with probability $\varepsilon_{\mathsf{UL}}$.

***Reducing* $\mathbf{0}$-$\mathsf{OMUF}_{\mathsf{BS}}$ *to* $\mathbf{G}_{13}$.** In the following we describe our reduction $\mathcal{R}_3$.

- **Setup Phase.** $\mathcal{R}_3$ takes as input the 0-$\mathsf{OMUF}_{\mathsf{BS}}$ challenger's public key $\mathsf{pk}'$, samples $\iota \leftarrow_{\$} [K]$, sets $\mathsf{pk}_\iota := \mathsf{pk}'$, samples $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow_{\$} \mathsf{BS}.\mathcal{G}$ for $i \in [K] \backslash \{\iota\}$, sets $\mathsf{pk} := (\mathsf{pk}_1, \ldots, \mathsf{pk}_K)$. $\mathcal{R}_3$ initializes a global counter for completed sessions $\mathsf{closed} := 0$, a global session ID counter $\mathsf{ctr} := 1$, and a state map $\mathsf{state}$. $\mathcal{R}_3$ then invokes $\mathcal{A}(\mathsf{pk})$.
- **Online Phase.** $\mathcal{R}_3$ simulates the signer oracles and the random oracles for $\mathcal{A}$. $\mathcal{A}$ may query the oracles concurrently and in an interleaved manner. $\mathcal{A}$ may complete at most $\ell$ sessions. $\mathcal{R}_3$ simulates the oracles as follows:
  - $\mathcal{RG}_{\mathsf{S}}(\boldsymbol{L})$. Exactly as described by Construction 6.
  - $\mathcal{S}_1(\boldsymbol{L})$. If $\boldsymbol{L}$ is not registered, abort and output $\perp$. Increment $\mathsf{ctr} := \mathsf{ctr} + 1$, and set $N := f(\mathsf{ctr})$. For $i \in [K]$, sample $\mathsf{seed}_i \leftarrow_{\$} \{0,1\}^\lambda$, compute $com_{\mathsf{seed}\,i} := \mathsf{H}_{\mathsf{J}}(\mathsf{seed}_i)$, compute $J_i := \mathsf{H}_{\mathsf{J}}(\mathsf{seed}_i, \mathsf{pk}_i)$. Store $(\boldsymbol{L}, N, \{\mathsf{seed}_i, J_i\}_{i\in[K]}, i := 1, \mathsf{round} := 1)$ in $\mathsf{state}[\mathsf{ctr}]$. Return $(\mathsf{ctr}, N, \{com_{\mathsf{seed}\,i}\}_{i\in[K]})$.
  - $\mathcal{S}_2(\mathsf{sk}, \mathsf{ctr}, \{com_j\}_{j\in[N]})$. Retrieve $(i, \mathsf{round})$ from $\mathsf{state}[\mathsf{ctr}]$. Abort and output $\perp$ if $\mathsf{round} \neq 1$. Parse $\mathsf{sk} := (\mathsf{sk}_1, \ldots, \mathsf{sk}_K)$.
    - **If $(i = \iota)$:** Set $\mathsf{cheat} := 0$. Compute $(\mathsf{msgS1}_j, \mathsf{simSt}_j) \leftarrow_{\$} \mathsf{BS}.\mathcal{SIM}_{\mathcal{S}_1}(\mathsf{pk}_i)$. Store $\mathsf{msgS1}_{i,j} := \mathsf{msgS1}_j, \mathsf{simSt}_{i,j} := \mathsf{simSt}_j, \mathsf{Urnd}'_{i,j} := \mathsf{Urnd}'_j$, $com_{i,j} := com_j$ in $\mathsf{state}[\mathsf{ctr}]$.
    - **Else $(i \neq \iota)$:** For $j \in [N]$: compute $(\mathsf{msgS1}_j, \mathsf{rnd}_{\mathsf{S}\,j}) \leftarrow_{\$} \mathsf{BS}.\mathcal{S}_1(\mathsf{sk}_i)$. Update the state and store $\mathsf{msgS1}_{i,j} := \mathsf{msgS1}_j, \mathsf{rnd}_{\mathsf{S}\,i,j} := \mathsf{rnd}_{\mathsf{S}\,j}$, $com_{i,j} := com_j$ in $\mathsf{state}[\mathsf{ctr}]$.

    Store $(\mathsf{round} := 2, \mathsf{cheat})$ in $\mathsf{state}[\mathsf{ctr}]$ (if 'cheat' was defined). Return $\{\mathsf{msgS1}_j\}_{j\in[N]}$.
  - $\mathcal{S}_3(\mathsf{ctr}, \{\mathsf{msgU}_j\}_{j\in[N]})$. Retrieve $(i, \mathsf{round}, \mathsf{seed})$ from $\mathsf{state}[\mathsf{ctr}]$. Abort and output $\perp$ if $\mathsf{round} \neq 2$. For $j \in [N]$, store $\mathsf{msgU}_{i,j} := \mathsf{msgU}_j$ and $\mathsf{round} := 3$ in $\mathsf{state}[\mathsf{ctr}]$. Return $\mathsf{seed}_i$.
  - $\mathcal{S}_4(\mathsf{sk}, \mathsf{ctr}, \{R_j, \mu_j, \mathsf{Urnd}_j\}_{j\neq J_i})$. Retrieve $(\boldsymbol{L}, N, \{J_i\}_{i\in[K]}, i, \mathsf{round}, \mathsf{cheat}, \mathsf{msgS1}, \mathsf{rnd}_{\mathsf{S}\,i,j}, \mathsf{simSt}_{i,j}, \mathsf{msgU}_{i,j})$ from $\mathsf{state}[\mathsf{ctr}]$. Retrieve $\mathsf{Urnd}'_{i,j}$ from the table $\mathsf{T}_{\mathsf{com}}$. If $\mathsf{BS}$ is a two-round scheme, find the (unique due to $\mathbf{G}_7$) unassigned tuple $(\mathsf{pk}, \Upsilon, \mathsf{simSt})$ in $\mathsf{state}$ such that $\mathsf{msgU}_{i,j} = \mathcal{U}_1(\mathsf{pk}, \Upsilon, \perp, \mathsf{Urnd}_{i,j})$ and set $\mathsf{simSt}_{i,j} := \mathsf{simSt}$. Abort and output $\perp$ if $\mathsf{round} \neq 3$ or $\mathsf{closed} = \ell$. Parse $\mathsf{sk} := (\mathsf{sk}_1, \ldots, \mathsf{sk}_K)$. For $j \in [N] \backslash \{J_i\}$:
    1. Abort and output $\perp$ if $com_j \neq \mathsf{H}_{\mathrm{rnd}}(\mathsf{Urnd}_j)$.
    2. Compute $\zeta_j := \mathsf{MT}.\mathcal{C}(\boldsymbol{L}; R_j)$ and $\Upsilon_j := \mathsf{H}_{\Upsilon}(\mathsf{pk}_i, \mu_j, \zeta_j)$.
    3. Abort and output $\perp$ if $\mathsf{msgU}_{i,j} \neq \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_i, \Upsilon_j, \mathsf{msgS1}_{i,j};\ \mathsf{Urnd}_j)$.

    **If $(i = \iota)$:**
    - If the hash query corresponding to $\mathsf{Urnd}_{i,j}$ (as described in Game $\mathbf{G}_6$) was generated by calling the challenger's oracle $\mathsf{H}_{\mathbf{ch}}$ (see the simulation of $\mathsf{H}$ when $\mathsf{ctr}_{\mathsf{H}} = \kappa$), abort and output $\perp$ (this is excluded in Game $\mathbf{G}_{11}$).
    - Search $\mathsf{T}_{\Upsilon}$ for $\Upsilon_{J_i}$ such that $\mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_\iota,\ \Upsilon_{J_i}, \mathsf{msgS1}_{i,J_i}; \mathsf{Urnd}'_{i,J_i}) = \mathsf{msgU}_{i,J_i}$. If $\Upsilon_{J_i} = \perp$ or $\mathsf{cheat} = 1$, abort and output $\perp$ (this is excluded in Game $\mathbf{G}_9$).

* Generate the signer second message $\mathsf{msgS2}_i \leftarrow \mathsf{BS}.\mathcal{SIM}_{\mathcal{S}_2}(\mathsf{msgS1}_{i,J_i}, \mathsf{simSt}_{i,J_i}, \mathsf{msgU}_{i,J_i}, \Upsilon_{J_i}, \mathsf{Urnd}'_{i,J_i})$.

**Else ($i \neq \iota$):**

* Compute $\mathsf{msgS2}_i \leftarrow \mathsf{BS}.\mathcal{S}_2(\mathsf{sk}_i, \mathsf{rnd}_{\mathsf{S}i,J_i}, \mathsf{msgU}_{i,J_i})$.
* **If ($i = K$):** Increment $\mathsf{closed} \coloneqq \mathsf{closed} + 1$ and store $\mathsf{round} \coloneqq 4$ in $\mathsf{state}[\mathsf{ctr}]$.
* **Else ($i < K$):** Store $(i \coloneqq i + 1, \mathsf{round} \coloneqq 1)$ in $\mathsf{state}[\mathsf{ctr}]$.

Return $\mathsf{msgS2}_i$.

- $[\mathsf{H}_{\mathrm{rnd}}, \mathsf{H}_\Upsilon](Q)$. Simulate via lazy sampling, but abort if $\mathcal{A}$ finds a preimage (ruled out in Games $\mathbf{G}_2$ and $\mathbf{G}_8$) or a collision (ruled out in Game $\mathbf{G}_3$).
- $[\mathsf{H}_\mu, \mathsf{H}_{\mathsf{MT}}, \mathsf{H}_{\mathsf{J}}, \mathsf{H}_m](Q)$. Via lazy sampling.
- $\mathsf{H}(Q)$. Intuitively, the oracle attempts to parse the query $Q$ as it would do with an honest hash query sent by $\mathsf{BS}.\mathcal{U}_1(\mathsf{pk}_i, \ldots)$. If the parsing fails (i.e., $Q$ is not an honest query), it answers via lazy sampling. Otherwise, it extracts the public key $\mathsf{pk}_i$ from the query. If this key is not the challenger's key $\mathsf{pk}_\iota$, it answers via lazy sampling. Otherwise, i.e., $i = \iota$, it answers the queries using the signer simulation algorithm $\mathcal{SIM}_{\mathcal{S}}$, except for one random query (the $\kappa$-th), it forwards the query to the challenger's oracle and answers with $\mathsf{H}_{\mathbf{ch}}(Q)$. Concretely, the oracle behaves as follows:
  * Initially, sample an index $\kappa \leftarrow_{\$} [\mathsf{Q}]$ uniformly at random and start a counter $\mathsf{ctr}_{\mathsf{H}} \coloneqq 0$ that increments by one after each query.
  * If $\mathsf{T}[Q] \neq \bot$, return $\mathsf{T}[Q]$.
  * Parse $(\ldots, \Upsilon, \ldots) \leftarrow Q$ (recall that queries sent by $\mathsf{BS}.\mathcal{U}_1$ have this format due to Game $\mathbf{G}_5$). If the parsing fails, answer via lazy sampling.
  * Look up $\Upsilon$ in Table $\mathsf{T}_\Upsilon$ and extract the corresponding query $Q_\Upsilon \leftarrow \mathsf{T}_\Upsilon^{-1}[\Upsilon]$. If $Q_\Upsilon = \bot$, simulate via lazy sampling (this corresponds to a message that will not be used to close any signing session due to Game $\mathbf{G}_8$).
  * If $Q_\Upsilon$ is not of the form $(\mathsf{pk}_i, \mu, \zeta)$ or $i \neq \iota$, simulate via lazy sampling.
  * If $\mathsf{ctr}_{\mathsf{H}} = \kappa$, set $\mathsf{T}[Q] \coloneqq \mathsf{H}_{\mathbf{ch}}(Q)$, and return $\mathsf{T}[Q]$.
  * Otherwise ($\kappa \neq \mathsf{ctr}_{\mathsf{H}}$). Knowing which signing session this query belongs to (due to Game $\mathbf{G}_{13}$), identify the value $(\mathsf{pk}, \mathsf{msgS1}, \mathsf{simSt}, \mathsf{Urnd}, \Upsilon)$ corresponding to that session. Let $\mathsf{T}[Q] \leftarrow_{\$} \mathcal{SIM}_{\mathsf{H}}(\mathsf{pk}, \mathsf{msgS1}, \mathsf{simSt}, \mathsf{Urnd}, \Upsilon)$, and return $\mathsf{T}[Q]$. In the case that the underlying scheme $\mathsf{BS}$ is a two-move scheme, identify $(\mathsf{pk}, \Upsilon)$ and run $\mathcal{SIM}_{\mathsf{H}}$ to obtain $c$ and $\mathsf{simSt}$. Let $\mathsf{T}[Q] = c$ and store $(\mathsf{pk}, \Upsilon, \mathsf{simSt})$ in state, not yet assigned to any session.

– **Winning 0-$\mathsf{OMUF}^{\mathcal{R}_3}_{\mathsf{BS}}$.** When $\mathcal{A}$ terminates and outputs $\ell+1$ valid message-signature pairs $(M_1, \Sigma_1), \ldots, (M_{\ell+1}, \Sigma_{\ell+1})$ for our transform $\mathbf{P}[\mathsf{BS}]$, the reduction $\mathcal{R}_3$ searches for a *sub-signature* $\sigma_\iota$ on a message $\Upsilon_\iota$ such that $\mathsf{BS}.\mathcal{V}^{\mathsf{H}_{\mathbf{ch}}}(\mathsf{pk}_\iota, \Upsilon_\iota, \sigma_\iota) = 1$. More specifically, $\mathcal{R}_3$ does the following. For all $t \in [\ell + 1]$:

1. Parse $(\boldsymbol{\sigma}_t, \boldsymbol{\eta}_t, \boldsymbol{\zeta}_t) \leftarrow \Sigma_t$.
2. Parse $(\sigma_1^{(t)}, \ldots, \sigma_K^{(t)}) \leftarrow \boldsymbol{\sigma}_t$, $(\eta_1^{(t)}, \ldots, \eta_K^{(t)}) \leftarrow \boldsymbol{\eta}_t$, and $(\zeta_1^{(t)}, \ldots, \zeta_K^{(t)}) \leftarrow \boldsymbol{\zeta}_t$.
3. Set $\sigma_0^{(t)} \coloneqq \eta_0^{(t)} \coloneqq \zeta_0^{(t)} \coloneqq \bot$, $m_0^{(t)} \coloneqq M_t$.
4. For $i = 1 \ldots \iota$:
   a. Compute $m_i^{(t)} \coloneqq \mathsf{H}_m(\sigma_{i-1}^{(t)}, \eta_{i-1}^{(t)}, \zeta_{i-1}^{(t)}, m_{i-1}^{(t)})$, $\mu_i^{(t)} \coloneqq \mathsf{H}_\mu(m_i^{(t)}, \eta_i^{(t)})$, $\Upsilon_i^{(t)} \coloneqq \mathsf{H}_\Upsilon(\mathsf{pk}_i, \mu_i^{(t)}, \zeta_i^{(t)})$.
   b. If $\mathsf{BS}.\mathcal{V}^{\mathsf{H}_{\mathsf{ch}}}(\mathsf{pk}_\iota, \Upsilon_\iota^{(t)}, \sigma_\iota^{(t)}) = 1$: Abort and output the forgery $(\Upsilon_\iota, \sigma_\iota)$.

Obviously, this is a valid forgery against $\mathsf{0\text{-}OMUF}_{\mathsf{BS}}$ allowing $\mathcal{R}_3$ to win the game. Given that Game $\mathbf{G}_{12}$ does not abort, $\mathcal{R}_3$ can find a subsignature for the challenger's public key $\mathsf{pk}_\iota$ that is valid under the $\kappa$-th hash query (the one programmed using the challenger's oracle).

Overall, we have:

$$
\begin{aligned}
\mathbf{Adv}^{\ell\text{-}\mathsf{OMUF}}_{\mathbf{P}[\mathsf{BS}]}(\mathcal{A}) &\coloneqq \mathbf{Adv}^{\mathbf{G}_1} \\
&\leq \mathbf{Adv}^{\mathbf{G}_2} + \frac{N \cdot \mathsf{Q}_{\mathcal{S}_2} \cdot \mathsf{Q}_{\mathrm{rnd}}}{2^\lambda} \\
&\leq \mathbf{Adv}^{\mathbf{G}_3} + \frac{\sum_{\mathsf{ro} \in RO} \mathsf{Q}_{\mathsf{ro}} \cdot (\mathsf{Q}_{\mathsf{ro}} - 1)}{2^{\lambda+1}} + \frac{N \cdot \mathsf{Q}_{\mathcal{S}_2} \cdot \mathsf{Q}_{\mathrm{rnd}}}{2^\lambda} \\
&\leq \mathbf{Adv}^{\mathbf{G}_4} + \varepsilon_{\mathsf{SHL}} \cdot K \cdot (\ell + 1) + \epsilon'_{\mathrm{ro}} \\
&\leq \mathbf{Adv}^{\mathbf{G}_5} + \varepsilon_{\mathsf{SHL}} \cdot K \cdot (\ell + 1) + \epsilon'_{\mathrm{ro}} \\
&\leq \mathbf{Adv}^{\mathbf{G}_6} + \varepsilon_{\mathsf{UUF}} \cdot \ell \cdot K \cdot (N - 1) + \varepsilon_{\mathsf{SHL}} \cdot K \cdot (\ell + 1) + \epsilon'_{\mathrm{ro}} \\
&\leq \mathbf{Adv}^{\mathbf{G}_7} + (\varepsilon_{\mathsf{UUF}} + \varepsilon_{\mathsf{UUO}}) \cdot \ell \cdot K \cdot (N - 1) + \varepsilon_{\mathsf{SHL}} \cdot K \cdot (\ell + 1) + \epsilon'_{\mathrm{ro}} \\
&\leq \mathbf{Adv}^{\mathbf{G}_8} + \epsilon_{\mathrm{bad}} + \epsilon'_{\mathrm{ro}} + \frac{N \cdot \mathsf{Q}_{\mathcal{S}_3} \cdot \mathsf{Q}_\Upsilon}{2^\lambda} \\
&\leq \mathbf{Adv}^{\mathbf{G}_9} + \exp(-\delta \cdot K) + \epsilon_{\mathrm{bad}} + \epsilon_{\mathrm{ro}} \\
&\leq K \cdot \mathbf{Adv}^{\mathbf{G}_{10}} + \epsilon_{\mathrm{cheat}} + \epsilon_{\mathrm{bad}} + \epsilon_{\mathrm{ro}} \\
&\leq K \cdot \frac{\mathsf{Q}_{\mathsf{H}}}{\mathsf{Q}_{\mathsf{H}} - \ell} \cdot \mathbf{Adv}^{\mathbf{G}_{11}} + \epsilon_{\mathrm{cheat}} + \epsilon_{\mathrm{bad}} + \epsilon_{\mathrm{ro}} \\
&\leq K \cdot \mathsf{Q}_{\mathsf{H}} \cdot (\mathbf{Adv}^{\mathbf{G}_{12}} + \varepsilon_{\mathsf{DS}}) + \epsilon_{\mathrm{cheat}} + \epsilon_{\mathrm{bad}} + \epsilon_{\mathrm{ro}} \\
&\leq K \cdot \mathsf{Q}_{\mathsf{H}} \cdot (\mathbf{Adv}^{\mathbf{G}_{13}} + \varepsilon_{\mathsf{UL}} + \varepsilon_{\mathsf{DS}}) + \epsilon_{\mathrm{cheat}} + \epsilon_{\mathrm{bad}} + \epsilon_{\mathrm{ro}} \\
&\leq K \cdot \mathsf{Q}_{\mathsf{H}} \cdot \mathbf{Adv}^{\mathbf{G}_{13}} + \epsilon_{\mathrm{cheat}} + \epsilon_{\mathrm{ro}} + \epsilon_{\mathrm{bs}} \\
&\leq K \cdot \mathsf{Q}_{\mathsf{H}} \cdot \mathbf{Adv}^{\mathsf{0\text{-}OMUF}}_{\mathsf{BS}}(\mathcal{R}_3) + \epsilon_{\mathrm{cheat}} + \epsilon_{\mathrm{ro}} + \epsilon_{\mathrm{bs}}.
\end{aligned}
$$

where

$$
\begin{aligned}
\epsilon'_{\mathrm{ro}} &\coloneqq \frac{\sum_{\mathsf{ro} \in RO} \mathsf{Q}_{\mathsf{ro}} \cdot (\mathsf{Q}_{\mathsf{ro}} - 1)}{2^{\lambda+1}} + \frac{N \cdot \mathsf{Q}_{\mathcal{S}_2} \cdot \mathsf{Q}_{\mathrm{rnd}}}{2^\lambda}, \\
\epsilon_{\mathrm{ro}} &\coloneqq \epsilon'_{\mathrm{ro}} + \frac{N \cdot \mathsf{Q}_{\mathcal{S}_3} \cdot \mathsf{Q}_\Upsilon}{2^\lambda}, \\
\epsilon_{\mathrm{cheat}} &\coloneqq \exp(-\delta \cdot K), \\
\epsilon_{\mathrm{bad}} &\coloneqq \varepsilon_{\mathsf{SHL}} \cdot K \cdot (\ell + 1) + \ell \cdot K \cdot (N - 1) \cdot (\varepsilon_{\mathsf{UUF}} + \varepsilon_{\mathsf{UUO}}), \\
\epsilon_{\mathrm{bs}} &\coloneqq \epsilon_{\mathrm{bad}} + K \cdot \mathsf{Q}_{\mathsf{H}} \cdot (\varepsilon_{\mathsf{UL}} + \varepsilon_{\mathsf{DS}}).
\end{aligned}
$$

### C.2 Full Proof of Secure Showing

**Theorem 2.** *Let* $\mathsf{BS}$ *be a blind signature scheme satisfying the properties of Definition 4. Let* $\mathbf{P}[\mathsf{BS}]$ *be the scheme resulting from applying our transform on* $\mathsf{BS}$*. If* $\mathbf{P}[\mathsf{BS}]$ *is* $(t,\varepsilon)$*-*$\ell$*-*$\mathsf{OMUF}*, then* $\mathbf{P}[\mathsf{BS}]$ *is* $(t',\varepsilon')$*-*$\ell$*-*$\mathsf{SH}$ *secure with unique messages (item 4.c.i) and selective reveal (item 4.d.ii), with* $t' \approx t$ *and*

$$\varepsilon' \leq K \cdot \mathbf{Adv}^{\mathsf{vcBINDING}}_{\mathsf{MT}}(\mathcal{R}_4) + \varepsilon.$$

*Proof.* Assume there is an adversary $\mathcal{A}$ winning Game $\ell$-$\mathsf{SH}$ by outputting $\ell$ pairs $(\mathsf{rst}^{\mathsf{ctr}}_{\mathsf{U}}, \boldsymbol{L}'_{\mathsf{ctr}})$ along with $q \in [\ell]$ tuples $(M_{\mathsf{ctr}}, \Sigma_{\mathsf{ctr}}, \boldsymbol{L}_{\mathsf{ctr}}, I_{op,\mathsf{ctr}}, \pi_{\mathsf{ctr}})$ for $\mathsf{ctr} \in [q]$ satisfying the constraints defined in Game $\ell$-$\mathsf{SH}$.

Recall that in our transform, the user does not send a commitment of its attributes to the signer in the registration phase, instead it sends the attribute vector in plain $\boldsymbol{L}$. Therefore, whenever $\mathsf{rst}^{\mathsf{ctr}}_{\mathsf{U}}.C \neq \bot$, the winning condition requires $\boldsymbol{L}'_{\mathsf{ctr}}$ to be the attribute vector corresponding to the $\mathsf{ctr}$-th signing session.

Akin to the one-more unforgeability proof, we want to show that the adversary cannot cheat for all keys $\mathsf{pk}_i$ for $i \in [K]$. That is, there must exist some public key $\mathsf{pk}_i$, for which $\mathcal{A}$ does not cheat at all. We note that the simulation of both games $\ell$-$\mathsf{OMUF}$ and $\ell - \mathsf{SH}$ is identical, since the simulator must simulate at most $\ell$ signing sessions for $\mathcal{A}$, in addition to the signer registration oracle and the random oracles. Thus, we use the same game hops we used in the OMUF proof to rule out the same bad events ruled out in the OMUF proof.

We proceed via game series.

*Game* $\mathbf{G}_1$*.* This is Game $\ell$-$\mathsf{SH}$.

*Game* $\mathbf{G}_2$*.* This game is identical to $\mathbf{G}_1$, except that it aborts and outputs 0 if $\mathcal{A}$ provokes Event $\mathbf{BAD}_1$.

*Claim 17.* Let $\mathsf{Q}_{\mathrm{rnd}}$ be the maximum number of hash queries $\mathcal{A}$ may make to the oracle $\mathsf{H}_{\mathrm{rnd}}$, and let $\mathsf{Q}_{\mathcal{S}_2}$ be the maximum number of $\mathcal{S}_2$ queries $\mathcal{A}$ may make. Then it holds that

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_2} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_1} - \frac{N \cdot \mathsf{Q}_{\mathcal{S}_2} \cdot \mathsf{Q}_{\mathrm{rnd}}}{2^{\lambda}}.$$

The proof is analogous to the proof of Claim 5.

*Game* $\mathbf{G}_3$*.* This game is identical to $\mathbf{G}_2$, except that it aborts and outputs 0 if $\mathcal{A}$ provokes Event $\mathbf{BAD}_2$.

*Claim 18.* Let $RO \coloneqq \{m, \mu, \mathrm{rnd}, \Upsilon, \mathsf{MT}\}$ and let $\mathsf{Q}_{\mathsf{ro}}$ be an upper bound for the number of queries $\mathcal{A}$ makes to $\mathsf{H}_{\mathsf{ro}}$. Then it holds

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_3} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_2} - \frac{\sum_{\mathsf{ro} \in RO} \mathsf{Q}_{\mathsf{ro}} \cdot (\mathsf{Q}_{\mathsf{ro}} - 1)}{2^{\lambda+1}}.$$

The proof is analogous to the proof of claim 6.

*Game* $\mathbf{G}_4$. This game is identical to $\mathbf{G}_3$, except that it aborts and outputs 0 if $\mathcal{A}$ provokes Event $\mathbf{BAD}_3$.

*Claim 19.*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_4} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_3} - \varepsilon_{\mathsf{SHL}} \cdot K \cdot (\ell + 1).$$

The proof is analogous to the proof of Claim 7.

*Game* $\mathbf{G}_5$. This game is identical to $\mathbf{G}_4$, except that it aborts and outputs 0 if $\mathcal{A}$ provokes Event $\mathbf{BAD}_4$.

*Claim 20.*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_5} = \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_4}$$

The proof is analogous to the proof of Claim 8.

*Game* $\mathbf{G}_6$. This game is identical to $\mathbf{G}_5$, except that it aborts and outputs 0 if $\mathcal{A}$ provokes Event $\mathbf{BAD}_6$.

*Claim 21.*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_6} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_5} - \varepsilon_{\mathsf{UUF}} \cdot \ell \cdot K \cdot (N - 1).$$

The proof is analogous to the proof of Claim 9

*Game* $\mathbf{G}_7$. This game is identical to $\mathbf{G}_6$, except that it aborts and outputs 0 if $\mathcal{A}$ provokes Event $\mathbf{BAD}_6$.

*Claim 22.*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_7} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_6} - \ell \cdot K \cdot (N - 1) \cdot \varepsilon_{\mathsf{UUO}}.$$

The proof is analogous to the proof of Claim 10.

*Game* $\mathbf{G}_8$. This game is identical to $\mathbf{G}_7$, except that it aborts and outputs 0 if $\mathcal{A}$ provokes Event $\mathbf{BAD}_7$.

*Claim 23.* Let $\mathsf{Q}_{\Upsilon}$ and $\mathsf{Q}_{\mathcal{S}_3}$ be the maximum number of hash queries $\mathcal{A}$ makes to $\mathsf{H}_{\Upsilon}$ and $\mathcal{S}_3$, respectively. Then it holds

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_8} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_7} - \frac{N \cdot \mathsf{Q}_{\mathcal{S}_3} \cdot \mathsf{Q}_{\Upsilon}}{2^{\lambda}}.$$

The proof is analogous to the proof of Claim 11.

*Game* $\mathbf{G}_9$. This game is identical to $\mathbf{G}_8$, except that it aborts and outputs 0 if $\mathcal{A}$ provokes Event $\mathbf{BAD}_8$.

*Claim 24.* Recall that $f(\mathsf{ctr}) = \lceil 3 \cdot \vartheta \cdot \ln(\ell+1) \cdot \mathsf{ctr} \rceil$ and $\delta > 0$ with $(1-\delta) \cdot \vartheta > 1$. It holds

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_9} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_8} - \exp(-\delta \cdot K).$$

The proof is analogous to the proof of Claim 12.

*Game* $\mathbf{G}_{10}$. Identical to $\mathbf{G}_9$, except that it aborts and outputs 0 if Even $\mathbf{BAD}_9$ occurs.

*Claim 25.*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_{10}} \geq \frac{1}{K} \cdot \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_9}.$$

The proof is analogous to the proof of Claim 13.

Per this game, we know there is a uniformly chosen $\iota \in [K]$ such that $\mathcal{A}$ does not cheat for $\mathsf{pk}_\iota$. Recall that $\mathcal{A}$ outputs $(M_j, \Sigma_j, \boldsymbol{L}_j, I_{op,j}, \pi_j)$ for $j \in [q]$. Each signature-message pair $(M_j, \Sigma_j)$ consists of $K$ message-subsignature pairs $(\Upsilon_i, \sigma_i)$ for $i \in [K]$. Additionally, recall that $\mathcal{A}$ closes at most $\ell$ signing sessions and each session consists of $K$ subsessions, a subsession per key $\mathsf{pk}_i$. Now, we want to rule out that all subsignatures corresponding to the $\iota$-th key $\mathsf{pk}_\iota$ are linked to some closed signing subsession corresponding to that key. That is, we want a game ensuring that there exists a message-subsignature pair $(\Upsilon_\iota^{(\mathsf{ctr}^*)}, \sigma_\iota^{(\mathsf{ctr}^*)})$ that is not linked to any closed signing subsession $(\mathsf{ctr}, \iota)$.

The remainder of the proof is in Section 7.

### C.3 Blindness

**Theorem 3.** *For a blind signature scheme* $\mathsf{BS}$ *fulfilling definition 4 that is* $(t, \varepsilon)$*-blind, the scheme* $\mathbf{P}[\mathsf{BS}]$ *is* $(t', \varepsilon')$*-blind with* $t' \approx t$ *and* $\varepsilon' = K \cdot \varepsilon + 2 \cdot \frac{Q^2_{\mathsf{H_J}}}{2^\lambda} + 2\frac{Q_{\mathsf{H_J}}}{2^{\lambda-1}} + 2\frac{2K \cdot Q_{\mathsf{H_{com}}}}{2^\lambda} + 2\frac{2KQ_{\mathsf{H_{MT}}}}{\lambda\nu}$.

*Proof.* In the following, we gradually transition from the blindness game with $b = 0$ to the blindness game with $b = 1$. We denote by $\mathbf{G}_\mathbf{i} = 1$ that the adversary outputs 1 when playing the game.

*Game* $\mathbf{G}_1$. This is the original blindness game of the scheme $\mathbf{P}[\mathsf{BS}]$ with $b = 0$.

*Game* $\mathbf{G}_2$. This game introduces an abort condition that aborts when there is a collision in $\mathsf{H_J}$, i.e., whenever the adversary queries $x \neq x'$ such that $\mathsf{H_J}(x) = \mathsf{H_J}(x')$. It holds that

$$|\Pr[\mathbf{G}_2 = 1] - \Pr[\mathbf{G}_1 = 1]| \leq \frac{Q^2_{\mathsf{H_J}}}{2^\lambda}.$$

*Game* $\mathbf{G}_3$. In this game, when the adversary opens a new signing session by sending a cut-and-choose parameter $N$ and a commitment $com_{\mathsf{seed}_i}$ to either of the user oracles, the game aborts if there exists no query $\mathsf{H_J}(\mathsf{seed}_{J_i}) = com_{\mathsf{seed}_i}$ at that point, but the adversary successfully opens the commitment $com_{\mathsf{seed}_i}$ later on. The probability that the adversary can later open $com_{\mathsf{seed}_i}$ to a value $\mathsf{seed}_i$ is $\frac{Q_{\mathsf{H_J}}}{2^\lambda}$. Union-bounding over the two user oracles yields

$$|\Pr[\mathbf{G}_3 = 1] - \Pr[\mathbf{G}_2 = 1]| \leq \frac{Q_{\mathsf{H_J}}}{2^{\lambda-1}}.$$

*Game* $\mathbf{G}_4$. In this game, we change how the commitments $com_{\text{rnd}}^{(i,J_i)}$ are generated. Instead of picking $\mathsf{Urnd}_{i,J_i} \leftarrow_{\$} \{0,1\}^\lambda$ and computing $com_{\text{rnd}}^{(i,J_i)} := \mathsf{H}_{\text{rnd}}(\mathsf{Urnd}_{i,J_i})$, the game samples $com_{\text{rnd}}^{(i,J_i)} \leftarrow_{\$} \{0,1\}^\lambda$. Clearly, the adversary does not notice this difference unless it queries one of the $\mathsf{Urnd}_{i,J_i}$ of one of the sessions to $\mathsf{H}_{\mathsf{com}}$. Thus, it holds that

$$|\Pr[\mathbf{G}_4 = 1] - \Pr[\mathbf{G}_3 = 1]| \leq \frac{2KQ_{\mathsf{H}_{\mathsf{com}}}}{2^\lambda}.$$

*Game* $\mathbf{G}_5$. In this game, we change how the commitments $\mu_{i,j}$ for $j \neq J_i$ are generated. Namely, instead of computing $\mu_{i,j} = \mathsf{H}_\mu(m, \eta_{i,j})$ , we randomly sample $\mu_{i,j} \leftarrow_{\$} \{0,1\}^\lambda$. This is noticeable if the adversary queried one of the $\eta_{i,j_i}$ to the RO which happens with probability at most $\frac{K \cdot N \cdot Q_{\mathsf{H}_\mu}}{2^\lambda}$. Thus it holds that $|\Pr[\mathbf{G}_5 = 1] - \Pr[\mathbf{G}_4 = 1]| \leq \frac{K \cdot N \cdot Q_{\mathsf{H}_\mu}}{2^\lambda}$.

*Game* $\mathbf{G}_6$. This game aborts whenever the adversary makes a query to $\mathsf{H}_{\mathsf{MT}}$ that is made in the Merkle-Trees of the indices from $\boldsymbol{J}$ for some challenge message before receiving the openings during the showing phase.

This happens with probability at most $\frac{2 \cdot K \cdot Q_{\mathsf{H}_{\mathsf{MT}}}}{\lambda \cdot \nu}$ and thus

$$|\Pr[\mathbf{G}_6 = 1] - \Pr[\mathbf{G}_5 = 1]| \leq \frac{2 \cdot K \cdot Q_{\mathsf{H}_{\mathsf{MT}}}}{\lambda \cdot \nu}.$$

*Game* $\mathbf{G}_7$. In this game, we change how the commitments $\zeta_{i,J_i}$ are generated. Namely, instead of computing a Merkle-Tree $\zeta_{i,J_i} := \mathsf{MT}(\boldsymbol{L}; \boldsymbol{R}_{i,J_i})$, the game samples $\zeta_{i,J_i} \leftarrow_{\$} \{0,1\}^\lambda$. When the adversary asks to show the signature, the game picks random values $\boldsymbol{R}_{i,J_i,k}, k \in \boldsymbol{I_{op}}$ as well as values for the co-path of the opened indices. It then programs the Random oracle to so that the root of the Merkle-Tree is $\zeta_{i,J_i}$. By the changes introduced in $\mathbf{G}_3$ and $\mathbf{G}_6$, $\mathbf{G}_7$ is identically distributed to $\mathbf{G}_6$.

*Game* $\mathbf{G}_8$. In this game, we swap the bits $b$ to 1. We do a hybrid argument over the underlying public keys. We start out with $\mathbf{G}_8^0$, which is the same as $\mathbf{G}_7$. In $\mathbf{G}_8^i$, for the first $i$ public keys, the roles of $M_0$ and $M_1$ are swapped in the generation of the messages $\Upsilon_{k,J_k}$ for $k \leq i$. In the end, the underlying message-signature pairs are reassembled according to the messages, i.e. the first $i$ pairs are swapped back. Showing including the opening of Merkle-Trees works as before. It is easy to see that $\mathbf{G}_8$ is the same as $\mathbf{G}_8^K$. Each of these steps can be bounded by the blindness of the underlying scheme. We provide a reduction for the game hop from $\mathbf{G}_8^{i-1}$ to $\mathbf{G}_8^i$:

- The reduction receives the public key from the adversary. It forwards $\mathsf{pk}_i$ to its challenger. Using the messages $M_0$ and $M_1$ from the signer, it generates $\mu_{i,J_i}$ for each session, samples $\zeta_{i,J_i}$, and computes $\Upsilon_{i,J_i,0}$ and $\Upsilon_{i,J_i,1}$ accordingly. It provides $\Upsilon_{i,J_i,0}$ and $\Upsilon_{i,J_i,1}$ to its challenger as the challenge messages.

– It implements the signing oracles as in $\mathbf{G}_8^{i-1}$ using its control over the random oracles, except that to generate the user messages of the $i$-th public key, it queries the user oracles of its challenger.
– To generate the output signatures, it reassembles the signatures as described above.
– When showing the signatures, it programs the random oracles to open the Merkle-Trees as described in $\mathbf{G}_7$.
– When asked to output a bit, it outputs the bit the adversary outputs.

It is easy to see that the reduction simulates $\mathbf{G}_8^{i-1}$ or $\mathbf{G}_8^{i}$ perfectly, depending on the challenger's bit. It thus holds that

$$\left|\Pr[\mathbf{G}_8^{i} = 1] - \Pr[\mathbf{G}_8^{i-1}]\right| \leq \mathbf{Adv}_{\mathsf{BS}}^{\mathsf{BLIND}}(\mathcal{R}).$$

Summing over $i = 1, \ldots, K$ yields

$$|\Pr[\mathbf{G}_8 = 1] - \Pr[\mathbf{G}_7 = 1]| \leq K \cdot \mathbf{Adv}_{\mathsf{BS}}^{\mathsf{BLIND}}(\mathcal{R}).$$

*Game* $\mathbf{G}_9$. In this game, we revert the change from $\mathbf{G}_7$. By the same argument as in $\mathbf{G}_7$, it holds that

$$|\Pr[\mathbf{G}_9 = 1] - \Pr[\mathbf{G}_8 = 1]| \leq \frac{2KQ_{\mathsf{H}_{\mathsf{MT}}}}{\lambda \times \nu}.$$

*Game* $\mathbf{G}_{10}$. This game reverts the changes introduced in $\mathbf{G}_6$. By the same argument as in $\mathbf{G}_6$ it holds that

$$|\Pr[\mathbf{G}_{10} = 1] - \Pr[\mathbf{G}_9 = 1]| \leq \frac{2KQ_{\mathsf{H}_{\mathsf{MT}}}}{\lambda \times \nu}.$$

*Game* $\mathbf{G}_{11}$. In this game, we revert the change from $\mathbf{G}_5$, but swapping $M_0$ and $M_1$. By the same argument as in $\mathbf{G}_5$, it holds that

$$|\Pr[\mathbf{G}_{11} = 1] - \Pr[\mathbf{G}_{10} = 1]| \leq \frac{K \cdot N \cdot Q_{\mathsf{H}_\mu}}{2^\lambda}.$$

*Game* $\mathbf{G}_{12}$. In this game, we revert the change from $\mathbf{G}_4$. By the same argument as in $\mathbf{G}_4$, it holds that

$$|\Pr[\mathbf{G}_{12} = 1] - \Pr[\mathbf{G}_{11} = 1]| \leq \frac{2KQ_{\mathsf{H}_{\mathsf{com}}}}{2^\lambda}.$$

*Game* $\mathbf{G}_{13}$. In this game, we revert the change from $\mathbf{G}_3$. By the same argument as in $\mathbf{G}_3$, it holds that

$$|\Pr[\mathbf{G}_{13} = 1] - \Pr[\mathbf{G}_{12} = 1]| \leq \frac{Q_{\mathsf{H_J}}}{2^{\lambda-1}}.$$

*Game* $\mathbf{G}_{14}$. In this game, we revert the change from $\mathbf{G}_2$. By the same argument as in $\mathbf{G}_2$, it holds that

$$|\Pr[\mathbf{G}_{14} = 1] - \Pr[\mathbf{G}_{13} = 1]| \leq \frac{Q_{\mathsf{H_J}}^2}{2^{\lambda}}.$$

$\mathbf{G}_{14}$ is identical to the original blindness game but with $b = 1$.
Summing up the probabilities we get that

$$\begin{aligned}\mathbf{Adv}^{\mathsf{BLIND}}_{\mathbf{P}[\mathsf{BS}]}(\mathcal{A}) &= 2\left|\Pr[b = b'] - \frac{1}{2}\right| = |\Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0]| \\ &\leq 2 \cdot \left(\frac{Q_{\mathsf{H_J}}^2}{2^{\lambda}} + \frac{Q_{\mathsf{H_J}}}{2^{\lambda-1}} + \frac{2 \cdot K \cdot Q_{\mathsf{H_{com}}}}{2^{\lambda}} + \frac{2 \cdot K \cdot Q_{\mathsf{H_{MT}}}}{\lambda \cdot \nu}\right) \\ &+ K \cdot \mathbf{Adv}^{\mathsf{BLIND}}_{\mathsf{BS}}(\mathcal{R}).\end{aligned}$$

# D Blind Schnorr Signatures as an Instantiation of Three-Move Simulatable Blind Signature Schemes

**Construction 7 (Blind Schnorr Signature).** For a security parameter $\lambda$, let $\mathsf{pp} \coloneqq (\mathbb{G}, p, g)$ be a public parameter specifying a group $\mathbb{G}$ of order $p$ and a generator $g$, and $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p$ be a hash function treated as a random oracle. The Blind Schnorr signature scheme $\mathsf{BS} = (\mathcal{G}, \mathcal{S}, \mathcal{SIM}_{\mathcal{S}}, \mathcal{U}, \mathcal{V})$ is defined as follows:

- **Key Generation** $\mathcal{G}(\mathsf{pp})$: Sample a secret key $x \leftarrow_\$ \mathbb{Z}_p$ and set $\mathsf{pk} \coloneqq g^x$. Output $(\mathsf{pk}, \mathsf{sk} \coloneqq x)$.
- **Signer Algorithms** $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$:
  - $\mathcal{S}_1^{\mathsf{H}}(\mathsf{sk})$: Sample $r \leftarrow_\$ \mathbb{Z}_p$. Compute $R \coloneqq g^r$. Output $\mathsf{msgS1} \coloneqq R$ and $\mathsf{rnd_S} \coloneqq r$.
  - $\mathcal{S}_2^{\mathsf{H}}(\mathsf{sk}, \mathsf{rnd_S}, \mathsf{msgU})$: Parse $\mathsf{rnd_S}$ as $r$ and $\mathsf{msgU}$ as $c$. Compute $s \coloneqq r + c \cdot x \pmod p$. Output $\mathsf{msgS2} \coloneqq s$.
- **User Algorithms** $\mathcal{U}^{\mathsf{H}} = (\mathcal{U}_1^{\mathsf{H}}, \mathcal{U}_2^{\mathsf{H}})$:
  - $\mathcal{U}_1^{\mathsf{H}}(\mathsf{pk}, m, \mathsf{msgS1}; \mathsf{Urnd})$: Parse $\mathsf{msgS1}$ as $R$ and $\mathsf{Urnd}$ as $(\alpha, \beta) \in \mathbb{Z}_p^2$. Compute $R' \coloneqq R \cdot g^{\alpha} \cdot \mathsf{pk}^{-\beta}$. Query $c' \coloneqq \mathsf{H}(R', m)$. Compute $c \coloneqq c' - \beta \pmod p$. Output $\mathsf{msgU} \coloneqq c$.
  - $\mathcal{U}_2^{\mathsf{H}}(\mathsf{pk}, m, \mathsf{Urnd}, \mathsf{msgS2})$: Parse $\mathsf{Urnd}$ as $(\alpha, \beta)$ and $\mathsf{msgS2}$ as $s$. Compute $s' \coloneqq s + \alpha \pmod p$. Output $\sigma \coloneqq (c', s')$.

– **Verification** $\mathcal{V}^{\mathsf{H}}(\mathsf{pk}, m, \sigma)$: Parse $\sigma$ as $(c', s')$. Compute $R' := g^{s'} \cdot \mathsf{pk}^{-c'}$. Output 1 if $c' = \mathsf{H}(R', m)$, and 0 otherwise.
– **Signer Simulation Algorithms** $\mathcal{SIM}^{\mathsf{H}}_{\mathcal{S}} = (\mathcal{SIM}^{\mathsf{H}}_{\mathcal{S}_1}, \mathcal{SIM}^{\mathsf{H}}_{\mathcal{S}_2}, \mathcal{SIM}_{\mathsf{H}})$:
  • $\mathcal{SIM}_{\mathcal{S}_1}(\mathsf{pk})$: Sample $s, c \leftarrow_{\$} \mathbb{Z}_p$. Compute $R := g^s \cdot \mathsf{pk}^{-c}$. Output $\mathsf{msgS1} := R$ and $\mathsf{simSt} := (s, c)$.
  • $\mathcal{SIM}_{\mathcal{S}_2}(\mathsf{pk}, \mathsf{msgS1}, \mathsf{simSt}, \mathsf{Urnd}, m)$: Parse $\mathsf{simSt}$ as $(s, c)$. Run the User algorithm to obtain the challenge: $\mathsf{msgU} \leftarrow \mathcal{U}_1^{\mathcal{SIM}_{\mathsf{H}}}(\mathsf{pk}, m, \mathsf{msgS1}; \mathsf{Urnd})$. Output $\mathsf{msgS2} := s$.
  • $\mathcal{SIM}_{\mathsf{H}}(\mathsf{pk}, \mathsf{msgS1}, \mathsf{simSt}, \mathsf{Urnd}, m)$: Parse $\mathsf{simSt}$ as $(s, c)$ and $\mathsf{Urnd}$ as $(\alpha, \beta)$. Compute $R' := R \cdot g^{\alpha} \cdot \mathsf{pk}^{-\beta}$ (which equals $g^{s+\alpha}\mathsf{pk}^{-\beta-c}$). Program the random oracle such that $\mathsf{H}(R', m) := c + \beta \pmod p$. Output $c_{\mathsf{H}} := c + \beta$.

We show that Blind Schnorr satisfies the properties described in Section 4.
**Property 1:** holds, because in an honest protocol execution, $\mathcal{U}_1$ and $\mathcal{V}$ make the same hash query to the random oracle. The hash query is derived from the signer's message $\mathsf{msgS1}$, the user's randomness $\mathsf{Urnd}$, and the message $m$, and the key $\mathsf{pk}$as $R' := \mathsf{msgS1} \cdot g^{\alpha} \cdot \mathsf{pk}^{-\beta}$; therefore, the hash can be uniquely identified by $\mathsf{hq}(\mathsf{pk}, m, \mathsf{msgS1}, \mathsf{Urnd} = (\alpha, \beta)) = (R', m)$ with $R'$ as above.
**Property 2:** holds since violating this property is equivalent to guessing the output of $\mathsf{H}$. More specifically, given a valid message-signature pair $(m, \sigma)$ satisfying $\mathcal{V}(\mathsf{pk}, m, \sigma) = 1$, it holds for $(c', s')$ that $c' = \mathsf{H}(g^{s'} \cdot \mathsf{pk}^{-c'}, m)$. Since $\mathsf{H}$ behaves like a true random function, the probability of this property being violated is at most $\frac{1}{p} =: \varepsilon_{\mathsf{UUF}}$.
**Property 3:** This property holds since the user message $\mathsf{msgU} = c$ is derived from the output of the random oracle $c'$ and the user randomness, i.e. $c = c' - \beta$, and outputting a user message without having queried the random oracle $\mathsf{H}$ beforehand is, again, equivalent to guessing the output of the random oracle. To see this, let $\mathcal{A}$ be an adversary violating Property 3 by outputting $(\mathsf{msgU}, \mathsf{pk}, m, \mathsf{msgS1}, \mathsf{Urnd})$, hence $\mathsf{msgU} = \mathcal{U}_1(\mathsf{pk}, m, \mathsf{msgS1}, \mathsf{Urnd}) = \mathsf{H}(R', m) - \beta$ for $R' := \mathsf{msgS1} \cdot g^{\alpha} \cdot \mathsf{pk}^{-\beta}$ and $\mathsf{Urnd} = (\alpha, \beta)$. It follows that $\mathsf{msgU} + \beta = \mathsf{H}(R', m)$. Since $\mathsf{H}$ is a truly random function, $\mathcal{A}$ succeeds with probability at most $\frac{1}{p}$.
**Property 4:** Let $\mathcal{A}$ be an adversary violating this property by outputting distinct tuples $(m_1, \mathsf{Urnd}_1)$ and $(m_2, \mathsf{Urnd}_2)$ with $\mathsf{Urnd}_i = (\alpha_i, \beta_i)$ satisfying $\mathcal{U}_1(\mathsf{pk}, m_1, \mathsf{msgS1}, \mathsf{Urnd}_1) = \mathcal{U}_1(\mathsf{pk}, m_2, \mathsf{msgS1}, \mathsf{Urnd}_2)$ for some public key $\mathsf{pk}$ and a signer's first message $\mathsf{msgS1}$.

Recall that the hash query is of the form $\mathsf{hq}(\mathsf{pk}, m_i, \mathsf{msgS1}, \mathsf{Urnd}_i) = (R' = \mathsf{msgS1} \cdot g^{\alpha_i} \cdot \mathsf{pk}^{\beta_i}, m_i)$. We consider the following cases:

– $\mathsf{hq}(\mathsf{pk}, m_1, \mathsf{msgS1}, \mathsf{Urnd}_1) = \mathsf{hq}(\mathsf{pk}, m_2, \mathsf{msgS1}, \mathsf{Urnd}_2)$. In this case, it holds that $m_1 = m_2$ and $R' = R \cdot g^{\alpha_1} \cdot \mathsf{pk}^{-\beta_1} = R \cdot g^{\alpha_2} \cdot \mathsf{pk}^{-\beta_2}$ with $(\alpha_1, \beta_1) \neq (\alpha_2, \beta_2)$. A reduction embedding a discrete logarithm challenge in $\mathsf{pk}$ can solve for $\mathrm{dlog}_g \mathsf{pk} = \frac{\alpha_1 - \alpha_2}{\beta_1 - \beta_2}$. The adversary's advantage can be bounded by $\varepsilon_{\mathsf{DLP}}$.
– $\mathsf{hq}(\mathsf{pk}, m_1, \mathsf{msgS1}, \mathsf{Urnd}_1) \neq \mathsf{hq}(\mathsf{pk}, m_2, \mathsf{msgS1}, \mathsf{Urnd}_2)$ and at least one of the hash queries was not made by the adversary before. This boils down to the adversary guessing the hash output and succeeding with probability $\frac{1}{p}$.

– Otherwise: The two hash queries are distinct and have both been made. In this case, a reduction embedding a discrete logarithm challenge into $\mathsf{pk}$ can solve for $\mathsf{pk}$ via rewinding the adversary and reprogramming the RO at the latter of the two queries. Without loss of generality, consider the case that $\mathsf{hq}(\mathsf{pk}, m_1, \mathsf{msgS1}; \mathsf{Urnd}_1)$ was made first. Denote by $\mathsf{msgU} = \mathcal{U}_1(\mathsf{pk}, m_1, \mathsf{msgS1}; \mathsf{Urnd}_1)$. If the adversary succeeds in both runs at producing $(\mathsf{pk}, m_2, \mathsf{msgS1}, \mathsf{Urnd}_2)$ and $(\mathsf{pk}, m_2', \mathsf{msgS1}, \mathsf{Urnd}_2')$ that both produce $\mathsf{msgU}$ with the same hash query, then the reduction can solve for $\mathrm{dlog}_g \mathsf{pk}$ as above. The reduction succeeds with probability approximately $\frac{\varepsilon_{\mathsf{UUO}}}{2}\left(\frac{\varepsilon_{\mathsf{UUO}}}{2 \cdot \mathsf{Q_H}} - \frac{1}{p}\right) \leq \varepsilon_{\mathsf{DLP}}$.

Overall, we can bound $\varepsilon_{\mathsf{UL}} \leq 2 \cdot \mathsf{Q_H}\sqrt{\varepsilon_{\mathsf{DLP}}} + \varepsilon_{\mathsf{DLP}} + \frac{1}{p}$.

**Property 5:** Assume an adversary $\mathcal{A}$ violates this property by outputting a tuple $(m, \mathsf{Urnd}_1, \mathsf{Urnd}_2)$ such that $\mathsf{hq}(\mathsf{pk}, m, \mathsf{msgS1}_1, \mathsf{Urnd}_1) = \mathsf{hq}(\mathsf{pk}, m, \mathsf{msgS1}_2, \mathsf{Urnd}_2)$ for some public key $\mathsf{pk}$ and honestly-generated signer's first messages $\mathsf{msgS1}_1$ and $\mathsf{msgS1}_2$. Let $(\alpha_1, \beta_1) \coloneqq \mathsf{Urnd}_1$ and $(\alpha_2, \beta_2) \coloneqq \mathsf{Urnd}_2$. It must hold that $\mathsf{msgS1}_1 \cdot g^{\alpha_1} \cdot \mathsf{pk}^{-\beta_1} = \mathsf{msgS1}_2 \cdot g^{\alpha_2} \cdot \mathsf{pk}^{-\beta_2}$. If $\mathcal{A}$ violates Property 5, we distinguish two cases:

– Either $\mathsf{msgS1}_1 = \mathsf{msgS1}_2$. This implies that $\mathsf{Urnd}_1 \neq \mathsf{Urnd}_2$, hence $\beta_1 \neq \beta_2$. Since $\beta_2 - \beta_1 \neq 0$, a reduction that embeds in the public key $\mathsf{pk}$ can compute $\mathrm{dlog}_g \mathsf{pk} = \mathrm{dlog}_g \mathbf{ch} = \frac{\alpha_1 - \alpha_2}{\beta_1 - \beta_2}$ and solve the discrete logarithm problem.
– Or $\mathsf{msgS1}_1 \neq \mathsf{msgS1}_2$. In this case, the reduction embeds the discrete logarithm in $\mathsf{msgS1}$ and generates $\mathsf{msgS1}_2 = g^r$. It also picks the public key as $g^x$ for $x \leftarrow_\$ \mathbb{Z}_p$. When the adversary produces $m, \mathsf{Urnd}_1, \mathsf{Urnd}_2$ that break the property, it must hold that $\mathsf{msgS1}_1 \cdot g^{\alpha_1} \cdot g^{-\beta_1 x} = g^r \cdot g^{\alpha_2} \cdot g^{-\beta_2 x}$ which the reduction can solve for $\mathrm{dlog}_g \mathsf{msgS1} = \mathrm{dlog}_g \mathbf{ch} = r + \alpha_2 - x \cdot \beta_2 - \alpha_1 + \beta_1 \cdot x$.

Overall, this means that $\varepsilon_{\mathsf{UL}} \leq 2 \cdot \varepsilon_{\mathsf{DLP}}$.

**Property 6:** holds for the Blind Schnorr scheme. To see this, assume an adversary provides two distinct signatures $\sigma_1 \coloneqq (c_1, s_1)$ and $\sigma_2 \coloneqq (c_2, s_2)$ for the same message $m$. The verification procedure $\mathcal{V}(\mathsf{pk}, m, \sigma_i)$ ensures that $\mathsf{H}(g^{s_i} \cdot \mathsf{pk}^{-c_i}, m) = c_i$. If both signatures correspond to the same hash, it must hold that $c_1 = c_2$ and $g^{s_1} \cdot \mathsf{pk}^{-c_1} = g^{s_2} \cdot \mathsf{pk}^{-c_2}$. However, $c_1 = c_2 \Rightarrow s_1 = s_2$; otherwise, $g^{s_1} \cdot \mathsf{pk}^{-c_1} \neq g^{s_2} \cdot \mathsf{pk}^{-c_2}$. This is a contradiction.

## E Abe-Okamoto Signatures as an Instantiation of Three-Move Simulatable Blind Signature Schemes

In this section, we discuss the compatibility between our definition for three-move simulatable blind signatures (Section 4) and Blind Signature Schemes based on the OR-Proof technique [26]. In this section, we study the Abe-Okamoto scheme (AO) [3]. Similar arguments apply to the other OR-proof based schemes such as Abe [1].

First, we write down the scheme in the style of Definition 4.

**Construction 8 (Abe-Okamoto Blind Signature).** For a security parameter $\lambda$, let $\mathsf{pp} \coloneqq (\mathbb{G}, p, g)$ be a public parameter specifying a group $\mathbb{G}$ of order $p$ and a generator $g$, and $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p$ and $\mathcal{F} : \{0,1\}^* \to \mathbb{G}$ be hash functions treated as random oracles. Let $\mathsf{info} \in \{0,1\}^*$ be a common string. The Abe-Okamoto blind signature scheme $\mathsf{BS} = (\mathcal{G}, \mathcal{S}, \mathcal{SIM}_{\mathcal{S}}, \mathcal{U}, \mathcal{V})$ is defined as follows:

- **Key Generation** $\mathcal{G}(\mathsf{pp})$: Sample $x \leftarrow_{\$} \mathbb{Z}_p$, compute $y \coloneqq g^x$, and output $(\mathsf{pk}, \mathsf{sk}) \coloneqq (y, x)$.
- **Signer Algorithms** $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$:
  - $\mathcal{S}_1^{\mathsf{H}}(\mathsf{sk})$: Sample $u, s, d \leftarrow_{\$} \mathbb{Z}_p$. Compute $z \coloneqq \mathcal{F}(\mathsf{info})$, $a \coloneqq g^u$, and $b \coloneqq g^s \cdot z^d$. Output $\mathsf{msgS1} \coloneqq (a, b)$ and $\mathsf{rnd_S} \coloneqq (u, s, d)$.
  - $\mathcal{S}_2^{\mathsf{H}}(\mathsf{sk}, \mathsf{rnd_S}, \mathsf{msgU})$: Parse $\mathsf{rnd_S}$ as $(u, s, d)$ and $\mathsf{msgU}$ as $e$. Compute $c \coloneqq e - d \pmod p$ and $r \coloneqq u - c \cdot x \pmod p$. Output $\mathsf{msgS2} \coloneqq (r, c, s, d)$.
- **User Algorithms** $\mathcal{U}^{\mathsf{H}} = (\mathcal{U}_1^{\mathsf{H}}, \mathcal{U}_2^{\mathsf{H}})$:
  - $\mathcal{U}_1^{\mathsf{H}}(\mathsf{pk}, m, \mathsf{msgS1}; \mathsf{Urnd})$: Parse $\mathsf{msgS1}$ as $(a, b)$ and $\mathsf{Urnd}$ as $(t_1, t_2, t_3, t_4) \in \mathbb{Z}_p^4$. Compute $z \coloneqq \mathcal{F}(\mathsf{info})$, $\alpha \coloneqq a \cdot g^{t_1} \cdot y^{t_2}$, and $\beta \coloneqq b \cdot g^{t_3} \cdot z^{t_4}$. Query $\varepsilon \coloneqq \mathsf{H}(\alpha, \beta, z, m)$. Compute $e \coloneqq \varepsilon - t_2 - t_4 \pmod p$ and output $\mathsf{msgU} \coloneqq e$.
  - $\mathcal{U}_2^{\mathsf{H}}(\mathsf{pk}, m, \mathsf{Urnd}, \mathsf{msgS2})$: Parse $\mathsf{Urnd}$ as $(t_1, t_2, t_3, t_4)$ and $\mathsf{msgS2}$ as $(r, c, s, d)$. Compute $\rho \coloneqq r + t_1 \pmod p$, $\omega \coloneqq c + t_2 \pmod p$, $\sigma \coloneqq s + t_3 \pmod p$, and $\delta \coloneqq d + t_4 \pmod p$. Output $\sigma \coloneqq (\rho, \omega, \sigma, \delta)$.
- **Verification** $\mathcal{V}^{\mathsf{H}}(\mathsf{pk}, m, \sigma)$: Parse $\sigma$ as $(\rho, \omega, \sigma, \delta)$. Compute $z \coloneqq \mathcal{F}(\mathsf{info})$. Output 1 if $\omega + \delta \pmod p \equiv \mathsf{H}(g^\rho \cdot y^\omega, g^\sigma \cdot z^\delta, z, m)$, and 0 otherwise.
- **Signer Simulation Algorithms** $\mathcal{SIM}_{\mathcal{S}}^{\mathsf{H}} = (\mathcal{SIM}_{\mathcal{S}_1}^{\mathsf{H}}, \mathcal{SIM}_{\mathcal{S}_2}^{\mathsf{H}}, \mathcal{SIM}_{\mathsf{H}})$:
  - $\mathcal{SIM}_{\mathcal{S}_1}(\mathsf{pk})$: By programming the random oracle $\mathcal{F}(\mathsf{info}) \coloneqq g^w$, the algorithm knows $\mathrm{dlog}_g z = w$. Sample $v, r, c \leftarrow_{\$} \mathbb{Z}_p$. Compute $a \coloneqq g^r \cdot y^c$ and $b \coloneqq g^v$. Output $\mathsf{msgS1} \coloneqq (a, b)$ and $\mathsf{simSt} \coloneqq (w, v, r, c)$.
  - $\mathcal{SIM}_{\mathcal{S}_2}(\mathsf{pk}, \mathsf{msgS1}, \mathsf{simSt}, \mathsf{Urnd}, m)$: Parse $\mathsf{simSt}$ as $(w, v, r, c)$. Run the User algorithm to obtain the challenge: $e \leftarrow \mathcal{U}_1^{\mathcal{SIM}_{\mathsf{H}}}(\mathsf{pk}, m, \mathsf{msgS1}; \mathsf{Urnd})$. Compute $d \coloneqq e - c \pmod p$ and $s \coloneqq v - d \cdot w \pmod p$. Output $\mathsf{msgS2} \coloneqq (r, c, s, d)$.
  - $\mathcal{SIM}_{\mathsf{H}}(\mathsf{pk}, \mathsf{msgS1}, \mathsf{simSt}, \mathsf{Urnd}, m)$: Answer via lazy sampling. Upon fresh queries, sample and return $\varepsilon \leftarrow_{\$} \mathbb{Z}_p$.

Now, we show that AO satisfies the properties required by Definition 4.

**Property 1:** Since both $\mathcal{U}_1$ and $\mathcal{V}$ make a single hash query to the challenge random oracle $\mathsf{H}$. This hash query contains the signed message $m$, and a function $\mathsf{blind}(\mathsf{msgS1}, \mathsf{Urnd})$ of the signer's first message $\mathsf{msgS1} = (a, b)$ and the user's randomness $\mathsf{Urnd} = (t_1, t_2, t_3, t_4)$, where $\mathsf{blind}((a, b), (t_1, \ldots, t_4)) \coloneqq (\alpha = a \cdot g^{t_1} \cdot y^{t_2}, b \cdot g^{t_3} \cdot z^{t_4})$. Finally, $\alpha$ is derived from the public key $\mathsf{pk} = y$ and other values. Consequently, in an honest protocol execution, the tuple $(\mathsf{pk}, m, \mathsf{msgS1}, \mathsf{Urnd})$ determines the hash.

**Property 2:** An adversary outputting a valid signature without having queried the respective hash query previously is essentially guessing the output of the oracle $\mathsf{H}$ successfully. To see this, let $(\rho, \omega, \sigma, \delta)$ be a signature for a message $m$

under $\mathsf{pk}$. If the signature is valid, the adversary has successfully guessed the output of $\mathsf{H}$ on input $Q = (g^{\rho} \cdot y^{\omega}, g^{\sigma} \cdot z^{\delta}, z, m)$ since $\omega + \delta = \mathsf{H}(Q)$ holds. Since the output of $\mathsf{H}$ is uniformly random in $\mathbb{Z}_p$, the probability of finding such a signature is at most $\frac{1}{p} =: \varepsilon_{\mathsf{SHL}}$.

**Property 3:** The first user message $\mathsf{msgU} = e := \mathsf{H}(Q) - t_2 - t_4$ is a blinded version of the hash query. Similar to the argument above, outputting a user message without having queried the random oracle $\mathsf{H}$ is equivalent to guessing the output of $\mathsf{H}$, which is possible with probability at most $\frac{1}{p} =: \varepsilon_{\mathsf{UUF}}$.

**Property 4:** Breaking this property by an adversary implies that the adversary can solve the discrete logarithm problem. First, recall that the adversary outputs $(m_1, m_2, \mathsf{Urnd}_1, \mathsf{Urnd}_2)$ with $\mathcal{U}_1(\mathsf{pk}, m_1, \mathsf{msgS1}; \mathsf{Urnd}_1) = \mathcal{U}_1(\mathsf{pk}, m_2, \mathsf{msgS1}; \mathsf{Urnd}_2)$, where $\mathsf{msgS1} = (a, b)$ and $\mathsf{Urnd}_i = (t_{1,i}, \ldots, t_{4,i})$. It follows

$$\begin{aligned} &\mathsf{H}(a \cdot g^{t_{1,1}} \cdot \mathsf{pk}^{t_{2,1}}, b \cdot g^{t_{3,1}} \cdot z^{t_{4,1}}, z, m_1) - t_{2,1} - t_{4,1} \\ =\, &\mathsf{H}(a \cdot g^{t_{1,2}} \cdot \mathsf{pk}^{t_{2,2}}, b \cdot g^{t_{3,2}} \cdot z^{t_{4,2}}, z, m_2) - t_{2,2} - t_{4,2}. \end{aligned}$$

We distinguish the following cases:

- **Case 1.** $\mathsf{hq}(\mathsf{pk}, m_1, \mathsf{msgS1}, \mathsf{Urnd}_1) = \mathsf{hq}(\mathsf{pk}, m_2, \mathsf{msgS1}, \mathsf{Urnd}_2)$. In this case, it holds that $m_1 = m_2$, $\alpha_1 = \alpha_2$, and $\beta = \beta_2$. Consequently, it must hold that $\mathsf{Urnd}_1 \neq \mathsf{Urnd}_2$, hence $t_{j,1} \neq t_{j,2}$ for some $j$. Again, we distinguish two cases:
  - $j = 1 \vee j = 2$. It follows that $g^{t_{1,1}} \cdot \mathsf{pk}^{t_{2,1}} = g^{t_{1,2}} \cdot \mathsf{pk}^{t_{2,2}}$. In this case, a reduction embedding a discrete logarithm challenge $\mathbf{ch}$ in $\mathsf{pk}$ can solve $\mathrm{dlog}_g \mathbf{ch} = \mathrm{dlog}_g \mathbf{ch} = \frac{t_{1,1} - t_{1,2}}{t_{2,2} - t_{2,1}}$. This equality is solvable since $t_{2,1} \neq t_{2,2}$ (note that this follows from $t_{1,1} \neq t_{1,2}$, too; otherwise, $\alpha_1 \neq \alpha_2$).
  - $j = 3 \vee j = 4$. It follows that $g^{t_{3,1}} \cdot z^{t_{4,1}} = g^{t_{3,2}} \cdot z^{t_{4,2}}$. In this case, a reduction embedding a discrete logarithm challenge $\mathbf{ch}$ in $z$ by programming the random oracle $\mathcal{F}$ (while randomizing $z$ for each fresh query) can solve $\mathrm{dlog}_g z = \mathrm{dlog}_g \mathbf{ch} = \frac{t_{3,1} - t_{3,2}}{t_{4,2} - t_{4,1}}$. This equality is defined since $t_{4,1} \neq t_{4,2}$ (this follows from $t_{3,1} \neq t_{3,2}$, too; otherwise, $\beta_1 \neq \beta_2$).

  The adversary's advantage in provoking this case is bounded by $2 \cdot \varepsilon_{\mathsf{DLP}}$
- **Case 2.** $\mathsf{hq}(\mathsf{pk}, m_1, \mathsf{msgS1}, \mathsf{Urnd}_1) \neq \mathsf{hq}(\mathsf{pk}, m_2, \mathsf{msgS1}, \mathsf{Urnd}_2)$, and the adversary did not make one of the hash queries. In this case, the adversary succeeds with probability at most $\frac{1}{p}$.
- **Case 3.** $\mathsf{hq}(\mathsf{pk}, m_1, \mathsf{msgS1}, \mathsf{Urnd}_1) \neq \mathsf{hq}(\mathsf{pk}, m_2, \mathsf{msgS1}, \mathsf{Urnd}_2)$ and the adversary makes both hash queries to the oracle $\mathsf{H}$. We construct a reduction embedding a discrete logarithm challenge $\mathbf{ch}$ in $\mathsf{pk}$ or $z$ and solving for $\mathrm{dlog}_g \mathbf{ch}$. Assume w.l.o.g. $\mathsf{hq}(\mathsf{pk}, m_1, \mathsf{msgS1}, \mathsf{Urnd}_1)$ is the first hash query the adversary makes and let $\mathsf{msgU} := \mathcal{U}_1(\mathsf{pk}, m_1, \mathsf{msgS1}_1; \mathsf{Urnd}_1)$. Upon the second hash query, the reduction answers the second query and the adversary produces $(\mathsf{pk}, m_2, \mathsf{msgS1}, \mathsf{Urnd}_2)$. Then the reduction rewinds the adversary to the point before the second query and reprograms the random oracle $\mathsf{H}$. The adversary then outputs another tuple $(\mathsf{pk}, m_2', \mathsf{msgS1}, \mathsf{Urnd}')$. The reduction proceeds similar to Case 1 and computes $\mathrm{dlog}_g \mathsf{pk}$ or $\mathrm{dlog}_g z$, depending on which subcase occurs.

Note that for Cases 1 and 3, the reduction does not know which subcase occurs before the adversary terminates. Therefore, it flips a coin and embeds either in $\mathsf{pk}$ or $z$; therefore, the probability that it extracts successfully for each subcase is $\frac{1}{2}$. Therefore, the advantage of the adversary in violating this property is bounded by $\varepsilon_{\mathsf{UUO}} \coloneqq \frac{1}{p} + 2 \cdot \varepsilon_{\mathsf{DLP}} + \sqrt{\mathsf{Q}_{\mathsf{H}} \cdot \varepsilon_{\mathsf{DLP}}}$.

**Property 5:** holds since violating this property is as hard as solving the discrete logarithm problem. Let $\mathcal{A}$ be an adversary violating this property by outputting a tuple $(m, \mathsf{Urnd}_1, \mathsf{Urnd}_2)$ with $\mathsf{Urnd}_i = (t_{1,i}, \ldots, t_{4,i})$ satisfying $\mathsf{hq}(\mathsf{pk}, m, \mathsf{msgS1}_1, \mathsf{Urnd}_1) = \mathsf{hq}(\mathsf{pk}, m, \mathsf{msgS1}_2, \mathsf{Urnd}_2)$ for some public key $\mathsf{pk}$ and a signer's first messages $\mathsf{msgS1}_i = (a_i, b_i)$. It follows that

$$\begin{aligned} &\mathsf{H}(a_1 \cdot g^{t_{1,1}} \cdot \mathsf{pk}^{t_{2,1}}, b_1 \cdot g^{t_{3,1}} \cdot z^{t_{4,1}}, z, m) \\ =&\mathsf{H}(a_2 \cdot g^{t_{1,2}} \cdot \mathsf{pk}^{t_{2,2}}, b_2 \cdot g^{t_{3,2}} \cdot z^{t_{4,2}}, z, m). \end{aligned}$$

We distinguish two cases:

- $\mathsf{msgS1}_1 = \mathsf{msgS1}_2$. In this case, a reduction takes a $\mathsf{DLP}$ challenge $\mathbf{ch}$ as input, it flips a bit $B \leftarrow_\$ \{0,1\}$ and does the following
  - If $B = 0$, the reduction extracts $\mathrm{dlog}_g \mathbf{ch} = \frac{t_{1,1} - t_{1,2}}{t_{2,2} - t_{2,1}}$ successfully if $t_{2,1} \neq t_{2,2}$.
  - If $B = 1$, the reduction embeds the challenge $\mathbf{ch}$ in $z$ by programming the random oracle $\mathcal{F}$ (with the necessary randomization for each fresh query). It simulates the signing algorithm normally using the secret key. The reduction extracts $\mathrm{dlog}_g \mathbf{ch} = \mathrm{dlog}_g z = \frac{t_{3,1} - t_{3,2}}{t_{4,2} - t_{4,1}}$ if $t_{4,2} \neq t_{4,1}$.
- $\mathsf{msgS1}_1 \neq \mathsf{msgS1}_2$. This implies that either $a_1 \neq a_2$ or $b_1 \neq b_2$. A reduction takes a $\mathsf{DLP}$ challenge $\mathbf{ch}$ as input, generates the keypair $(\mathsf{pk}, \mathsf{sk})$ itself, programs the random oracle $z \coloneqq \mathcal{F}(\mathsf{info}) = g^w$ for a known $w \leftarrow_\$ \mathbb{Z}_p$, and flips a coin $B' \leftarrow_\$ \{0,1\}$ to guess the collision type:
  - **Case** $B' = 0$ **(Assumes** $a_1 \neq a_2$**):** The reduction embeds the challenge in the first component of $\mathsf{msgS1}_1$ by setting $a_1 \coloneqq \mathbf{ch}$. It generates $\mathsf{msgS1}_2$ with $a_2 \coloneqq g^r$. The components $b_1$ and $b_2$ are generated honestly. From the collision $a_1 \cdot g^{t_{1,1}} \cdot \mathsf{pk}^{t_{2,1}} = a_2 \cdot g^{t_{1,2}} \cdot \mathsf{pk}^{t_{2,2}}$, the reduction extracts $\mathrm{dlog}_g \mathbf{ch} = \mathrm{dlog}_g a_1 = r + t_{1,2} - t_{1,1} + \mathsf{sk} \cdot (t_{2,2} - t_{2,1}) \pmod p$.
  - **Case** $B' = 1$ **(Assumes** $a_1 = a_2 \wedge b_1 \neq b_2$**):** The reduction generates $a_1$ and $a_2$ honestly. It embeds the challenge in the second component of $\mathsf{msgS1}_1$ by setting $b_1 \coloneqq \mathbf{ch}$, and generates $b_2 \coloneqq g^r$. From the collision $b_1 \cdot g^{t_{3,1}} \cdot z^{t_{4,1}} = b_2 \cdot g^{t_{3,2}} \cdot z^{t_{4,2}}$, the reduction extracts $\mathrm{dlog}_g \mathbf{ch} = \mathrm{dlog}_g b_1 = r + t_{3,2} - t_{3,1} + w \cdot (t_{4,2} - t_{4,1}) \pmod p$.

Overall, $\mathcal{A}$ violates this property with probability at most $4 \cdot \varepsilon_{\mathsf{DLP}} + \frac{1}{p} =: \varepsilon_{\mathsf{UL}}$.

**Property 6:** Recall that a signature-message pair $(m, (\rho, \omega, \sigma, \delta))$ is valid if $\omega + \delta = \mathsf{H}(g^\rho \cdot \mathsf{pk}^\omega, g^\sigma \cdot z^\delta, z, m)$. This verification equation does not directly tie a hash value to a specific message-signature pair. For example, knowing the secret key $\mathsf{sk}$ and $\mathrm{dlog}_g z$, one can generate multiple signatures that are tied to the same hash value as follows: Choose $\omega'$ and $\delta'$ arbitrarily such that $\delta' \neq \delta$ and $\omega \neq \omega'$ with $\omega + \delta = \mathsf{H}(g^\rho \cdot \mathsf{pk}^\omega, g^\sigma \cdot z^\delta, z, m) = \omega' + \delta'$. Then compute

$\rho' \coloneqq \rho + (\omega - \omega') \cdot \mathsf{sk}$ and $\sigma' \coloneqq \sigma + (\delta - \delta') \cdot \mathrm{dlog}_g z$, and output the signature-message pair $(m, (\rho', \omega', \sigma', \delta'))$. Obviously, this is a valid pair since $\delta' + \omega' = \delta + \omega = \mathsf{H}(g^{\rho} \cdot \mathsf{pk}^{\omega}, g^{\sigma} \cdot z^{\delta}, z, m) = \mathsf{H}(g^{\rho'} \cdot \mathsf{pk}^{\omega'}, g^{\sigma'} \cdot z^{\delta'}, z, m)$. However, such behaviour is only possible if $\mathsf{sk}$ and $\mathrm{dlog}_g z$ are known; therefore, no user should be able to find a pair of signatures linked to the same hash query, which we show in the following.

Let $\mathcal{A}$ be an adversary playing the double signature game described by Property 6. Assume $\mathcal{A}$ outputs two distinct valid message-signature pairs $(m_1, (\rho_1, \omega_1, \sigma_1, \delta_1))$ and $(m_2, (\rho_2, \omega_2, \sigma_2, \delta_2))$ with $(m_1, (\rho_1, \omega_1, \sigma_1, \delta_1)) \neq (m_2, (\rho_2, \omega_2, \sigma_2, \delta_2))$ and $\omega_1 + \delta_1 = \omega_2 + \delta_2$, hence it wins the game. We provide a reduction $\mathcal{R}_5$ that solves the discrete logarithm problem.

Given a discrete logarithm challenge $\mathbf{ch}$, $\mathcal{R}_5$ behaves as follows: It sets $\mathsf{pk} \coloneqq \mathbf{ch}$ and invokes $\mathcal{A}(\mathsf{pk})$ with access to oracles $\mathcal{SIM}^{\mathsf{H}}_{\mathcal{S}}, \mathcal{F}$, and $\mathsf{H}$. $\mathcal{R}_5$ simulates the oracles as follows:

- $\mathcal{SIM}^{\mathsf{H}}_{\mathcal{S}}$. As described by *construction* 8.
- $\mathsf{H}(Q)$. Via Lazy sampling.
- $\mathcal{F}(Q)$. $\mathcal{SIM}^{\mathsf{H}}_{\mathcal{S}}$ initially programms $\mathcal{F}$ on the tag $\mathsf{info}$ as $\mathcal{F}(\mathsf{info}) = g^w = z$. Further queries by the adversary are answered via lazy sampling.

Assume $\mathcal{A}$ terminates and outputs two distinct forgeries $(m_1, (\rho_1, \omega_1, \sigma_1, \delta_1))$ and $(m_2, (\rho_2, \omega_2, \sigma_2, \delta_2))$ linked to the same hash query. Consequently, it holds that $\omega_1 + \delta_1 = \omega_2 + \delta_2$, Hence, we have $g^{\rho_1} \cdot \mathsf{pk}^{\omega_1} = g^{\rho_2} \cdot \mathsf{pk}^{\omega_2}$, $g^{\sigma_1} \cdot z^{\delta_1} = g^{\sigma_2} \cdot z^{\delta_2}$, and $m_1 = m_2$. It follows that

$$\rho_1 + \mathrm{dlog}_g \mathsf{pk} \cdot \omega_1 = \rho_2 + \mathrm{dlog}_g \mathsf{pk} \cdot \omega_2,$$

hence

$$\mathrm{dlog}_g \mathsf{pk} = \frac{\rho_1 - \rho_2}{\omega_2 - \omega_1}. \tag{2}$$

This equation is solvable if $\omega_2 \neq \omega_1$.

Since the signatures are distinct, one of the following cases must hold.

- **Case 1.** $\omega_1 \neq \omega_2$. In this case, Equation (2) is defined and solvable for $\mathrm{dlog}_g \mathsf{pk}$, so the reduction succeeds.
- **Case 2.** $\omega_1 = \omega_2$. Since $\omega_1 + \delta_1 = \omega_2 + \delta_2$, it follows that $\delta_1 = \delta_2$. Since $g^{\rho_1} \cdot \mathsf{pk}^{\omega_1} = g^{\rho_2} \cdot \mathsf{pk}^{\omega_2}$ and $\omega_1 = \omega_2$, it follows that $g^{\rho_1} = g^{\rho_2}$, which implies $\rho_1 = \rho_2$. Similarly, since $g^{\sigma_1} \cdot z^{\delta_1} = g^{\sigma_2} \cdot z^{\delta_2}$ and $\delta_1 = \delta_2$, it follows that $g^{\sigma_1} = g^{\sigma_2}$, which implies $\sigma_1 = \sigma_2$.
  Consequently, $(\rho_1, \omega_1, \sigma_1, \delta_1) = (\rho_2, \omega_2, \sigma_2, \delta_2)$. This contradicts the premise that the adversary outputs two distinct signatures. Therefore, this case is impossible.

Consequently, Eq. (2) is defined and $\mathcal{R}_5$ can compute $\mathrm{dlog}_g \mathbf{ch}$ successfully and win Game $\mathsf{DLP}$. Thus, it holds $\varepsilon_{\mathsf{DS}} \coloneqq \varepsilon_{\mathsf{DLP}}$.

# F Partially Opening Blinded Pedersen Commitments

## F.1 Proof of Same Discrete Logarithm

**Construction 9 (Sigma Protocol for Proof of Same Discrete Logarithm).** We describe how a prover $\mathcal{P}$ can prove in zero knowledge that the elements of a vector of group elements $\boldsymbol{h} = (h_1, \ldots, h_n)$ have the same discrete logarithms to the bases of another vector of group elements $\boldsymbol{h'} = (h'_1, \ldots, h'_n)$. That is $\text{dlog}_{h_1} h'_1 = \ldots = \text{dlog}_{h_n} h'_n$.

$\mathcal{P}_1(x = (\boldsymbol{h}, \boldsymbol{h}'), w = \gamma)$ The prover takes as input two vectors of the same length and a witness $\gamma$ which is the discrete logarithm relation between them. It samples $r_{sdl} \leftarrow_\$ \mathbb{Z}_p$ and sets $\boldsymbol{h}_{sdl} = \boldsymbol{h}^{r_{sdl}}$, i.e. the component wise exponentiation of $\boldsymbol{h}$ with $r_{sdl}$. It outputs the commitment vector $\boldsymbol{h_{sdl}}$ to the verifier.

$\mathcal{V}_1(x = (\boldsymbol{h}, \boldsymbol{h}'), \boldsymbol{h_{sdl}})$ The verifier takes as input the statement and the prover's first message. It samples a random value $c \leftarrow_\$ \mathbb{Z}_p$ and outputs it to $\mathcal{P}$.

$\mathcal{P}_2(x = (\boldsymbol{h}, \boldsymbol{h}'), w = \gamma, r_{sdl}, c)$ The prover takes as input the statement $\boldsymbol{h}, \boldsymbol{h}'$, the witness $\gamma$, its internal state $r_{sdl}$ and the verifier's challenge $c$. It computes $s_{sdl} = r_{sdl} - c \cdot \gamma$.

$\mathcal{V}_2(x = (\boldsymbol{h}, \boldsymbol{h}'), \boldsymbol{h}_{sdl}, c, s_{sdl})$ The verifier checks that $\boldsymbol{h}_{sdl} = \boldsymbol{h}^{s_{sdl}} \cdot \boldsymbol{h}'^{c}$ where exponentiation is taken component wise. It returns 1 if the above equation holds and 0 otherwise.

**Lemma 8 (Soundness of construction 9).** *The protocol from construction 9 is $\frac{1}{p}$-sound.*

*Proof.* Let $\boldsymbol{h}, \boldsymbol{h}'$ be two vectors such that there exists indices $i, j$ such that $\gamma_i = \text{dlog}_{h_i} h'_i \neq \text{dlog}_{h_j} h'_j = \gamma_j$. Let $\boldsymbol{h}^*_{sdl}$ be an adversarial prover's first message. Let $r_i = \text{dlog}_{h_i} h^*_{sdl,i}$ and $r_j = \text{dlog}_{h_j} h^*_{sdl,j}$. Then, there exists at most one $c^* \in \mathbb{Z}_p$ such that there exists a $s_{sdl}$ such that $r_i = s_{sdl} + \gamma_i \cdot c^*$ and $r_j = s_{sdl} + \gamma_j \cdot c^*$. The probability that the hones verifier samples $c = c^*$ is $\frac{1}{p}$.

**Lemma 9 (Special Knowledge Soundness of construction 9).** *The protocol from construction 9 is 2-special sound.*

*Proof.* Given two transcripts $\boldsymbol{h}_{sdl}, c_1, s_1$ and $\boldsymbol{h_{sdl}}, c_2, s_2$ with the same initial message $\boldsymbol{h}_{sdl}$, the extractor computes $\gamma = \frac{s_1 - s_2}{c_2 - c_1}$.

**Lemma 10 (Extraction in the AGM).** *Let $\mathcal{P}^*$ be an algebraic adversary acting as the prover in the protocol from construction 9 that interacts with a verifier. Then there exists an extractor $\mathcal{E}_{alg}$ that extracts a witness $\gamma$ with probability $1 - \frac{1+\nu}{p}$ from the algebraic adversary.*

*Proof.* The algebraic prover takes as input the group elements $\boldsymbol{h}$. It outputs a vector $\boldsymbol{h}'$ along with algebraic representations $\boldsymbol{a}_i$ s.t. $h'_i = \prod h_i^{a_i}$. For each proof transcript, the prover outputs a vector $\boldsymbol{h}_{sdl}$ along with algebraic representations

$\{\boldsymbol{z}_i\}_{i\in[\nu]}$ for each element $h_{sdl,i}$ to the basis $\boldsymbol{h}$. the verifier then picks a challenge $c \leftarrow_\$ \mathbb{Z}_p$ uniformly at random.

The prover responds with $s$ such that for all $i \in [\nu]$, it holds that $h_i^s \cdot h_i'^c = h_{sdl,i}$.

By lemma 8, the probability that the prover succeeds in outputting a valid response when $\boldsymbol{h}, \boldsymbol{h}'$ are not in the language is $\frac{1}{p}$. In the following, we assume that this doesn't happen.

This yields $\nu$ linear equations of the form

$$\sum_{j=1}^{\nu} z_{i,j} \cdot \mathrm{dlog} h_j = s \cdot \mathrm{dlog} h_i + c \cdot \sum_{j=1} a_{i,j} \cdot \mathrm{dlog} h_i \tag{3}$$

where the discrete logarithms are unknown as well as $\nu$ equations of the form $\gamma \cdot \mathrm{dlog} h_i = \sum_{j=1} a_{i,j} \cdot \mathrm{dlog} h_i$ where both $\gamma$ and the discrete logarithms are unknown. If no fixed generator is given for the group, we assume w.l.o.g. that discrete logarithms are taken to the base $h_1$ (i.e.$\mathrm{dlog} h_1 = 1$).

W.l.o.g we assume that all $h_i$ are pairwise distinct and no $h_i$ is the neutral element. Otherwise, we conflate the equal elements as well as their representations.

We consider the following cases:

1. $\exists i \colon a_{i,i} \neq 0 \wedge \forall i' \neq i \colon a_{i,i'} = 0$: In this case, the extractor outputs the witness $\gamma = a_i$.
2. The system of linear equations is linearly dependent. We consider the following subcases:
   a. The matrix $A$ consisting of the representation vectors $\boldsymbol{a}_i$ has full rank. We make an information-theoretic argument why the system of equations has to be linearly dependent with probability at most $\frac{\nu}{p}$ or with probability 1 pver the choice of $c$.
   First, we note that information-theoretically, the choice of $c$ already fixes $s$. Also, information-theoretically $\gamma$ is fixed by the vectors $\boldsymbol{h}$ and $\boldsymbol{h}'$ already, as well as the adversary's implicit choice of $r_{sdl}$ through $\boldsymbol{h}_{sdl}$.
   The determinant of the matrix $A$ with $c, s$ as formal variables is a polynomial of total degree at most $n$. As $c$ information-theoretically fixes $s$, we can reduce the number of formal variables in the determinant polynomial to 1 by replacing $s$ with $r_{sdl} - c \cdot \gamma$. This yields a polynomial in one variable $c$ of degree at most $n$. If this polynomial is the constant 0 polynomial, the probability of linear dependence is 1, i.e. for any pair $c, s$ that fulfills the linear relation above, the matrix does not have full rank. In this case the extractor picks a random $c' \leftarrow_\$ \mathbb{Z}_p$. It plugs it into the matrix and computes the determinant of the matrix with $s$ as a variable (but not $c$). It then computes the roots $s'$ of the resulting polynomial. This yields at most $\nu$ possible values of $s'$. The extractor tests (using the verification equation of the $\Sigma$-protocol) which one is the correct one. It then solves for the witness $\gamma = \frac{s-s'}{c'-c}$. With probability $1 - \frac{1}{p}$ it holds that $c' \neq c$ and thus the extraction succeeds.

If it is not the 0 polynomoial, by the Schwartz-Zippel-Lemma, the probability of finding a root of the polynomial when randomly choosing $c$ is at most $\frac{n}{p}$. In this case, the extractor cannot extract, so this happens with probability $\frac{\nu}{p}$.

b. The matrix $A$ consisting of the representation vectors $\boldsymbol{a}_i$ does not have full rank. In this case, there exists a linear combination $\boldsymbol{b}$ such that $\sum_{i=1}^{\nu} b_i \cdot \boldsymbol{a}_i = 0$. Due to soundness of the sigma protocol (implying correctness of the statement), this means that with probability $1 - \frac{1}{p}$ it holds that $\prod_{i=1}^{\nu} h_i^{b_i} = 0$. We therefore know a way to express one of the $h_i$ as a linear combination of the others. We remove this $h_i$ from all systems of equations and replace it with its linear representation of the other $h_i$. This yields systems of equations with one variable less. We consecutively apply this transformation until the matrix $A'$ resulting from these transformations has full rank. If $A'$ has rank 1, we proceed as in Case 1. Otherwise we continue as in Case 2.a.

3. Otherwise: As we excluded Case 1 and Case 2, the system of equations is linearly independent. The extractor can thus compute a the discrete logarithms of all $h_i$ w.r.t. $h_1$ by solving the linear equation system. It can use the representation of $h'_1$ to compute $\gamma = \mathrm{dlog}_{h_1} h'_1$ and output $\gamma$.

Overall, the failure probability of the extractor can be bounded by $\frac{\nu}{p} + \frac{1}{p}$.

**Lemma 11 (HVZK of construction 9).** *The protocol from construction 9 fullfills special honest verifier zero-knowledge.*

*Proof.* We give a simulator. On input of a challenge $c$ and a statement $\boldsymbol{h}, \boldsymbol{h}'$, the simulator samples $s_{sdl} \leftarrow_{\$} \mathbb{Z}_p$ and computes $\boldsymbol{h}_{sdl} \coloneqq \boldsymbol{h}^{s_{sdl}} \cdot \boldsymbol{h'}^{c}$ where exponentiation and multiplication is component wise. It is easy to see that for $\boldsymbol{h}, \boldsymbol{h}'$ where there exists $\gamma$ such that $\boldsymbol{h}^{\gamma} = \boldsymbol{h}'$ the transcripts generated by the simulator are identically distributed to real transcripts that use $c$ as a challenge.

### F.2 Proofs of Partial Openings

**Construction 10 (Protocol for Partially Opening a Pedersen Commitment).** We describe a sigma protocol that allows for partially opening a generalized Pedersen Commitment while proving knowledge of the remainder of the opening.

Let $\boldsymbol{h}$ be the base vector of the Pedersen commitment, and $\boldsymbol{a}$ be the full opening. Let $C$ be the Pedersen commitment obtained by $C \coloneqq \prod_{i=1}^{n} h_i^{a_i}$. For simplicity, we treat the randomness used for committing as part of the full opening. Let $I$ be a set of indices where the commitment is to be opened.

$\mathcal{P}_1(\boldsymbol{h}, \boldsymbol{a}, I)$ For all $j \notin I$, the prover samples $r_j \leftarrow_{\$} \mathbb{Z}_p$. It then computes $R \coloneqq \prod_{j \notin I} h_j^{r_j}$. It outputs $R$ and $A_I \coloneqq \{a_i \colon i \in I\}$ to the verifier.

$\mathcal{V}_1(\boldsymbol{h}, C, A_I, R)$ The verifier takes the prover's first message as input along with the commitment and the parameters $\boldsymbol{h}$ and samples $c \leftarrow_{\$} \mathbb{Z}_p$. It sends $c$ to the prover.

$\mathcal{P}_2(\boldsymbol{h}, \boldsymbol{a}, I, \{r_j : j \notin I\}, c)$ The prover takes the commitment parameters $\boldsymbol{h}$, the opening $\boldsymbol{a}$, the opening indices $i$, and the randomness for his first message $r_j$, along with the challenge from the verifier and for all $j \notin I$ computes $s_j \coloneqq r_j - c \cdot a_j$. It then outputs $S = \{s_j \colon j \notin I\}$ to the verifier.

$\mathcal{V}_2(\boldsymbol{h}, C, A_I, R, c, S)$ The verifier computes $C' = \frac{C}{\prod_{i \in I} h_i^{a_i}}$ and then checks that $R = C'^c \cdot \prod_{j \notin I} h_j^{s_j}$. It outputs 1 if this equality holds and 0 otherwise.

**Lemma 12 (2-special soundness of construction 10).** *The protocol from construction 10 is 2-special sound.*

*Proof.* Given two transcripts $(R, c_1, S_1)$ and $(R, c_2, S_2)$ with the same initial message, the extractor $\mathcal{E}$ computes the witness $a_i$ for each $i \in I$ as $a_i = \frac{s_{1,i} - s_{2,i}}{c_2 - c_1}$.

**Lemma 13 (HVZK of construction 10).** *The protocol from construction 10 is HVZK.*

*Proof.* The simulator works similar to that from lemma 11.

**Definition 24.** *Let $\boldsymbol{h}$ be the public parameters of a Pedersen commitment and $g$ the generator of the group over which the Pedersen commitment is instantiated. For a Pedersen commitment $C$, $C^\gamma \cdot g^{r\gamma}$ is a* blinded *Pedersen commitment with public parameters $\boldsymbol{h}^\gamma, g^\gamma$.*

**Lemma 14 (Extraction in the AGM).** *Let $\mathcal{P}^*$ be an algebraic adversary acting as the prover in the protocol from construction 10 that interacts with a verifier. Then there exists an extractor $\mathcal{E}_{alg}$ that extracts a witness $\{a_i \colon i \notin I\}$ or a solution to the* BINDING *game of Pedersen commitments with probability $1 - \frac{2}{p}$ from the algebraic adversary.*

*Proof.* The adversary outputs a commitment $C$ along with a representation $\boldsymbol{y}$ such that $C = \prod_{i=1}^{\nu} h_i^{y_i}$ In the first round of the protocol, the adversary $\mathcal{P}^*$ outputs $R$ along with a representation $\boldsymbol{z}$ such that $R = \prod_{i=1}^{\nu} h_i^{z_i}$ as well as $I$, $A_I$.

We consider the following cases:

1. For all $i \in I$ it holds that $z_i = 0$: Then, the extractor simply computes all $a_i$ where $i \notin I$ $a_i = \frac{z_i - s_i}{c}$. This works if $c \neq 0$ which happens with probability $1 - \frac{1}{p}$
2. There exists at least one $i \in I$ such that $z_i \neq 0$.
   a. For all $i \in I$ it holds that $y_i = a_i$. In this case, it holds that $C = \prod_{i \in I} h_i^{a_i} \cdot \prod_{i \notin I} h_i^{y_i}$, and the extractor outputs $\{y_i \colon i \notin I\}$ as the witness.
   b. This yields an equation of the form $\prod_{i=1}^{\nu} h_i^{c \cdot y_i} \cdot \prod_{i \notin I} h_i^{s_i} = \prod_{i=1}^{\nu} h_i^{z_i}$. With probability $1 - \frac{1}{p}$ over the choice of $c$ it holds that there exists $i \in I$ such that $z_i \neq c \cdot y_i$. Thus, the extractor can output a BINDING solution, namely the commitment $R$ along with $\boldsymbol{z}$ and $\boldsymbol{z}'$ where $z'_i = c \cdot y_i$ for $i \in I$ and $z'_i = c \cdot y_i + s_i$ for $i \notin I$.

**Theorem 4 (Opening a Blinded Pedersen Commitment).** *For any algebraic adversary $\mathcal{U}$ that on input of commitment parameters $\boldsymbol{h}$ outputs blinded commitment parameters $\boldsymbol{\Psi}'$, a commitment $\zeta_1$, a partial vector $\boldsymbol{L}'$ that has non-$\perp$ entries at index $I$ a proof $\pi = (\pi_{sdl}, \pi_{op})$ auxiliary information aux such that $\pi = (\pi_{sdl}, \pi_{op})$ is the Fiat-Shamir transformation (see construction 5) of the AND-proof (see lemma 4) of the protocols described in constructions 9 and 10 along with group element representations for all group elements output by the adversary $\mathcal{U}$, there exists an extractor $\mathcal{E}$ that on input of the adversary's output including all algebraic representations that were made during the output or to the Random Oracle, outputs a witness $\gamma$ with probability $1 - Q_h \frac{1+n}{p}$ (over the choice of the random oracle) and (if the former happens) additionally either a witness $\boldsymbol{L}$ which is a partial opening to the commitment $\zeta_1$ on the indices $i \notin I$, or a solution $R, \boldsymbol{z}, \boldsymbol{z}'$ to the* BINDING *game of Pedersen commitments with probability $1 - \frac{2}{p}$ over the choice of the random oracle.*

*Proof.* The extractor proceeds as follows.

First, it calls the extractor from lemma 10 on the blinded commitment parameters transcript $\pi_{sdl}$ along with the commitment parameters $\boldsymbol{h}$ and the blinded parameters $\boldsymbol{\psi}$. This extractor outputs a witness $\gamma$ with probability $1 - Q_h \cdot \frac{1+n}{p}$ as the adversary may query the RO $Q_h$ many times, so we union bound over the choice of the challenges for the $\Sigma$-protocol transcript.

The extractor then uses $\gamma$ to transform the commitment $\zeta_1$ into $C = \zeta_1^{\frac{1}{\gamma}}$ and $R_{op}$ from $\pi_{op}$ into $R'_{op} = R_{op}^{\frac{1}{\gamma}}$. It transforms the algebraic representations by dividing by $\gamma$. It then runs the extractor from lemma 14 to extract either a witness or a binding game solution. It outputs the outputs of the two extractors.

### F.3 Double-Spending Definition

In the following, we extend the definition of blind signature with attributes (Def. 1) to also support **double-spend protection**, by modifying the user side registration algorithm, $\mathcal{RG}_{\mathcal{U}}$, and defining additional algorithms for double-spending detection, $\mathcal{DSI}$ (the *double-spend identification algorithm*) and $\mathcal{DSV}$(the *double-spend verification algorithm*), resulting in a type of scheme called $\mathsf{BSADS} = (\mathcal{G}, \mathcal{RG}, \mathsf{S}, \mathsf{U}, \mathcal{V}, \mathcal{SH}, \mathcal{DSI}, \mathcal{DSV})$. Unless stated otherwise below, the algorithms of a BSADS have the same input and output behaviour, correctness, and security notions as BSA.

- The user's registration algorithm $\mathcal{RG}_{\mathcal{U}}$ takes as input a vector of attributes $\boldsymbol{L}$ where the first attribute, denoted by $ID$, is a unique identifier of the user. It outputs a commitment a user's registration state $\mathsf{rst}_{\mathsf{U}}$ and a proof $\pi$. We assume $\mathsf{rst}_{\mathsf{U}}$ contains a commitment $C$ of $\boldsymbol{L}$ and access it as $\mathsf{rst}_{\mathsf{U}}.C$. We use the proof of knowledge of a partial opening as described in construction 10 to reveal $ID$.
- The signer's issuance algorithm $\mathsf{S}$ takes as input a secret key $\mathsf{sk}$, a registration state $\mathsf{rst}_{\mathsf{S}}$, and a commitment $C$, and outputs 1 (COMPLETE) or $\perp$ (ERROR). Additionally, it outputs a signing state $st_{\mathcal{S}}$ if it outputs 1.

– The double-spend identification algorithm $\mathcal{DSI}$ takes as input the issuer's public key $\mathsf{pk}$, a signature $\sigma$, the signer's internal state $st_{\mathcal{S}}$, and two different presentation proofs $\bar{\pi_1}, \bar{\pi_2}$. It either outputs a user $ID$ and a proof of guilt $\pi_Q$, or $\perp$.
– The double-spend verification algorithm $\mathcal{DSV}$ takes as input a user $ID$ and a proof of guilt $\pi_Q$ and it outputs 1 (ACCEPT) or 0 (REJECT).

**Discussion.** The specifics of the registration protocol will depend on the exact policy of each application scenario. For simplicity, in our definition above, we assume that the first attribute of the vector $\boldsymbol{L}$ is a unique identifier of the user, denoted as $ID$, and this uniqueness is guaranteed by the attached proof $\pi$. For example, in certain cases, during the registration process, the user might present a physical document, such as a passport, and prove (e.g., by partially opening the commitment $C$) that the encoded attributes match those in their passport. In such scenarios, $ID$ could be the unique passport number. Regardless of the specific implementation, $ID$ must be unique for each user and should be a value that cannot be duplicated or obtained by another user to prevent framing attacks where a malicious user attempts to impersonate another by registering under their identity and engaging in fraudulent activities.

A blind signature with attributes and double-spend protection $\mathsf{BSADS}$ needs satisfy two additional security properties: double-spending identification and exculpability.

Double-spending identification guarantees that no adversary is able to present the same token twice without having their public key revealed.

**Definition 25 (Double-spending Identification).** *For a blind signature with attributes and double-spending protection* $\mathsf{BSADS} = (\mathcal{G}, \mathcal{RG}, \mathsf{S}, \mathsf{U}, \mathcal{V}, \mathcal{SH}, \mathcal{DSI}, \mathcal{DSV})$*, and an adversary* $\mathcal{A}$*, we define the game* $\mathcal{DSP}$ *as follows.*

1. *Generate a pair of keys* $(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathcal{G}(\mathsf{pp})$*, and invoke* $\mathcal{A}^{\mathcal{RG}(.), \mathsf{S}(\mathsf{sk},.)}$.
2. $\mathcal{A}$ *may query its oracles* $\mathcal{RG}_{\mathsf{S}}(.), \mathsf{S}(\mathsf{sk}, .)$ *arbitrarily and in an interleaved fashion.*
3. *The game outputs 1 iff* $\mathcal{A}$ *outputs:* $(\sigma, \boldsymbol{L'_1}, \bar{\pi_1})$ *and* $(\sigma, \boldsymbol{L'_2}, \bar{\pi_2})$ *such that* $\boldsymbol{L'_1} \neq \boldsymbol{L'_2} \vee \bar{\pi_1} \neq \bar{\pi_2}$ $\mathcal{SH}_V(\sigma, \boldsymbol{L'_1}, \bar{\pi_1}) = 1$ *and* $\mathcal{SH}_V(\sigma, \boldsymbol{L'_2}, \bar{\pi_2}) = 1$ *and* $\mathcal{DSI}(\sigma, \bar{\pi_1}, \bar{\pi_2}) = \perp$.

*We define* $\mathcal{A}$*'s advantage in winning the game* $\mathcal{DSP}$ *against a blind signature scheme with attributes* $\mathsf{BSADS}$ *as*

$$\mathbf{Adv}^{\mathcal{D}}\mathcal{SP}_{\mathsf{BSADS}}(\mathcal{A}) \coloneqq \Pr[\mathcal{DSP}^{\mathcal{A}}_{\mathsf{BSADS}} = 1].$$

*We say that* $\mathsf{BSADS}$ *is* $(t, \varepsilon)$*-*$\mathcal{DSP}$*-secure if for all adversaries* $\mathcal{A}$ *that run in time at most* $t$*,* $\mathbf{Adv}^{\mathcal{D}}\mathcal{SP}_{\mathsf{BSADS}}(\mathcal{A}) \leq \varepsilon$.

Finally, we also define exculpability which guarantees that a malicious signer cannot wrongly accuse an honest user of double-spending.

**Definition 26 (Exculpability).** *For a blind signature with attributes and double-spending protection* $\mathsf{BSADS} = (\mathcal{G}, \mathcal{RG}, \mathsf{S}, \mathsf{U}, \mathcal{V}, \mathcal{SH}, \mathcal{DSI}, \mathcal{DSV})$*, and an adversary* $\mathcal{A}$*, we define the game* $\mathsf{Exc}$ *as follows.*

1. *Invoke* $\mathsf{pk} \leftarrow_{\$} \mathcal{A}$.
2. *Invoke* $\mathcal{A}^{\mathcal{RG}_{\mathcal{U}}(.),\mathsf{U}(.),\mathcal{SH}_{\mathsf{U}}(.)}$ *and allow it to query it oracles arbitrarily and in an interleaved fashion.*
3. *Initialize* $Q_U$ *to hold the set of honest user* $ID$*s generated by the oracle calls.*
4. *Answer the oracle queries as follows:*
   a. *the registration oracle* $\mathcal{RG}_{\mathcal{U}}(.)$ *takes as input an attributes vector* $\boldsymbol{L}$ *with a unique first attribute (recall that* $ID$ *is the first attribute) and runs the user's attributes registration algorithm* $\mathcal{RG}_{\mathcal{U}}$ *on it to obtain* $(\mathsf{rst}_{\mathsf{U}}, \pi)$*. It outputs* $(\mathsf{rst}_{\mathsf{U}}.C, \pi)$ *to* $\mathcal{A}$*.*
   b. *The user oracle* $\mathsf{U}(.)$ *takes as input a commitment* $C$ *and a message* $m$*. If* $\mathcal{A}$ *previously queried the registration oracle* $\mathcal{RG}_{\mathsf{U}}$ *with output state* $\mathsf{rst}_{\mathsf{U}}$ *such that* $\mathsf{rst}_{\mathsf{U}}.C = C$*, the oracle simulates the user's algorithm for*$\mathcal{A}$ *and returns the resulting signature on* $m$ *to* $\mathcal{A}$*. Otherwise, it outputs* $\perp$*. The showing state* $\mathsf{ost}_{\mathcal{U}}$ *is stored by the experiment.*
   c. *The showing oracle* $\mathcal{SH}_{\mathsf{U}}(.)$ *takes as input a signature* $\sigma$ *from the outputs of the user oracle (it outputs* $\perp$ *if* $\sigma$ *was not output by* $\mathsf{U}$*) and a set* $I_{op}$ *of indices to open. If the signature has never been queried to the showing oracle before, it runs the showing algorithm* $\mathcal{SH}_{\mathsf{U}}$ *for this signature using the internal states* $\mathsf{rst}_{\mathsf{U}}$ *and* $\mathsf{ost}_{\mathcal{U}}$ *it kept from registration and signature generation.*
5. *The game outputs 1 iff* $\mathcal{A}$ *outputs:* $(ID, \pi_Q)$ *such that* $ID \in Q_U$ *and* $\mathcal{DSV}(ID, \pi_Q) = 1$*.*

*We define* $\mathcal{A}$*'s advantage in winning the game* Exc *against a blind signature scheme with attributes* BSADS*as*

$$\mathbf{Adv}^{\mathsf{Exc}}_{\mathsf{BSADS}}(\mathcal{A}) \coloneqq \Pr[\mathsf{Exc}^{\mathcal{A}}_{\mathsf{BSADS}} = 1].$$

*We say that* BSADS*is* $(t, \varepsilon)$*-*Exc*-secure if for all adversaries* $\mathcal{A}$ *that run in time at most* $t$*,* $\mathbf{Adv}^{\mathsf{Exc}}_{\mathsf{BSADS}}(\mathcal{A}) \leq \varepsilon$*.*

# G Anonymous Credentials Light

We now present Anonymous Credentials light scheme of [7] along with protocols for partial attribute showing and double spending detection.

## G.1 The Main ACL Construction

We first recall the Anonymous Credentials Light scheme from [7]. For the reader's convenience, we have included a figure of the signing interaction in fig. 7 in section I.

Let $\mathbb{G}$ be a group with generator $g$ and prime order $p$ described by parameters $\mathsf{pp}_{\mathbb{G}}$ and hash functions $\mathsf{H}_{\mathsf{pp}}, \mathsf{H}_1 \colon \{0,1\}^* \to \mathbb{G}$, $\mathsf{H}_{\mathsf{reg}}, \mathsf{H}_{\mathsf{sh}}, \mathsf{H}_3 \colon \{0,1\}^* \to \mathbb{Z}_p$ modeled as random oracles. Let $\mathsf{rst}_{\mathsf{S}}$ and $\mathsf{rst}_{\mathsf{U}}$ be empty lists. Then the scheme $\mathsf{ACL} = (\mathcal{G}, \mathcal{RG}, \mathsf{S}, \mathsf{U}, \mathcal{V}, \mathcal{SH})$ is defined as follows:

- $\mathcal{G}(\mathsf{pp}_{\mathbb{G}})$: Sample $x \leftarrow_{\$} \mathbb{Z}_p, h \leftarrow_{\$} \mathbb{G}$, compute $y \coloneqq g^x$, generate $z \coloneqq \mathsf{H}_1(g, h, y)$,set $\mathsf{sk} \coloneqq x$ and $\mathsf{pk} \coloneqq (g, h, y, z)$, and output $(\mathsf{pk}, \mathsf{sk})$.
- $\mathcal{RG}_{\mathcal{U}}(\boldsymbol{L})$: Parse $(L_1, \ldots, L_n) \leftarrow \boldsymbol{L}$, obtain $h_0, \ldots, h_n$ by querying $0, \ldots, \nu$ to $\mathsf{H}_{\mathsf{pp}}$, sample $r_{com} \leftarrow_{\$} \mathbb{Z}_p$ and compute $C \coloneqq h_0^{r_{com}} \cdot h_1^{L_1} \cdot \ldots \cdot h_n^{L_n}$. Compute a proof of knowledge $\pi$ of an opening of $C$, using the Fiat-Shamir transform of the protocol described in construction 3 with random oracle $\mathsf{H}_{\mathsf{reg}}$, set $\mathsf{rst}_{\mathsf{U}} = (C, r_{com}, \boldsymbol{L})$, and outputs $(\mathsf{rst}_{\mathsf{U}}, \pi)$.
- $\mathcal{RG}_{\mathcal{S}}(C, \pi, \mathsf{rst}'_{\mathsf{S}})$: Verify that $\pi$ is a proof of knowledge of an opening of $C$ using the verification of the protocol described in construction 3. If the proof verifies, output an updated signer registration state $\mathsf{rst}_{\mathsf{S}} \coloneqq \mathsf{rst}'_{\mathsf{S}} \cup \{C\}$ else output $\perp$ (Error). Return 1 (Complete) or 0 (Error) to the user.
- $\mathsf{S}$: We describe the interactive signer algorithm in two parts, $\mathcal{S}_1$ and $\mathcal{S}_2$:
  - $\mathcal{S}_1(\mathsf{sk}, \mathsf{rst}_{\mathsf{S}}, C)$: Abort and output $\perp$ if $C \notin \mathsf{rst}_{\mathsf{S}}$. Otherwise, sample $d, s_1, s_2, u, \mathrm{rnd} \leftarrow_{\$} \mathbb{Z}_p$, generate $z_1 \coloneqq g^{\mathrm{rnd}} \cdot C, z_2 \coloneqq z/z_1$, compute $a \coloneqq g^u$, $b_1 \coloneqq g^{s_1} \cdot z_1^d, b_2 \coloneqq h^{s_2} \cdot z_2^d$, set internal state $st_{\mathcal{S}} \coloneqq (d, s_1, s_2, u), R \coloneqq (\mathrm{rnd}, a, b_1, b_2)$, and output $(R, st_{\mathcal{S}})$.
  - $\mathcal{S}_2(\mathsf{sk}, st_{\mathcal{S}}, R, e)$: Parse $d, s_1, s_2, u \leftarrow st_{\mathcal{S}}, x \coloneqq \mathsf{sk}$, compute $c \coloneqq e - d \mod q$ and $r \coloneqq u - c \cdot x \mod q$, and return $S \coloneqq (c, d, r, s_1, s_2)$.

  $\mathsf{S}$ outputs 1 (complete) if $\mathcal{S}_2$ outputs $S$.
- $\mathsf{U}$: The user algorithm consists of two algorithms $\mathcal{U}_1$ and $\mathcal{U}_2$:
  - $\mathcal{U}_1(\mathsf{pk}, \mathsf{rst}_{\mathsf{U}}, m, R)$: Parse $(\mathrm{rnd}, a, b_1, b_2) \leftarrow R$, sample $\gamma \leftarrow_{\$} \mathbb{Z}_p^*, \tau, t_1, t_2, t_3, t_4, t_5 \leftarrow_{\$} \mathbb{Z}_p$, generate $z_1 \coloneqq g^{\mathrm{rnd}} \cdot \mathsf{rst}_{\mathsf{U}}.C, \zeta \coloneqq z^{\gamma}, \zeta_1 \coloneqq z_1^{\gamma}, \zeta_2 \coloneqq \zeta/\zeta_1$, compute $\alpha \coloneqq a \cdot g^{t_1} \cdot y^{t_2}, \beta_1 \coloneqq b_1^{\gamma} \cdot g^{t_3} \cdot \zeta_1^{t_4}, \beta_2 \coloneqq b_2^{\gamma} \cdot h^{t_5} \cdot \zeta_2^{t_4}, \eta \coloneqq z^{\tau}$, generate $\varepsilon \coloneqq \mathsf{H}_3(\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta, m)$, compute $e \coloneqq \varepsilon - t_2 - t_4 \mod q$, set internal state $st_{\mathcal{U}} \coloneqq (\gamma, \tau, t_1, t_2, t_3, t_4, t_5, \mathrm{rnd})$, and return $(e, st_{\mathcal{U}})$.
  - $\mathcal{U}_2(\mathsf{pk}, st_{\mathcal{U}}, R, e, S, m)$: Parse $(\gamma, \tau, t_1, t_2, t_3, t_4, t_5, \mathrm{rnd} \leftarrow st_{\mathcal{U}}$, and $(c, d, r, s_1, s_2) \leftarrow S$ , compute $\rho \coloneqq r + t_1, \omega \coloneqq c + t_2, \sigma_1 \coloneqq \gamma \cdot s_1 + t_3, \sigma_2 \coloneqq \gamma \cdot s_2 + t_5, \delta \coloneqq d + t_4, \mu \coloneqq \tau - \delta \cdot \gamma$, and return the signature $\hat{\sigma} \coloneqq (\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu)$ and $\mathsf{ost}_{\mathcal{U}} = (\gamma, \mathrm{rnd})$ if $\omega + \delta \equiv \mathsf{H}_3(\zeta, \zeta_1, g^{\rho} \cdot y^{\omega}, g^{\sigma_1} \cdot \zeta_1^{\delta}, h^{\sigma_2} \cdot \zeta_2^{\delta}, z^{\mu} \cdot \zeta^{\delta}, m)$, $\perp$ otherwise.
- $\mathcal{SH}_U, \mathcal{SH}_V$: see fig. 5.
- $\mathcal{V}(\mathsf{pk}, m, \hat{\sigma})$: Parse $\hat{\sigma}$ as $(\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu)$, and output 1 if $\zeta \not\equiv 1$ and $\omega + \delta \equiv \mathsf{H}_3(\zeta, \zeta_1, g^{\rho} \cdot y^{\omega} g^{\sigma_1} \cdot \zeta_1^{\delta}, h^{\sigma_2} \cdot (\zeta/\zeta_1)^{\delta}, z^{\mu} \cdot \zeta^{\delta})$, 0 otherwise.

### G.2 ACL Showing of Partial Attributes

Showing and verification of showing is depicted in fig. 5. We note that it is also possible to "show" a signature without revealing any attributes, i.e. when $I_{op}$ is empty. In this case, the user can run a proof of same discrete logarithm (see construction 9) for the blinded commitment parameters, as well as a proof of knowledge of an opening of a Pedersen commitment to the blinded parameters like the one used at registration.

The procedure described in fig. 5 allows for extraction in the AGM:

**Theorem 5.** *For any algebraic adversary $\mathcal{U}$ that, on input of commitment parameters $\mathsf{pp}_{\mathsf{GPC}} = \boldsymbol{h}$, it outputs an* ACL *public key* $\mathsf{pk}$ *along with a tuple* $(m, \sigma, \boldsymbol{L}'$,

$I_{op}, \bar{\pi})$ *such that* $\mathcal{SH}_V(\mathsf{pk}, m, \sigma, \boldsymbol{L}', I_{op}, \bar{\pi}) = 1$*, there exists an extractor* $\mathcal{E}$ *that takes* $\mathcal{U}$*'s output* $(\mathsf{pk}, (m, \sigma, \boldsymbol{L}', I_{op}, \bar{\pi}))$ *along with all algebraic representations* $\mathcal{U}$ *outputs as input, and it outputs a witness* $\gamma$ *of the proof of same discrete logarithm with probability* $1 - Q_h \frac{1+n}{p}$ *(over the choice of the random oracle) and (if the former happens) additionally either a witness* $\boldsymbol{L}$ *which is a partial opening to the commitment* $\zeta_1$ *on the indices* $i \notin I$*, or a solution* $R, \boldsymbol{z}, \boldsymbol{z}'$ *to the* BINDING *game of Pedersen commitments with probability* $1 - \frac{2}{p}$ *over the choice of the random oracle.*

*Proof.* This follows directly from theorem 4.

### G.3 Secure Showing of ACL

**Theorem 6.** *If the discrete logarithm problem is* $(t, \varepsilon)$*-hard in the group* $\mathbb{G}$*, then* ACL*is* $(t', \varepsilon')$*-SS secure according to definition 5 with arbitrary messages (item 4.c.ii) and selective reveal (item 4.d.ii) in the AGM + ROM with* $t' \approx t$ *and*

$$\varepsilon' \leq (\nu + 21) \cdot \varepsilon + \frac{(\nu + 3) \cdot \ell \cdot Q_H + 52 + 4 \cdot Q_C}{p},$$

*where* $\ell$ *is the number of signing sessions closed,* $Q_H$ *is the number of hash queries the adversary made to* $\mathsf{H}_{\mathsf{sh}}$*,* $Q_C$ *is the number of commitments* $\mathcal{A}$ *registers, and* $\nu$ *is the maximal number of attributes contained in each attribute vector.*

*Proof.* We work in the AGM and show that an adversary that breaks the secure showing property can be used to break the discrete logarithm assumption in the underlying group. Our proof technique varies depending on the adversary's strategy for breaking the secure showing property - namely it can either provide a set of signatures where for some signatures, the blinded $\zeta_1$ components do not match the $C \cdot g^{\text{rnd}}$ components of the signing sessions. This can happen either in the form of one "odd one out" signature that matches no session (we rule this out in $\mathbf{G}_2$), or in two signatures that match the same signing session (we rule this out in $\mathbf{G}_3$). The last possible strategy is for the adversary to break the binding property of the underlying Pedersen commitment scheme which we rule out in lemma 17.

*Game* $\mathbf{G}_1$*.* This is the secure showing game for ACL.

*Game* $\mathbf{G}_2$*.* This game aborts if the following happens: The adversary outputs at least one signature $\sigma^*$ such that the commitment $\zeta^*, \zeta_1^*$ does not "match" any of the session tags on the signer's side. This means, there exists no $z_{1,i}$ such that $\text{dlog}_z \zeta^* = \text{dlog}_{z_{1,i}} \zeta_1^*$.

We will show that this contradicts the so-called *restrictive blinding lemma*, see lemma 15. We prove this lemma using mostly techniques from [41] to show that an adversary can be used to break DLP if it wins this game. A formal definition of the game $\mathsf{RB}_{\mathsf{ACL}}$ can be found in definition 27 in section H.

$\mathcal{SH}_U(\mathsf{pk}, \mathsf{rst}_U, m, \sigma, \mathsf{ost}_U, I_{op})$

$(C, r_{com}, \boldsymbol{L}) \coloneqq \mathsf{rst}_U$
$(\mathrm{rnd}, \gamma) \coloneqq \mathsf{ost}_U$
$\Gamma \coloneqq g^{\gamma}$
$\boldsymbol{\Psi} \coloneqq \{\psi_k \coloneqq h_k^{\gamma}\}_{k \in [\![\nu]\!]}$
$r_{sdl} \leftarrow_{\$} \mathbb{Z}_p$
$\boldsymbol{h}_{sdl} \coloneqq \{h_{sdl,k} \coloneqq h_k^{r_{sdl}}\}_{k \in [\![\nu]\!]}$
$z_{sdl} \coloneqq z^{r_{sdl}}$
$g_{sdl} \coloneqq g^{r_{sdl}}$
$r_g, \{r_i\}_{i \in [\nu]_0 \setminus I_{op}} \leftarrow_{\$} \mathbb{Z}_p$
$R \coloneqq g^{r_g} \cdot \prod_{i \in [\nu]_0 \setminus I_{op}} \psi_i^{r_i}$
**for** $i \in [\nu] \setminus I_{op}$ :
  $L'_i \coloneqq \bot$
**for** $i \in I_{op}$ :
  $L'_i \coloneqq L_i$
$c \leftarrow_{\$} \mathsf{H}_{\mathsf{sh}}(\mathsf{pk}, \sigma, \sigma.\zeta_1, m, \boldsymbol{\Psi}, \Gamma, g_{sdl}, z_{sdl}, \boldsymbol{h}_{sdl}, R, \boldsymbol{L'})$
$s_{sdl} \coloneqq r_{sdl} - c \cdot \gamma$
$s_0 \coloneqq r_0 - c \cdot r_{com}$
$\{s_i \coloneqq r_i - L_i \cdot c\}_{i \in [\nu]_0 \setminus I_{op}}$
$s_\Gamma \coloneqq r_g - \mathrm{rnd} \cdot c$
$S \coloneqq \{s_i\}_{k \in I_{op}} \cup \{s_\Gamma\}$
$\pi_{sdl} \coloneqq (\Psi, \Gamma, g_{sdl}, z_{sdl}, s_{sdl}, h_{sdl})$
$\pi_{op} \coloneqq (R, S)$
**return** $(\boldsymbol{L'}, (\pi_{sdl}, \pi_{op}))$

$\mathcal{SH}_V(\mathsf{pk}, \sigma, m, \boldsymbol{L'}, I_{op}, \bar{\pi})$

$(\pi_{sdl}, \pi_{op}) \coloneqq \bar{\pi}$
**if** $\mathcal{V}(\mathsf{pk}, m, (\sigma, \zeta_1)) = 0$
  $\vee\, (\exists i \in I_{op} \colon L'_i = \bot) \vee (\exists i \notin I_{op} \colon L'_i \neq \bot)$ :
  **return** 0
$(\Psi, \Gamma, g_{sdl}, z_{sdl}, s_{sdl}, h_{sdl}) \coloneqq \pi_{sdl}$
$(\psi_1, \ldots, \psi_\nu) \coloneqq \Psi$
$(h_{sdl,1}, \ldots, h_{sdl,\nu}) \coloneqq h_{sdl}$
$c \leftarrow_{\$} \mathsf{H}_{\mathsf{sh}}(\mathsf{pk}, \sigma, \zeta_1, m, \Psi, \Gamma, g_{sdl}, z_{sdl}, \boldsymbol{h}_{sdl}, R, \boldsymbol{L'})$
**if** $\Gamma^c \cdot g^{s_{sdl}} \neq g_{sdl} \vee \zeta^c \cdot z^{s_{sdl}} \neq z_{sdl}$
  $\vee \bigvee_{i \in [\![\nu]\!]} \psi_i^c \cdot h_i^{s_{sdl}} \neq h_{sdl,i}$ :
  **return** 0
$(R, S) \coloneqq \pi_{op}$
$\zeta'_1 \coloneqq \zeta_1 / \prod_{i \in I_{op}} \psi_i^{L'_i}$
**if** $\zeta_1'^c \cdot \Gamma^{s_\Gamma} \cdot \prod_{i \in [\nu]_0 \setminus I_{op}} \psi_i^{s_i} \neq R$ :
  **return** 0
**return** 1

Fig. 5: The Showing procedure. The $\zeta_1$ component of the signature $\sigma$, denoted here by $\sigma.\zeta_1$ is a blinded commitment $C$, for each a signature $\sigma$ was issued via an interaction with an honest signer. $(\gamma, \mathrm{rnd})$ is the blinding factor such that $\zeta_1 = (g^{\mathrm{rnd}} \cdot C)^{\gamma}$. The proof consists of a part that proves the correct blinding of the commitment parameters via a proof of same discrete logarithm and of a part that proves knowledge of an opening for the remaining part of the commitment that is not revealed by the set of opened indices $I_{op}$.

**Lemma 15 (Restrictive Blinding).** *In the ROM and AGM, if* DLP *is* $(t, \varepsilon)$*-hard in* $\mathbb{G}$*, it holds that for any adversary running in time at most* $t$*, the advantage is at most*

$$\mathbf{Adv}^{\mathsf{RB}}_{\mathsf{ACL}}(\mathcal{A}) \leq 13 \cdot \varepsilon + \frac{4 \cdot Q_C + 31}{p}.$$

*where* $Q_C$ *is the number of registration queries made by the adversary.*

The proof can be found in section H.

**Corollary 1.** $\Pr[\mathbf{G}_1 = 1] \leq \Pr[\mathbf{G}_2 = 1] + 13 \cdot \varepsilon + \frac{4 \cdot Q_C + 31}{p}$.

*Proof.* We provide a reduction: The reduction simulates the secure showing game to the adversary by forwarding its queries to all oracles and their responses. The reduction then checks whether the adversary has won the secure showing game. If yes, it outputs the messages and signatures from the adversary's output to the $\mathsf{RB}$ game. It thus wins whenever the abort condition of $\mathbf{G}_2$ occurs.

*Game* $\mathbf{G}_3$. In this game, we abort if the adversary does the following: The adversary outputs at least two signatures $\sigma', \sigma''$ such that the commitments $\zeta', \zeta_1'$ and $\zeta'', \zeta_1''$ map to the same signing session, i.e. there exists a single $z_{1,i}$ such that both $\mathrm{dlog}_z \zeta' = \mathrm{dlog}_{z_{1,i}} \zeta_1'$ and $\mathrm{dlog}_z \zeta'' = \mathrm{dlog}_{z_{1,i}} \zeta_1''$. We will show that this contradicts what we call the *2-to-1-lemma*, see lemma 16 - this lemma can be derived from techniques used in the $\mathsf{OMUF}$-proof of [41]. A formal definition of the game $\ell$-$\mathsf{TTO}_{\mathsf{ACL}}$ can be found in definition 29 in section H.

**Lemma 16 (2-to-1 blinding).** *In the ROM and AGM, if* $\mathsf{DLP}$ *is* $(t, \varepsilon)$*-hard in* $\mathbb{G}$*, it holds that for any adversary running in time at most* $t$*, the advantage is at most*

$$\mathbf{Adv}^{\ell\text{-}\mathsf{TTO}}_{\mathsf{ACL}}(\mathcal{A}) \leq 7 \cdot \varepsilon + \frac{20}{p}.$$

The proof can be found in section H

We now apply this to the current game hop.

**Corollary 2.**

$$\Pr[\mathbf{G}_2 = 1] \leq \Pr[\mathbf{G}_3 = 1] + (7\varepsilon + \frac{20}{p}).$$

*Proof.* Again, the reduction forwards all queries from the adversary to the $\ell$-$\mathsf{TTO}$ and also forwards the responses. When the adversary outputs a solution, it checks that the adversary has won the secure showing game, and then outputs the messages and signatures from the adversary's output to the $\ell$-$\mathsf{TTO}$ game. It thus wins whenever the abort condition of $\mathbf{G}_3$ occurs.

**Lemma 17 (Partial Opening).** *In the ROM and AGM, if* $\mathsf{DLP}$ *is* $(t, \varepsilon)$*-hard in* $\mathbb{G}$*, it holds that for any adversary* $\mathcal{A}$ *that runs in time at most* $t$*, the probability of winning* $\mathbf{G}_3$ *is at most:*

$$\Pr[\mathbf{G}_3^{\mathcal{A}}] \leq (\nu + 1) \cdot \varepsilon + \frac{(\nu + 3) \cdot \ell \cdot Q_{\mathsf{H}_{\mathsf{sh}}} + 1}{p}$$

*where* $\ell$ *is the number of signing sessions closed by the adversary* $\mathcal{A}$ *and* $Q_{\mathsf{H}_{\mathsf{sh}}}$ *is the number of hash queries the adversary makes to the random oracle* $\mathsf{H}_{\mathsf{sh}}$

*Proof.* We construct an adversary $\mathcal{R}_5$ against the binding property of the Pedersen commitment $C$. The adversary simulates the signing procedure to the adversary using the $y$-side key like an honest signer would, and extracts openings for the commitments made by the adversary. We describe this below in more details.

First of all, the reduction constructs a wrapper $\mathcal{B}$ around the adversary which takes as input the commitment parameters and has access to the random oracle $\mathsf{H}_{\mathsf{sh}}$.

**Setup.** The wrapper $\mathcal{B}$ takes as input the commitment parameters $g, \boldsymbol{h}$ for the Pedersen commitment. It samples a key for ACL using the $\mathcal{G}(\mathsf{pp}_{\mathbb{G}})$ algorithm while programming the RO $\mathsf{H}_1$ so that it knows the discrete logarithms of its outputs (i.e. including $z$). It outputs the key to the adversary $\mathcal{A}$ and programs the random oracle $\mathsf{H}_{\mathsf{pp}}$ to output $h_i$ when queried on $i$ for $i \in [\nu]$.

**Online Phase** The wrapper simulates the secure showing game to the adversary by lazy-sampling all random oracles (in case of group elements with known discrete logarithm to base $g$) and by using the secret key to answer signing queries like an honest signer would. Whenever the adversary makes a query to a random oracle that contains a group element, the wrapper $\mathcal{B}$ takes a note of the algebraic representations submitted for that group element. It computes a reduced representation where it replaces all occurrences of $y, z, h, a_i, b_{1,i}, b_{2,i}$ with their respective discrete logarithms to base $g, \boldsymbol{h}$, concretely, replace $y$ by $g^x$, $z$ by $g^{\mathrm{dlog} z}$ (known from programming $\mathsf{H}_1$), $h$ by $\mathrm{dlog} h$, $a_i$ by $g^{u_i}$, $b_{1,i}$ by $g^{s_{1,i}+d_i \cdot \mathrm{rnd}_i} \cdot C_i^{d_i}$, $b_{2,i}$ by $g^{s_{2,i} \cdot \mathrm{dlog} h} \cdot (g^{\mathrm{dlog} z} / (g^{\mathrm{rnd}_i} \cdot C_i))^{d_i}$ where $C_i$ is recursively replaced by replacing the values in its representation. Note that by the time $b_{1,i}, b_{2,i}$ is output, the representation of $C_i$ is already fixed and thus there exist no circular dependencies.

**Output determination** The wrapper outputs whatever the adversary outputs, that is the public key $\mathsf{pk}$, a list of tuples $\{(m_i, \sigma_i, \zeta_{1,i}, L_i, \pi_i)\}_{i \in k}$ and tuples $\{(L'_i, r_i)\}_{i \in [\ell]}$, alongside the algebraic representations that the adversary output the first time it output a specific group element (i.e. if the group element was queried to a random oracle before, the wrapper $\mathcal{B}$ outputs the representation that was submitted along with the RO query).

The reduction now employs the extractor $\mathcal{E}$ from theorem 5 of $\Pi$ for each of the proofs $\pi_i = (\pi_{i,sdl}, \pi_{i,op})$ along with the commitments and algebraic representations that $\mathcal{B}$ has output. Note that $\mathcal{B}$ is a valid adversary that behaves like required in theorem 5. With probability at least $(1 - \frac{\ell \cdot Q_{\mathsf{H}_{\mathsf{sh}}} \cdot (\nu+1)}{p})(1 - \frac{\ell \cdot Q_{\mathsf{H}_{\mathsf{sh}}} \cdot 2}{p})$ $\geq 1 - \frac{(\nu+3) \cdot \ell \cdot Q_{\mathsf{H}_{\mathsf{sh}}} + 1}{p}$, the extractor outputs witnesses of the form $\gamma_i$, and either a commitment $R$ with two openings $\boldsymbol{Z}, \boldsymbol{Z}'$, or partial openings of the commitments $\zeta_{1,i}$ of the form $\mathrm{rnd}_i, r_{i,0}, r_{i,j} \forall j$ s.t. $L_{i,j} = \perp$.

If there is one proof that yielded a commitment $R$ and two openings $\boldsymbol{Z}, \boldsymbol{Z}'$, the reduction outputs these as its solution to the binding game. Otherwise, the reduction proceeds as follows.

As part of the output of the adversary, the reduction obtains the (full) opening vectors $\boldsymbol{L}'_i$ for each commitment $C_i$ that was used during registration.

The reduction now picks a session-signature-pair $(\mathrm{rnd}, C)$, $(m, \hat{\sigma}, \gamma)$ with $\mathrm{dlog}_{C \cdot g^{\mathrm{rnd}}} \zeta_1 = \mathrm{dlog}_z \zeta$ and $\boldsymbol{L} \not\subseteq \boldsymbol{L}'$ where $\boldsymbol{L}$ is the vector that $\mathcal{A}$ partially opened in the showing algorithm, and $\boldsymbol{L}'$ is the vector of the signing session that is matched through $\gamma$.

It unites the partial vector $\boldsymbol{L}$ with the corresponding partial opening $\mathrm{rnd}_i$, $r_{i,0}, r_{i,j}\, \forall j$ s.t. $L_{i,j} = \perp$ obtained from the extractor into an opening $\boldsymbol{L}''_i$.

The reduction now outputs the commitment $C$ along with the two openings $\boldsymbol{L}'_i$ and $\boldsymbol{L}''_i$ to the challenger.

Thus, the reduction wins the binding game for the Pedersen commitment scheme with probability of at least

$$\Pr[\mathbf{G}_3^{\mathcal{A}} = 1] \cdot (1 - \frac{(\nu+3) \cdot \ell \cdot Q_{\mathsf{H}_{\mathsf{sh}}} + 1}{p}) \geq \Pr[\mathbf{G}_3^{\mathcal{A}} = 1] - \frac{(\nu+3) \cdot \ell \cdot Q_{\mathsf{H}_{\mathsf{sh}}} + 1}{p}.$$

Putting this together with lemma 3 yields the statement.

Putting everything together we get that

$$\varepsilon' \leq (\nu + 21) \cdot \varepsilon + \frac{(\nu+3) \cdot \ell \cdot Q_H + 52 + 4 \cdot Q_C}{p}.$$

### G.4 ACL Blindness

We show that ACL fulfills our stronger definition of blindness that takes into account the showing algorithm.

**Theorem 7.** *If the* DDH *problem is* $(t, \varepsilon)$*-hard in* $\mathbb{G}$*, then* ACL *is* $(t', \varepsilon')$*-blind in the ROM with* $t' \approx t$ *and* $\varepsilon' \leq 2\varepsilon + \frac{2Q_{\mathsf{H}_{\mathsf{reg}}} + 2Q_{\mathsf{H}_{\mathsf{sh}}} + 2Q_{\mathsf{H}_3} + 3}{p}$ *where* $Q_{\mathsf{H}_{\mathsf{reg}}}, Q_{\mathsf{H}_{\mathsf{sh}}}$*, and* $Q_{\mathsf{H}_3}$ *are the adversary's numbers of queries to* $\mathsf{H}_{\mathsf{reg}}, \mathsf{H}_{\mathsf{sh}}$*, and* $\mathsf{H}_3$*, respectively.*

*Proof.* The proof mostly follows the same strategy as [7], but we make the registration and showing procedures more explicit.

We proceed through a series of games where we switch to simulating the proofs provided at registration and during showing, such that finally we can provide two reductions that embed DDH challenges in the signatures.

*Game* $\mathbf{G}_1$. This is the original blindness game.

*Game* $\mathbf{G}_2$. This game aborts whenever the challenger makes a query to $\mathsf{H}_{\mathsf{reg}}, \mathsf{H}_{\mathsf{sh}}$, $\mathsf{H}_3$ that has been made before. Each of the inputs to these oracles contain at least one component that is randomized over the entire group, in the case of $\mathsf{H}_{\mathsf{reg}}$ and $\mathsf{H}_{\mathsf{sh}}$ this is the first flow of the sigma protocol proof, and in the case of $\mathsf{H}_3$ this is (among others) the group element $\alpha$. Thus, the probability for colliding with a previously made query is $\frac{2Q_{\mathsf{H}_{\mathsf{reg}}} + 1 + 2Q_{\mathsf{H}_{\mathsf{sh}}} + 1 + 2Q_{\mathsf{H}_3} + 1}{p}$ and the difference to the previous game is

$$\mathbf{Adv}^{\mathsf{ACL}}_{\mathbf{G}_2}(\mathcal{A}) - \mathbf{Adv}^{\mathsf{ACL}}_{\mathbf{G}_1}(\mathcal{A}) \leq \frac{2Q_{\mathsf{H}_{\mathsf{reg}}} + 2Q_{\mathsf{H}_{\mathsf{sh}}} + 2Q_{\mathsf{H}_3} + 3}{p}.$$

*Game* $\mathbf{G}_3$. This game switches to simulating the proofs generated during showing. That is, for each session it picks values $c$, $s_{sdl}, s_\Gamma, \{s_i\}_{i\in[n]_0\setminus I}$ (where $I$ is $I_0$ or $I_1$ respectively) and sets $h_{sdl,k} = h_k^{s_{sdl}} \cdot \psi_k^c$, and $R \coloneqq \zeta_1'^c \cdot \Gamma^{s_\Gamma} \cdot \prod_{i\in[n]_0\setminus I} \psi_i^{s_i}$. All other elements it computes the same as the honest protocol. It sets $\mathsf{H_{sh}}(\mathsf{pk}, \sigma, \sigma.\zeta_1, m, \boldsymbol{\Psi}, \gamma, g_{sdl}, z_{sdl}, \boldsymbol{h}_{sdl}, R, \boldsymbol{L}')$ to output $c$. This last step works as the input to $\mathsf{H_{sh}}$ is identically distributed to $\mathbf{G}_2$ and $\mathbf{G}_2$ aborts in the case that $\mathsf{H_{sh}}$ has been queried on this value before. Thus, it holds that

$$\mathbf{Adv}^{\mathsf{ACL}}_{\mathbf{G}_3}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{ACL}}_{\mathbf{G}_2}(\mathcal{A}).$$

*Game* $\mathbf{G}_4$. In this game, we switch to simulating the proof $\pi$ generated at registration. The game samples $c, s_0, \ldots, s_\nu$ and sets $R \coloneqq C^c \cdot \prod_{i=0}^{\nu} h_i^{s_i}$ in each of the two sessions. It sets $c =: \mathsf{H_{reg}}(\boldsymbol{h}, C, R)$ and outputs $\pi$. This game is identically distributed to $\mathbf{G}_3$ by the same argument as the previous game hop and thus

$$\mathbf{Adv}^{\mathsf{ACL}}_{\mathbf{G}_4}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{ACL}}_{\mathbf{G}_3}(\mathcal{A}).$$

*Game* $\mathbf{G}_5$. In this game, we switch to simulating the signatures. For each signature, the game samples values $\gamma, \omega, \rho, \delta, \sigma_1, \sigma_2, \mu \leftarrow \mathbb{Z}_p$ and sets $\zeta = z^\gamma, \zeta_1 = (C \cdot g^{\mathrm{rnd}})^\gamma$ and $\mathsf{H}_3)(\zeta, \zeta_1, g^\rho \cdot y^\omega, g^{\sigma_1}\zeta_1^\delta, h^{\sigma_2}(\zeta/\zeta_1)^\delta, z^\mu\zeta^\delta) \coloneqq \omega + \delta$. As a challenge during signing, it sends a random value $e \leftarrow \mathbb{Z}_p$ and it checks after the signer's response $(c, d, r, s_1, s_2)$ that $a = g^r y^c, b_1 = g^{s_1}(C \cdot g^{\mathrm{rnd}})^d$, and $b_2 = h^{s_2}(z/(C \cdot g^{\mathrm{rnd}}))^d$. If any of these checks fail it sets the signature to $\perp$. This game is identically distributed to $\mathbf{G}_4$ by the same argument as the previous game hops and thus

$$\mathbf{Adv}^{\mathsf{ACL}}_{\mathbf{G}_5}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{ACL}}_{\mathbf{G}_4}(\mathcal{A}).$$

*Game* $\mathbf{G}_6$. In this game, we switch the $\zeta, \zeta_1$ pair of the signature on message $m_b$ to random group elements. This game hop is bounded by the hardness of $\mathsf{DDH}$ in $\mathbb{G}$. We provide a reduction $\mathcal{R}_1$:

- **Setup.** The reduction takes as input a DDH challenge $g, X, Y, Z$. It programs the random oracle $\mathsf{H}_1$ to output $X \cdot g^u$ for randomly chosen $u$ at each fresh query. It furthermore programs the random oracle $\mathsf{H_{pp}}(i)$ to output $g^{v_i}$ for $i = 0, \ldots, \nu$ with randomly chosen $v_i$. It then calls the adversary to obtain $\mathsf{pk}, m_0, m_1, \boldsymbol{L}_0, \boldsymbol{L}_1$. The reduction samples a bit $b \leftarrow \{0, 1\}$ and calls the adversary again to run. Denote by $u$ the value $u$ such that $z = X^u$ in the public key.
- **Online Phase** The reduction simulates the oracles as follows.
  - $\mathcal{RG}_{\mathcal{U}}(\boldsymbol{L}_0)$ sets the commitment to $X \cdot g^w$ for randomly chosen $w \leftarrow \mathbb{Z}_p$. It proceeds as $\mathbf{G}_5$.
  - $\mathcal{RG}_{\mathcal{U}}(\boldsymbol{L}_1)$ sets the commitment honestly to $\boldsymbol{h}^{\boldsymbol{L}_1}$

- $\mathsf{U}^0$ sets $\zeta = Z \cdot Y^u$ and $\zeta_1 = Z \cdot Y^{w+\mathrm{rnd}_0}$. It proceeds with signing as in $\mathbf{G}_5$.
- $\mathsf{U}^1$ proceeds with signing as in $\mathbf{G}_5$.

– **Showing Phase.** If the two signing sessions result in valid signatures, the game produces the proof for $(m_b, \sigma_b, \zeta_{1,b})$ as follows: It sets $\Gamma = Y$, $\psi_k = Y^{v_k}$ for $k \in [\![n]\!]$ and then proceeds as in $\mathbf{G}_5$. For the other signature it proceeds as in $\mathbf{G}_5$.
– **Output Determination.** The reduction outputs 0 if $b = b'$ and 1 otherwise.

It is easy to see that the reduction perfectly simulates $\mathbf{G}_5$ if the bit of the DDH game is 0 (Real) and $\mathbf{G}_5$ if the bit of the DDH game is 1 (Random). Thus it holds that

$$\mathbf{Adv}^{\mathsf{ACL}}_{\mathbf{G}_6}(\mathcal{A}) - \mathbf{Adv}^{\mathsf{ACL}}_{\mathbf{G}_5}(\mathcal{A}) \leq \mathbf{Adv}^{\mathbb{G}}_{\mathsf{DDH}}(\mathcal{R}_1)$$

*Game* $\mathbf{G}_7$. In this game, we switch the $\zeta, \zeta_1$ pair of the second signature to a random value.

– **Setup.** The reduction takes as input a DDH challenge $g, X, Y, Z$. It programs the random oracle $\mathsf{H}_1$ to output $X \cdot g^u$ for randomly chosen $u$ at each fresh query. It furthermore programs the random oracle $\mathsf{H}_{\mathsf{pp}}(i)$ to output $g^{v_i}$ for $i = 0, \ldots, \nu$ with randomly chosen $v_i$. It then calls the adversary to obtain $\mathsf{pk}, m_0, m_1, \boldsymbol{L}_0, \boldsymbol{L}_1$. The reduction samples a bit $b \leftarrow \{0, 1\}$ and calls the adversary again to run. Denote by $u$ the value $u$ such that $z = X^u$ in the public key.
– **Online Phase** The reduction simulates the oracles as follows.
- $\mathcal{RG}_{\mathcal{U}}(\boldsymbol{L}_0)$ sets the commitment to a randomly chosen value $C \leftarrow \mathbb{G}$ and proceeds as in $\mathbf{G}_6$.
- $\mathcal{RG}_{\mathcal{U}}(\boldsymbol{L}_1)$ sets the commitment to $X \cdot g^w$ for randomly chosen $w \leftarrow \mathbb{Z}_p$. It proceeds as $\mathbf{G}_6$.
- $\mathsf{U}^0$ proceeds with signing as in $\mathbf{G}_6$, i.e. it sets $\zeta$ and $\zeta_1$ to be uniformly random group elements and simulates the signature.
- $\mathsf{U}^1$ sets $\zeta = Z \cdot Y^u$ and $\zeta_1 = Z \cdot Y^{w+\mathrm{rnd}_0}$. It proceeds with signing as in $\mathbf{G}_6$.

– **Showing Phase.** If the two signing sessions result in valid signatures, the game produces the proof for $(m_{1-b}, \sigma_{1-b}, \zeta_{1,1-b})$ as follows: It sets $\Gamma = Y$, $\psi_k = Y^{v_k}$ for $k \in [\![n]\!]$ and then proceeds as in $\mathbf{G}_6$. For the other signature it proceeds as in $\mathbf{G}_6$, i.e. it sets $\psi_i = h_i^{\gamma}$ for a randomly chosen $\gamma$.
– **Output Determination.** The reduction outputs 0 if $b = b'$ and 1 otherwise.

It is easy to see that the reduction perfectly simulates $\mathbf{G}_6$ if the bit of the DDH game is 0 (Real) and $\mathbf{G}_6$ if the bit of the DDH game is 1 (Random). Thus it holds that

$$\mathbf{Adv}^{\mathsf{ACL}}_{\mathbf{G}_7}(\mathcal{A}) - \mathbf{Adv}^{\mathsf{ACL}}_{\mathbf{G}_6}(\mathcal{A}) \leq \mathbf{Adv}^{\mathbb{G}}_{\mathsf{DDH}}(\mathcal{R}_1)$$

In $\mathbf{G}_7$, it holds that $\mathbf{Adv}^{\mathsf{ACL}}_{\mathbf{G}_7}(\mathcal{A}) = 0$ as the signatures are completely unrelated to the signing sessions. Summing up the probabilities yields the claim

### G.5 ACL Double-Spending Detection

We now explain how the ACL protocol can also support Double-Spending detection. We require the following modifications:

- In the main protocol, as described in Fig. 7, we add the following *additional* steps to the User side of the protocol: (1) Pick $\tau_2 \leftarrow_\$ \mathbb{Z}_p$, (2) compute $\eta_2 := z^{\tau_2}$, (3) include $\eta_2$ in the hash computation of $\varepsilon$. The signature verification algorithm can either be modified to include the element $\eta_2$ in the hash computation, or one can treat $\eta_2$ as if it was part of the message when verifying signatures which enables using the verification algorithm of the plain scheme.
- In the showing protocol, as described in Fig. 5, we add the following steps: (1) The user includes $V_{\text{name}}$ and timestamp as part of the hash computation in $c$. (2) The user computes $\mu_2 = \tau_2 - c\gamma$ and outputs $\mu_2$ along with $(\pi_{sdl}, \pi_{op})$. (3) The verifier runs an updated signature verification algorithm which also includes $z^{\mu_2} \cdot \zeta^c$ as part of $\mathsf{H}_3$ (which verifies with $\eta_2$ part of $\varepsilon$ in the signature).
- $\mathcal{DSI}$: If a user double spends the same signature $\sigma$ using showings $\bar{\pi_1}, \bar{\pi_2}$, it results into two different values $c, c'$ and $\mu_2, \mu_2'$ from which one can extract $\gamma = \frac{\mu_2' - \mu_2}{c - c'}$ and thus $\zeta_1$ which uniquely identifies the issuing session. This is then used by the signer to identify the signing session. It outputs the proof of double spending as the user's commitment $C$, the user's proof of partial opening to $ID$, the signer's randomization factor rnd, and the user's blinding factor $\gamma$, along with the signature $\sigma$ and the two showings $\bar{\pi_1}, \bar{\pi_2}$.
- $\mathcal{DSV}$: On input of an ID and a double spending proof, the $\mathcal{DSV}$ algorithm outputs 1 iff both showings of the signature are valid, the proof of partial opening of $C$ to $ID$ is valid, and it holds that $(C \cdot g^{\text{rnd}})^\gamma = \zeta_1$.

*Discussion.* We note that double-spending detection for ACL was already discussed in [38] (a work that implements [7] as an e-cash scheme), but it was done in a slightly different way and without a formal security proof. Namely, in [38] the value $\mu$ is omitted from the signature[10] instead of adding the additional $\tau_2, \eta_2$ as we did above. Then, the showing protocol uses $\tau$ to compute $\mu_2$. It is critical that $\mu$ is removed from the signature, since otherwise a single credential showing already links it to a signing session because the signer can use $\frac{\mu_2 - \mu}{\delta - c}$ to compute $\gamma$ and then check which $z_1^\gamma = \zeta_1$.

While this double-spending protocol for ACL is correct, it does not compose well with the existing security proofs for ACL. Namely, removing $\mu$ from the ACL signature removes the guarantees provided by existing proofs of one-more unforgeability [7,41] as the signatures can no longer be verified on their own. Thus, we went for a modular approach, that does not affect the ACL proof, by computing additional elements $(\tau_2, \eta_2)$ of the same form as $(\tau, \eta)$ and adding $\eta_2$ to the hash during the signature generation in order to use this for the double spending detection. As noted above, this modification does not affect the ACL verification algorithm and thus the ACL security proofs still hold.

[10] This approach was also used by Abe for its blind signature scheme [1], on which ACL was based.

**Lemma 18.** *With the modification described above, for any adversary $\mathcal{A}$ making at most $Q_{\mathsf{H}_{\mathsf{sh}}}$ queries to the random oracle $\mathsf{H}_{\mathsf{sh}}$, it holds that* $\mathbf{Adv}^{\mathsf{DSP}}_{\mathsf{BSADS}}(\mathcal{A}) \leq \frac{Q^2_{\mathsf{H}_{\mathsf{sh}}}}{p}$.

*Proof.* Intuitively, the only way for the adversary to avoid being identified when double spending is to provoke a hash collision such that $c = c'$. We show this formally below using some game hops

*Game* $\mathbf{G}_1$. This is the original $\mathsf{DSP}$-game.

*Game* $\mathbf{G}_2$. This game is identical to $\mathbf{G}_1$ except that it aborts if there is a hash collision in the random oracle $\mathsf{H}_{\mathsf{sh}}$, i.e. two different queries by the adversary that lead to the same outcome.

It is easy to see that $\left|\Pr[\mathbf{G}^{\mathcal{A}}_2 = 1] - \Pr[\mathbf{G}^{\mathcal{A}}_1 = 1]\right| \leq \frac{Q^2_{\mathsf{H}_{\mathsf{sh}}}}{p}$.

We now show that the adversary $\mathcal{A}$ has success probability 0 in $\mathbf{G}_2$.

Since there is no hash collision, the two hashes $c$ and $c'$ are different. Consequently, the $\mathbf{G}_2$ computes and outputs $\gamma := \frac{\mu'_2 - \mu_2}{c - c'}$, and it holds $\gamma \neq \perp$ since $c - c' \neq 0$. This concludes the proof.

**Lemma 19.** *If the discrete logarithm problem is $(t, \varepsilon)$-hard in the group $\mathbb{G}$, then* $\mathsf{ACL}$*is $(t', \varepsilon')$-*$\mathsf{Exc}$ *secure in the AGM + ROM with $t' \approx t$ and*

$$\varepsilon' = \varepsilon + \frac{2Q_{\mathsf{H}_{\mathsf{sh}}} + \kappa + Q_{\mathsf{H}_{\mathsf{reg}}} + \ell}{p},$$

*where $Q_{\mathsf{H}_{\mathsf{sh}}}$ is the number of queries the adversary makes to $\mathsf{H}_{\mathsf{sh}}$, $Q_{\mathsf{H}_{\mathsf{reg}}}$ is the number if queries the adversary makes to $\mathsf{H}_{\mathsf{reg}}$, $\ell$ is the number of signing sessions closed with the user oracles, $\kappa$ is the number of registered users through the registration oracle.*

*Proof Idea.* In order to successfully frame a user, the adversary needs to output a signature along with two showing proofs $\bar{\pi}_1, \bar{\pi}_2$ as the $\mathcal{DSI}$ algorithm and provide values $\gamma$ and rnd that blinds the user's commitment into $\zeta_1$. As the signer can interact with itself to issue a signature, thus knowing $\gamma$ and rnd, we show that it is hard for the signer to provide the partial proofs of opening of the commitment by extracting a discrete logarithm solution from a signer that does so.

*Proof.* We do a quick game hop to exclude some situations where the reduction cannot win related to RO queries.

*Game* $\mathbf{G}_1$. This is the original $\mathsf{Exc}$-game.

*Game* $\mathbf{G}_2$. This game aborts whenever the challenger queries the hash oracles $\mathsf{H}_{\mathsf{reg}}$ and $\mathsf{H}_{\mathsf{sh}}$ during computation of a proof $\pi$ or $\bar{\pi}$, and the hash oracle has been queried on the same value before. As the challenger's queries to its own oracle contain first flows of the proofs $\pi, \bar{\pi}$, the probability for each first flow to have been queried before is at most $\frac{Q_{\mathsf{H}_{\mathsf{reg}}} + \kappa}{p}$ for $\mathsf{H}_{\mathsf{reg}}$ and $\frac{Q_{\mathsf{H}_{\mathsf{sh}}} + \ell}{p}$ for $\mathsf{H}_{\mathsf{sh}}$.

*Game* $\mathbf{G}_3$. This game introduces an additional abort condition related to the representations submitted by the adversary to the oracle. When the adversary outputs a proof of double spending $(C, ID, \pi, \mathrm{rnd}, \gamma, m, \sigma, \bar{\pi_1}, \bar{\pi_2})$, at least one of $\bar{\pi_1}$ and $\bar{\pi_2}$ was not generated by the showing oracle. Let $\bar{\pi} = (\pi_{sdl}, \pi_{op})$ be the showing proof that was not an output of the showing oracle. We consider the $\pi_{op} = (R_{op}, c, \boldsymbol{S}_{op})$ component.

We denote by $G_{[R]}$ the exponent of a group element $G$ in the representation of $R$, where $G$ can be any group element that was output by the challenger to the adversary through one of the oracles. The game aborts if

$$c = \sum_{j=1}^{\kappa} C_{j\,[R]} + \pi.c \cdot \pi.R_{j\,[R]} + \sum_{j=1}^{\ell} \gamma_j \cdot \zeta_{1,j\,[R]} + \sum_{j=1}^{\ell} \pi_{op}.c_j \cdot R_{j\,[R]},$$

where $\pi_{op}.c_j$ is the value $c$ computed as part of the showing, $\gamma_j$ is the $\gamma$ value chosen in the $j_{th}$ signing session, $\kappa$ is the number of registered users, and $C_j$ is the commitment of the $j_{th}$ registered user, $\pi.R_j$ is the first flow of the proof of (partial) opening of the registration commitment of the $j_{th}$ registered user, and $c$ is the hash output used in $\pi_{op}$. As the representations of $\gamma_j$ and $\pi.c$ are already fixed by the time the adversary receives the hash output, the probability of the above event happening is at most $\frac{Q_{\mathsf{H_{sh}}}}{p}$.

We describe a reduction that breaks the discrete logarithm problem if the adversary wins $\mathbf{G}_3$.

The reduction simulates the $\mathbf{G}_3$ game as follows to the adversary:

**Setup.** The reduction takes as input a discrete logarithm challenge $g, X$. It invokes the adversary $\mathcal{A}$ with the oracles described below. Keep a list of all group elements submitted by the adversary to the oracles along with their algebraic representations.

**Online Phase.** The reduction simulates the oracles as follows to the adversary:

- **Hash oracles** $\mathsf{H_{pp}}, \mathsf{H_{reg}}, \mathsf{H}_1, \mathsf{H}_3, \mathsf{H_{sh}}$ lazy sampling from the appropriate image spaces, in case of group element image spaces, sample the discrete logarithm to base $g$ of the group element from $\mathbb{Z}_p$ and keep the list of discrete logarithms along with the inputs and group element oracle outputs.
- **Registration queries** Takes as input the vector to be registered $\boldsymbol{L}$. Sample $r_{com}{}' \leftarrow_{\$} \mathbb{Z}_p$. Compute the commitment $C := h_0^{r_{com}{}'} X \cdot \prod_{i=1}^{\nu} h_i^{L_i}$ which implicitly sets $r_{com} := r_{com}{}' \cdot \mathrm{dlog} X$. Simulate the proof $\pi$ by programming the RO $\mathsf{H_{reg}}$.
- **User queries** participate in the signing protocol like an honest user would and output the resulting signatures to the signer.
- **Showing queries** On input of a signature previously output by the user oracle and an opening set $I_{op}$, produce an opening $\bar{\pi}$ to this opening index set by simulating the proof by programming the random oracle $\mathsf{H_{sh}}$.

**Output Determination.** The reduction retrieves the algebraic representation of $R$ that the adversary submitted when querying the random oracle $\mathsf{H_{sh}}$ to

generate $\bar{\pi}$. It computes a so-called *reduced representation*[11] of $R$ to the base $\boldsymbol{I} = (g, X)$ as follows: We write $R = X^{X_{[R]\boldsymbol{I}}} \cdot g^{g_{[R]\boldsymbol{I}}}$, where

$$X_{[R]_{\boldsymbol{I}}} \coloneqq \sum_{j=1}^{\kappa} C_{j\,[R]} + \pi.c \cdot \pi.R_{j\,[R]} + \sum_{j=1}^{\ell} \gamma_j \cdot \zeta_{1,j\,[R]} + \sum_{j=1}^{\ell} \pi_{op}.c_j \cdot R_{j\,[R]}$$

and $g_{[X]_{\boldsymbol{I}}} \coloneqq \mathrm{dlog}_g(R \cdot X^{-X_{[R]\boldsymbol{I}}})$. The reduction can compute $g_{[X]_{\boldsymbol{I}}}$ using the discrete logarithms of all group elements in its output to the base $g$, as any $X$ components have been removed from this equation.
If $c = X_{[R]_{\boldsymbol{I}}}$, the reduction aborts as this corresponds to the abort condition introduced in $\mathbf{G}_3$. Otherwise, it computes

$$\mathrm{dlog}_g X = \frac{g_{[R]_{\boldsymbol{I}}} - (\sum_{i=0}^{\nu} s_i + s_\Gamma + c \cdot (r_{com}{}' + \sum_{i=1}^{\nu} L_i + \mathrm{rnd}))}{c - X_{[R]_{\boldsymbol{I}}}}.$$

The reduction outputs a valid $\mathsf{DLP}$ solution whenever the adversary wins $\mathbf{G}_3$. Thus, putting together the probabilities from the game hops yields

$$\mathbf{Adv}^{\mathsf{DLP}}_{\mathbb{G}}(\mathcal{R}) = \mathbf{Adv}^{\mathsf{Exc}}_{\mathsf{ACL}}(\mathcal{A}) - \frac{Q_{\mathsf{H}_{\mathsf{sh}}}}{p} - \frac{Q_{\mathsf{H}_{\mathsf{reg}}} + \kappa}{p} - \frac{Q_{\mathsf{H}_{\mathsf{sh}}} + \ell}{p}$$

which yields the claim.

## H Proofs of of Lemmas 15 and 16

Next, we propose a simplified variant of the restrictive blinding game from [1,44] where we do not require that the adversary's output contains as many signatures as it requested signing sessions, but rather only a "mismatched" signature.

**Definition 27 (Restrictive Blinding Game).** *We describe the* restrictive blinding *game for an adversary $\mathcal{A}$:*

**Setup:** *The challenger samples a key pair $(\mathsf{pk}, \mathsf{sk})$ according to $\mathcal{G}(\mathsf{pp}_{\mathbb{G}})$. This uses the random oracle $\mathsf{H}_1$ which allows the challenger to keep track of the discrete logarithms of its outputs. It outputs $\mathsf{pk}$ to the adversary.*
**Online Phase:** *The adversary gets access to the following oracles*
**Oracle $\mathsf{H}_1$:** *lazily samples $w \leftarrow_\$ \mathbb{Z}_p$ and returns $g^w$. Keeps a list of the hash inputs, $w$, and the output $g^w$.*
**Oracle $\mathsf{H}_{\mathsf{pp}}$:** *lazily samples $v \leftarrow_\$ \mathbb{Z}_p$ and returns $g^v$. Keeps a list of the hash inputs $v$, and the output $g^v$.*
**Oracle $\mathsf{H}_3$:** *lazily samples $\varepsilon \leftarrow_\$ \mathbb{Z}_p$ and returns $\varepsilon$*
**$\mathcal{RG}_{\mathsf{S}}$:** *Register new commitments and store them in list to ensure all commitments coming in $\mathcal{S}_1$ queries were already registered.*
**$\mathcal{S}_1$:** *runs the signing procedure $\mathcal{S}_1$ of $\mathsf{ACL}$ and returns its output along with a fresh session ID. Keeps a list of the commitment $C$ and the random value rnd.*

[11] This strategy has been used before in [44,41].

$\mathcal{S}_2$**:** *takes as input a session ID and a challenge $e$. If the session ID has never been used before in a $\mathcal{S}_2$ query, it runs the $\mathcal{S}_2$ procedure of* $\mathsf{ACL}$ *using the internal signer state corresponding to the session ID with $e$ and returns its output.*

**Output Determination:** *$\mathcal{A}$ submits a list of message-signature pairs $(m_i, \sigma_i)$ for $i = 1..\ell$. The game outputs 1 if there is a pair $(m, \sigma = (\zeta, \zeta_1, \ldots))$ in the list such that the signature is valid on the message $m$ under the public key* $\mathsf{pk}$ *and it holds that there exists no closed signing session* $\mathsf{sid}$ *such that* $\mathrm{dlog}_z \zeta = \mathrm{dlog}_{z_{1,\mathsf{sid}}} \zeta_1$.

*We define the advantage of the adversary in the game* $\mathsf{RB}$ *as*

$$\mathbf{Adv}^{\mathsf{RB}}_{\mathsf{ACL}}(\mathcal{A}) \coloneqq \Pr[\mathsf{RB}^{\mathcal{A}}_{\mathsf{ACL}} = 1].$$

**Lemma 15 (Restrictive Blinding).** *In the ROM and AGM, if* $\mathsf{DLP}$ *is $(t, \varepsilon)$-hard in $\mathbb{G}$, it holds that for any adversary running in time at most $t$, the advantage is at most*

$$\mathbf{Adv}^{\mathsf{RB}}_{\mathsf{ACL}}(\mathcal{A}) \leq 13 \cdot \varepsilon + \frac{4 \cdot Q_C + 31}{p}.$$

*where $Q_C$ is the number of registration queries made by the adversary.*

*Proof.* We use a similar proof technique to the one used in Theorem 4.1. in [41] (and full version [42]). We proceed in a series of games to gradually rule out bad event, reaching a game that we can reduce $\mathsf{DLP}$ if $\mathcal{A}$ outputs a valid signature such that there exists no closed signing session $\mathsf{sid}$ satisfying $\mathrm{dlog}_z \zeta = \mathrm{dlog}_{z_{1,\mathsf{sid}}} \zeta_1$.

We start by ruling out the first abort condition corresponding to the representation of the registered commitments $C$.

*Game* $\mathbf{G}_1$. This is Game $\mathsf{RB}$.

Let $Q_C$ be the number of commitments $\mathcal{A}$ registers. We define $\mathbf{BAD}_1$ as the event in which $\mathcal{A}$ registers a commitment $C_j$ for $j \in [Q_C]$ successfully using a proof $\pi = (M, s_0, \ldots, s_\nu)$ while $\mathsf{T}_{\mathsf{pp}}[(C, M)] = \bot$, where $\mathsf{T}_{\mathsf{P}}$ is the table of the random oracle $\mathsf{H}_{\mathsf{reg}}$ used in the Fiat-Shamir transformation of the protocol from construction 3. Intuitively, this corresponds to $\mathcal{A}$ registering a commitment $C_j$ successfully without having made a hash query beforehand.

*Game* $\mathbf{G}_2$. This game is identical to Game $\mathbf{G}_1$, except that it aborts if Event $\mathbf{BAD}_1$ occurs.

*Claim 26.*

$$\Pr[\mathbf{BAD}_1] \leq \frac{Q_C}{p}.$$

*Proof.* Outputting a valid proof $\pi = (M, s_0, \ldots, s_\nu)$ for a commitment $C$ implies that the equation $C^c \cdot \prod_{i \in [\![\nu]\!]} h_i^{s_i} = M$, where $c = \mathsf{H}_{\mathsf{reg}}(C, M)$. Thus, outputting a valid proof without querying $\mathsf{H}_{\mathsf{reg}}$ beforehand to acquire $c$ is equivalent to guessing $c$. However, sine $\mathsf{H}_{\mathsf{reg}}$ is modeled as a random oracle, the value $c$ is chosen

uniformly at random, and the probability of $\mathcal{A}$ guessing this value correctly is at most $\frac{1}{p}$. Since $\mathcal{A}$ registers up to $Q_C$ values, the probability that it successfully registers some commitment without having made a corresponding hash query is at most $\frac{Q_C}{p}$.

By this claim, it holds $\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_2} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_1} - \frac{Q_C}{p}$.

As $\mathbf{BAD}_1$ does not occur, we assume in the following that for each valid commitment-proof tuple $(C, \pi = (M, s_0, \ldots, s_\nu))$ the adversary outputs, it holds that $\mathsf{T}_\mathsf{P} \neq \bot$.

Let $\mathbf{BAD}_2$ be the event in which $\mathcal{A}$ successfully registers a commitment $C_j$ for $j \in [Q_C]$ satisfying the condition

$$\bigvee_{j \in [Q_C]} y_{[C_j]} + \sum_{\mathsf{sid} \in Q_\mathsf{S}} c_\mathsf{sid} \cdot a_{\mathsf{sid}[C_j]} \neq 0,$$

where $Q_\mathsf{S}$ is the number of open signing sessions at the time $\mathcal{A}$ registers $C_j$, and $c_\mathsf{sid}$ is the random chosen by the signer to compute $a_\mathsf{sid}$. Intuitively, this condition means that $\mathcal{A}$ uses $y$ or some $a_\mathsf{sid}$ (or both) in the representation of some commitment $C_j$ such that they do not cancel each other out.

*Game* $\mathbf{G}_3$. This game is identical to $\mathbf{G}_2$, except that it aborts if event $\mathbf{BAD}_2$ occurs.

*Claim 27.*

$$\Pr[\mathbf{BAD}_2] \leq \mathbf{Adv}^{\mathcal{R}_1}_{\mathsf{DLP}} + \frac{1}{p}.$$

*Proof.* It follows directly from Claim E.1 in [41].

By the claim, we have $\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_3} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_2} - \mathbf{Adv}^{\mathcal{R}_1}_{\mathsf{DLP}} - \frac{1}{p}$.

By this claim, we assume that $\mathcal{A}$ wins $\mathbf{G}_3$ and $\mathbf{BAD}_2$ does not occur. However, $\mathcal{A}$ may still use $a_\mathsf{sid}$ or $y$ components in the representation of $C$ without triggering $\mathbf{BAD}_2$ in a way that prevents $\mathcal{R}$ from functioning properly. Specifically, if $\mathcal{A}$ uses the group element $a_\mathsf{sid}$ or $b_{2,\mathsf{sid}}$ of the signing session $\mathsf{sid}$ in the representation of some commitment $C$, but it never closes $\mathsf{sid}$, this may lead to an issue for $\mathcal{R}$, because it won't be able to compute $\mathrm{dlog}_g C$ without knowing the opening of $a_\mathsf{sid}$ or $b_\mathsf{sid}$, i.e. $(r_\mathsf{sid}, c_\mathsf{sid})$ and $(s_{2,\mathsf{sid}}, d_\mathsf{sid})$, respectively. Note that this does not cause an issue for the current game hop, because it is straight forward to known the opening of all group elements of any signing session $\mathsf{sid}$ since the signer oracles are simulated by the reduction. However, this is problematic later in the reduction from Game $\mathsf{OMMIM}$ to Game $\mathsf{OMUF}$, because $\mathcal{R}$ uses the challenger's prover oracles to simulate the signer oracles for $\mathcal{A}$. Consequently, the reduction will have access to the opening of a group element $a_\mathsf{sid}$ or $b_\mathsf{sid}2$ iff the adversary makes $\mathsf{S2}$ query for sessions $\mathsf{sid}$. Therefore, it is crucial to ensure that no $a_\mathsf{sid}$ or $b_{2,\mathsf{sid}}$ is used in the representation of any commitment $C$ if $\mathsf{sid}$ will not be closed. To capture this, we define $\mathbf{BAD}_3$ as the event that occurs if the equation

$$\bigvee_{j\in[Q_C]}(y_{[C_j]}+\sum_{\mathsf{sid}\in \boldsymbol{I}_{op}} c_{\mathsf{sid}}\cdot a_{\mathsf{sid}[C_j]}=-\sum_{\mathsf{sid}\in \boldsymbol{I}_{cl}} c_{\mathsf{sid}}\cdot a_{\mathsf{sid}[C_j]}).$$

*Game* $\mathbf{G}_4$. This game is identical to $\mathbf{G}_3$, except that it aborts if event $\mathbf{BAD}_3$ occurs.

*Claim 28.*

$$\Pr[\mathbf{BAD}_3]\leq\frac{Q_C}{p}.$$

*Proof.* Identical to Claim E.2 in [41].

Consequently, we have $\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_4}\geq\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_3}-\frac{Q_C}{p}$.

Next, we rule out the usage of any $h$ or $b_{2,\mathsf{sid}}$ component in the representation of $C_j$ for $\mathsf{sid}\in Q_{\mathsf{S}}$ and $j\in Q_C$. We define Event $\mathbf{BAD}_4$ as the event that occurs if

$$\bigvee_{j\in[Q_C]}(h_{[C_j]}+\sum_{\mathsf{sid}\in Q_{\mathsf{S}}} s_{2,\mathsf{sid}}\cdot b_{2,\mathsf{sid}[C_j]})\neq 0$$

holds, where $C_j$ are the commitments registered by $\mathcal{A}$. This condition means that $\mathcal{A}$ uses $h$ or some $b_{2,\mathsf{sid}}$) (or both) in the representation of some commitment $C_j$ such that they do not cancel each other out.

*Game* $\mathbf{G}_5$. This game is identical to $\mathbf{G}_4$, except that it aborts if event $\mathbf{BAD}_4$ occurs.

*Claim 29.*

$$\Pr[\mathbf{BAD}_4]\leq\mathbf{Adv}^{\mathcal{R}_2}_{\mathsf{DLP}}+\frac{1}{p}.$$

*Proof.* Identical to Claim E.3 in [41].

Per this claim, we have $\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_5}\geq\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_4}-\mathbf{Adv}^{\mathcal{R}_2}_{\mathsf{DLP}}-\frac{1}{p}$.

Next, we prevent $\mathcal{A}$ from using $b_{2,\mathsf{sid}}$ from an open signing session $\mathsf{sid}$ in the representation of some $C_j$ for $j\in[Q_C]$ without triggering $\mathbf{BAD}_4$.

Define $\mathbf{BAD}_5$ as the event that occurs if

$$\bigvee_{j\in[Q_C]}(h_{[C_j]}+\sum_{\mathsf{sid}\in \boldsymbol{I}_{cl}} s_{2,\mathsf{sid}}\cdot b_{2,\mathsf{sid}[C_j]}=-\sum_{\mathsf{sid}\in \boldsymbol{I}_{op}} s_{2,\mathsf{sid}}\cdot b_{2,\mathsf{sid}[C_j]}).$$

This condition means that $\mathcal{A}$ uses $h$ or some $b_{2,\mathsf{sid}}$ (or both) in the representation of some commitment $C_j$ such that they do not cancel each other out.

*Game* $\mathbf{G}_6$. This game is identical to $\mathbf{G}_5$, except that it aborts if event $\mathbf{BAD}_5$ occurs.

*Claim 30.*

$$\Pr[\mathbf{BAD}_5] \leq \frac{Q_C}{p}.$$

*Proof.* The same as Claim E.4 in [41].

By the claim, it holds $\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_6} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_5} - \frac{Q_C}{p}$.

Let Event $\mathbf{BAD}_6$ be the event that occurs if

$$\bigvee_{j \in [Q_C]} (z_{[C_j]} + \sum_{\mathsf{sid} \in Q_{\mathsf{S}}} d_{\mathsf{sid}} \cdot b_{2,\mathsf{sid}[C_j]} \neq 0).$$

holds. This condition is equivalent to $\mathcal{A}$ using $z$ or some $b_{2,\mathsf{sid}}$ (or both) in the representation of some commitment $C_j$ without canceling each other out.

*Game* $\mathbf{G}_7$. This game is identical to $\mathbf{G}_6$, except that it aborts if event $\mathbf{BAD}_6$ occurs.

*Claim 31.*

$$\Pr[\mathbf{BAD}_6] \leq \mathbf{Adv}^{\mathcal{R}_3}_{\mathsf{DLP}} + \frac{1}{p}.$$

*Proof.* The same as Claim E.5 in [41].

It follows that $\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_7} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_6} - \mathbf{Adv}^{\mathcal{R}_3}_{\mathsf{DLP}} - \frac{1}{p}$.

Similar to what we did in Games $\mathbf{G}_4$ and $\mathbf{G}_6$, we rule out using $b_{2,\mathsf{sid}}$ from a signing session $\mathsf{sid}$ that is never closed in the representation of some commitment $C_j$ without triggering $\mathbf{BAD}_6$.

Let Event $\mathbf{BAD}_7$ be the event that occurs if

$$\bigvee_{j \in [Q_C]} z_{[C_j]} + \sum_{\mathsf{sid} \in \boldsymbol{I}_{op}} d_{\mathsf{sid}} \cdot b_{2,\mathsf{sid}[C_j]} = - \sum_{\mathsf{sid} \in \boldsymbol{I}_{cl}} d_{\mathsf{sid}} \cdot b_{2,\mathsf{sid}[C_j]}.$$

holds.

*Game* $\mathbf{G}_8$. This game is identical to $\mathbf{G}_7$, except that it aborts if event $\mathbf{BAD}_7$ occurs.

*Claim 32.*

$$\Pr[\mathbf{BAD}_7] \leq \frac{Q_C}{p}.$$

*Proof.* Identical to Claim E.6 in [41].

It follows that $\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_8} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_7} - \frac{Q_C}{p}$.

Let $\mathsf{open_j}$ be a vector of all signing sessions identifiers that are still open at the time $\mathcal{A}$ registers a commitment $C_j$ for $j \in [Q_C]$, ordered by the time the sessions were opened. Let Event $\mathbf{BAD}_8$ be the event that occurs if

$$\bigvee_{j \in [Q_C]} \sum_{\mathsf{sid} \in \mathsf{open_j}} u_{3,\mathsf{sid}} \cdot b_{1,\mathsf{sid}[C_j]} \neq 0,$$

holds, where $u_{3,\mathsf{sid}} \coloneqq \mathrm{dlog}_g b_{1,\mathsf{sid}}$.

*Claim 33.*

$$\Pr[\mathbf{BAD}_8] \leq \frac{1}{\nu} \cdot \mathbf{Adv}^{\mathcal{R}_4}_{\mathsf{DLP}} + \frac{3}{p}$$

*Proof.* Identical to Claim E.7 in [41].

By this claim, it holds that $\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_8} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_7} - \frac{1}{\nu} \cdot \mathbf{Adv}^{\mathcal{R}_4}_{\mathsf{DLP}} - \frac{3}{p}$.

**Corollary 3.** *Given a representation of a successfully registered commitment $C$, $\mathcal{R}$ can efficiently compute* $\mathrm{dlog}_g C$.

*Proof.* If $\mathcal{A}$ wins $\mathbf{G}_8$, non of the abort conditions of the previous games (i.e., $\mathbf{BAD}_1 - \mathbf{BAD}_8$) was triggered, except with negligible probability. In this case, the representations of all registered commitments $C$ do not contain any group elements with unknown discrete logarithms to $\mathcal{R}$, such as the group elements of the challenger's public key or the group elements $\mathcal{A}$ receiver via $\mathcal{S}_1$ queries. Thus, $\mathcal{R}$ can efficiently compute $\mathrm{dlog}_g C$.

Next, we address the second abort condition, which relates to the representation of the group element $\zeta$.

Let $\mathbf{BAD}_9$ be the event in which $\mathcal{A}$ outputs a valid signature $(\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \mu, \delta)$ for a message $m$, while $\mathsf{T_3}[(\zeta, \zeta_1, g^\rho \cdot y^\omega, g^{\sigma_1} \cdot \zeta_1^\delta, h^{\sigma_2} \cdot \zeta_2^\delta, z^\mu \cdot \zeta^\delta, m)] = \bot$. This corresponds to $\mathcal{A}$ outputting a valid signature without making a corresponding hash query to $\mathsf{H}_3$ beforehand.

*Game* $\mathbf{G}_9$. This game is the same as $\mathbf{G}_8$ except that it aborts if Event $\mathbf{BAD}_9$ occurs.

*Claim 34.*

$$\Pr[\mathbf{BAD}_9] \leq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_8} + \frac{1}{p}.$$

*Proof.* A valid signature must satisfy the verification equation $\omega + \delta = \mathsf{H}_3(\zeta, \zeta_1, g^\rho \cdot y^\omega, g^{\sigma_1} \cdot \zeta_1^\delta, h^{\sigma_2} \cdot \zeta_2^\delta, z^\mu \cdot \zeta^\delta, m)$. However, outputting a valid signature without querying $\mathsf{H}_3$ beforehand is equivalent to guessing the output of $\mathsf{H}_3$, i.e. $\omega + \delta$, on input $(\zeta, \zeta_1, g^\rho \cdot y^\omega, g^{\sigma_1} \cdot \zeta_1^\delta, h^{\sigma_2} \cdot \zeta_2^\delta, z^\mu \cdot \zeta^\delta, m)$. As $\mathsf{H}_3$ is modeled as a random oracle, its output is chosen uniformly at random from $\mathbb{Z}_p$ and the probability that $\mathcal{A}$ succeeds in guessing its output is at most $\frac{1}{p}$.

By the claim, it holds $\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_9} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_8} - \frac{1}{p}$.

Define Event $\mathbf{BAD}_{10}$ as the event that occurs if $\mathcal{A}$ outputs a valid signature $(\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \mu, \delta)$ for a message $m$ with a corresponding $\mathsf{H}_3$ hash query $(\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta, m)$, and it holds $z^{z_{[\zeta]}} \neq \zeta$. That is, the event occurs if the representation of $\zeta$ contains any group element other than $z$.

*Game* $\mathbf{G}_{10}$. This game is identical to $\mathbf{G}_9$, except that it aborts if $\mathbf{BAD}_{10}$ occurs.

*Claim 35.*

$$\Pr[\mathbf{BAD}_{10}] \leq \mathbf{Adv}^{\mathcal{R}_5}_{\mathsf{DLP}} + 2 \cdot \mathbf{Adv}^{\mathcal{R}_6}_{\mathsf{DLP}} + \frac{6}{p}.$$

*Proof.* Identical to Claim E.11 in [41].

By this claim, we have $\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_{10}} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_9} - \mathbf{Adv}^{\mathcal{R}_5}_{\mathsf{DLP}} - 2 \cdot \mathbf{Adv}^{\mathcal{R}_6}_{\mathsf{DLP}} - \frac{6}{p}$, and therefore, we assume $\mathcal{A}$ wins $\mathbf{G}_{10}$ and $\zeta$ is computed honestly, i.e. its representation contains only a $z$ component.

**Corollary 4.** *For any valid signature $(\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \mu, \delta)$ with a corresponding $\mathsf{H}_3$ hash query $(\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta, m)$, $\mathcal{R}$ can use the representation of $\zeta$ submitted along the hash query to efficiently compute $\mathrm{dlog}_z \zeta$.*

*Proof.* If $\mathcal{A}$ wins $\mathbf{G}_{10}$, the abort conditions $\mathbf{BAD}_9$ and $\mathbf{BAD}_{10}$ cannot be triggered, except with probability $\frac{7}{p} + \mathbf{Adv}^{\mathcal{R}5}_{\mathsf{DLP}} + 2 \cdot \mathbf{Adv}^{\mathcal{R}6}_{\mathsf{DLP}}$. In this case, the representation of the group element $\zeta$ of every hash query corresponding to some valid signature only consists solely of a $z$ component, i.e. $\zeta = g^{z_{[\zeta]}}$. Consequently, $\mathcal{R}$ can efficiently compute $\mathrm{dlog}_z \zeta$, except with negligible probability.

Let Event $\mathbf{BAD}_{11}$ be the event that occurs if $\mathcal{A}$ outputs a valid signature $(\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \mu, \delta)$ for a message $m$ with a corresponding $\mathsf{H}_3$ hash query $(\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta, m)$ satisfying $\mathrm{dlog}_z z_{1,\mathsf{sid}} \neq \mathrm{dlog}_\zeta \zeta_1$ for all signing session $\mathsf{sid}$.

*Game* $\mathbf{G}_{11}$. This game is identical to $\mathbf{G}_{10}$, except that it aborts if $\mathbf{BAD}_{11}$ occurs.

*Claim 36.*

$$\Pr[\mathbf{BAD}_{11}] \leq \mathbf{Adv}^{\mathcal{R}_7}_{\mathsf{DLP}} + \mathbf{Adv}^{\mathcal{R}_8}_{\mathsf{DLP}} + 4 \cdot \mathbf{Adv}^{\mathcal{R}_9}_{\mathsf{DLP}} + \frac{18}{p}.$$

*Proof.* The same as Claim E.11 in [41].

By this claim, we have $\mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_{11}} \geq \mathbf{Adv}^{\mathcal{A}}_{\mathbf{G}_{10}} - \mathbf{Adv}^{\mathcal{R}7}_{\mathsf{DLP}} - \mathbf{Adv}^{\mathcal{R}8}_{\mathsf{DLP}} - 4 \cdot \mathbf{Adv}^{\mathcal{R}9}_{\mathsf{DLP}} - \frac{18}{p}$. Therefore, It must hold that each valid signature output by $\mathcal{A}$ is linked to some signing session $\mathsf{sid}$ by satisfying $\mathrm{dlog}_z z_{1,\mathsf{sid}} = \mathrm{dlog}_\zeta \zeta_1$.

**Corollary 5.** *For any valid signature $(\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \mu, \delta)$ with a corresponding $\mathsf{H}_3$ hash query $(\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta, m)$, let $\gamma := \mathrm{dlog}_z \zeta$. Then there exists a signing session $\mathsf{sid}$ such that $\zeta_1^{1/\gamma} = z_{1,\mathsf{sid}}$, hence $\mathrm{dlog}_g \zeta_1^{1/\gamma} = rnd_{\mathsf{sid}}$.*

*Proof.* If $\mathcal{A}$ wins $\mathbf{G}_{11}$, the abort condition $\mathbf{BAD}_{11}$ was not triggered, except with negligible probability. It follows that any valid signature satisfies $\mathrm{dlog}_z z_{1,\mathsf{sid}} = \mathrm{dlog}_\zeta \zeta_1$ for some signing session $\mathsf{sid}$.

Define Event $\mathbf{BAD}_{11}{}'$ as the event that occurs if $\mathcal{A}$ outputs a valid signature $(\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \mu, \delta)$ with a corresponding $\mathsf{H}_3$ hash query $(\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta, m)$ such that for all signing sessions $\mathsf{sid}$ it holds $\mathrm{dlog}_z \zeta = \mathrm{dlog}_{z_{\mathsf{sid}1}} \zeta_1$.

**Corollary 6.**

$$\Pr[\mathbf{BAD}_{11}{}'] \leq \mathbf{Adv}^{\mathcal{R}7}_{\mathsf{DLP}} + \mathbf{Adv}^{\mathcal{R}8}_{\mathsf{DLP}} + 4 \cdot \mathbf{Adv}^{\mathcal{R}9}_{\mathsf{DLP}} + \frac{18}{p}.$$

*Proof.* We know that $\Pr[\mathbf{BAD}_{11}] \leq \mathbf{Adv}^{\mathcal{R}7}_{\mathsf{DLP}} + \mathbf{Adv}^{\mathcal{R}8}_{\mathsf{DLP}} + 4 \cdot \mathbf{Adv}^{\mathcal{R}9}_{\mathsf{DLP}} + \frac{18}{p}$, thus, it suffices to show that $\mathbf{BAD}_{11} \Leftrightarrow \mathbf{BAD}_{11}{}'$. Let $\gamma \coloneqq \mathrm{dlog}_z \zeta$. It follows that $\mathrm{dlog}_z z_{1,\mathsf{sid}} = \mathrm{dlog}_{z^\gamma} \zeta_1 =: v_0$, hence $\zeta_1 = z^{v_0 \cdot \gamma} = z^\gamma_{1,\mathsf{sid}}$. It follows that $\mathrm{dlog}_z z_{1,\mathsf{sid}} = \mathrm{dlog}_\zeta \zeta_1 \Leftrightarrow \mathrm{dlog}_z \zeta = \mathrm{dlog}_{z_{1,\mathsf{sid}}} \zeta_1$.

Due to Corollary 6, we have

$$\mathbf{Adv}^{\mathsf{RB}}_{\mathsf{ACL}}(\mathcal{A}) \leq \frac{4 \cdot Q_C + 31}{p} + \mathbf{Adv}^{\mathcal{R}1}_{\mathsf{DLP}} + \mathbf{Adv}^{\mathcal{R}2}_{\mathsf{DLP}} + \mathbf{Adv}^{\mathcal{R}3}_{\mathsf{DLP}}$$
$$+ \frac{1}{\nu} \cdot \mathbf{Adv}^{\mathcal{R}4}_{\mathsf{DLP}} + \mathbf{Adv}^{\mathcal{R}5}_{\mathsf{DLP}} + 2 \cdot \mathbf{Adv}^{\mathcal{R}6}_{\mathsf{DLP}} + \mathbf{Adv}^{\mathcal{R}7}_{\mathsf{DLP}} + \mathbf{Adv}^{\mathcal{R}8}_{\mathsf{DLP}} + 4 \cdot \mathbf{Adv}^{\mathcal{R}9}_{\mathsf{DLP}},$$

hence

$$\mathbf{Adv}^{\mathsf{RB}}_{\mathsf{ACL}}(\mathcal{A}) \leq 13 \cdot \varepsilon + \frac{4 \cdot Q_C + 31}{p}.$$

**Definition 28 (2-to-1 tagging game).** *For a positive integer $\ell \in \mathbb{Z}^+$, we define the game $\ell$-RT for the identification scheme $\mathsf{ID}_{\mathsf{ACL}} \coloneqq (\mathcal{G}, \mathcal{P}, \mathcal{V})$ and an adversary $\mathcal{A}$ as follows.*

**Initialization.** *The challenger samples a key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathcal{G}(\mathsf{pp}_\mathbb{G})$. It outputs $\mathsf{pk}$ to the adversary $\mathcal{A}$.*

**Online Phase.** *The adversary gets to interact with prover oracles $\mathcal{P}_1$ and $\mathcal{P}_2$ and verifier oracles $\mathcal{V}_1$, $\mathcal{V}_2$.*

**Output Determination.** *The game outputs 1 there are two accepting sessions $i_1, i_2$ with the verifier, such that there exists a $\mathcal{P}_2$ invocation $j$, it holds that $\zeta_{i_1,i} = z_{1,j} = \zeta_{i_2,i}$. We call these two verifier sessions the* special sessions.

*We define the advantage of an adversary $\mathcal{A}$ in winning the game for the $\mathsf{ACL}$ identification scheme $\mathsf{ID}_{\mathsf{ACL}}$ as*

$$\mathbf{Adv}^{\ell\text{-}\mathsf{TTO}}_{\mathsf{ID}_{\mathsf{ACL}}}(\mathcal{A}) \coloneqq \Pr[\ell\text{-}\mathsf{TTO}^{\mathcal{A}}_{\mathsf{ID}_{\mathsf{ACL}}} = 1].$$

**Lemma 20 (2-to-1 tagging (adapted from Lemma 3.5 in [41])).** *There exists an adversary $\mathcal{R}$ against the discrete logarithm problem such that for any algebraic adversary $\mathcal{A}$ it holds that*

$$\mathbf{Adv}^{\mathsf{DLP}}(\mathcal{R}) \geq \frac{1}{2} \mathbf{Adv}^{\ell\text{-}\mathsf{RT}}_{\mathsf{ID}_{\mathsf{ACL}}}(\mathcal{A}) - \frac{1}{p}.$$

*Proof.* The proof is identical to the proof of Lemma 3.5 in [41] as their proof only uses the fact that there are two special sessions with the same tag.

**Definition 29 (2-to-1 Game).** *We describe the* 2-to-1 *game* $\ell$-TTO *for an adversary* $\mathcal{A}$*:*

**Setup:** *The challenger samples a key pair* $(\mathsf{pk}, \mathsf{sk})$ *according to* $\mathcal{G}(\mathsf{pp}_{\mathbb{G}})$*. This uses the random oracle* $\mathsf{H}_1$ *which allows the challenger to keep track of the discrete logarithms of its outputs. It outputs* pk *to the adversary.*

**Online Phase:** *The adversary gets access to the following oracles*

**Oracle** $\mathsf{H}_1$**:** *lazily samples* $w \leftarrow_\$ \mathbb{Z}_p$ *and returns* $g^w$*. Keeps a list of the hash inputs,* $w$*, and the output* $g^w$*.*

**Oracle** $\mathsf{H}_{\mathsf{pp}}$**:** *lazily samples* $v \leftarrow_\$ \mathbb{Z}_p$ *and returns* $g^v$*. Keeps a list of the hash inputs* $v$*, and the output* $g^v$*.*

**Oracle** $\mathsf{H}_3$ *lazily samples* $\varepsilon \leftarrow_\$ \mathbb{Z}_p$ *and returns* $\varepsilon$

$\mathcal{S}_1$**:** *runs the signing procedure* $\mathcal{S}_1$ *of* ACL *and returns its output along with a fresh session ID. Keeps a list of the commitment* $C$ *and the*

$\mathcal{S}_2$**:** *takes as input a session ID and a challenge* $e$*. If the session ID has never been used before in a* $\mathcal{S}_2$ *query, it runs the* $\mathcal{S}_2$ *procedure of* ACL *using the internal signer state corresponding to the session ID with* $e$ *and returns its output.*

**Output Determination:** *The adversary submits a list of message-signature pairs* $(m_i, \sigma_i)$ *for* $i = 1..\ell$*. The game outputs* 1 *if there are two messages* $m_1$*,* $m_2$ *and two signatures* $(\zeta^{(1)}, \zeta_1^{(1)}, \rho^{(1)}, \omega^{(1)}, \sigma_1^{(1)}, \sigma_2^{(1)}, \mu^{(1)}, \delta^{(1)})$ *and* $(\zeta^{(2)}, \zeta_1^{(2)}, \rho^{(2)}, \omega^{(2)}, \sigma_1^{(2)}, \sigma_2^{(2)}, \mu^{(2)}, \delta^{(2)})$ *im the list such that the signatures are valid on the respective messages under the public key* pk *and it holds that there exists a single closed signing session such that* $\mathrm{dlog}_z \zeta^{(1)} = \mathrm{dlog}_{C \cdot g^{rnd}} \zeta_1^{(1)}$ *and* $\mathrm{dlog}_z \zeta^{(2)} = \mathrm{dlog}_{C \cdot g^{rnd}} \zeta_1^{(2)}$*.*

*We define the advantage of the adversary in the game* $\ell$-TTO *as*

$$\mathbf{Adv}^{\ell}\text{-}\mathsf{TTO}_{\mathsf{ACL}}(\mathcal{A}) \coloneqq \Pr[\ell\text{-}\mathsf{TTO}^{\mathcal{A}}_{\mathsf{ACL}} = 1].$$

**Lemma 16 (2-to-1 blinding).** *In the ROM and AGM, if* DLP *is* $(t, \varepsilon)$*-hard in* $\mathbb{G}$*, it holds that for any adversary running in time at most* $t$*, the advantage is at most*

$$\mathbf{Adv}^{\ell\text{-}\mathsf{TTO}}_{\mathsf{ACL}}(\mathcal{A}) \leq 7 \cdot \varepsilon + \frac{20}{p}.$$

*Proof.* Assume there is an algebraic adversary $\mathcal{A}$ that runs and wins experiment $\ell$-TTO. It follows that $\mathcal{A}$ provides two valid signatures $(\zeta^{(1)}, \zeta_1^{(1)}, \rho^{(1)}, \omega^{(1)}, \sigma_1^{(1)}, \sigma_2^{(1)}, \mu^{(1)}, \delta^{(1)})$ and $(\zeta^{(2)}, \zeta_1^{(2)}, \rho^{(2)}, \omega^{(2)}, \sigma_1^{(2)}, \sigma_2^{(2)}, \mu^{(2)}, \delta^{(2)})$ satisfying the equalities $\mathrm{dlog}_z \zeta^{(1)} = \mathrm{dlog}_{C \cdot g^{\mathrm{rnd}}} \zeta_1^{(1)}$ and $\mathrm{dlog}_z \zeta^{(2)} = \mathrm{dlog}_{C \cdot g^{\mathrm{rnd}}} \zeta_1^{(2)}$. Using the same game hop from the proof of Lemma 15 ($\mathbf{G}_2$-$\mathbf{G}_{11}$), we rule out the abort events of these Games. Consequently, Corollaries 3, 4, and 5 hold, and $\mathcal{R}_0$ can efficiently compute $\mathrm{dlog}_g C$ for each registered commitment $C$, $\mathrm{dlog}_z \zeta$, and $\mathrm{dlog}_{z1,\mathsf{sid}} \zeta_1$ of

both hash queries linked to the signatures $\mathcal{A}$ outputs, for some signing sessions $\mathsf{sid}$.

When $\mathcal{R}$ is executed with game $\ell$-$\mathsf{RT}$, it takes $\mathsf{pk} = (h, y, z)$ as input and gets access to the oracles $\mathcal{P}_1, \mathcal{P}_2, \mathcal{V}_1$, and $\mathcal{V}_2$. Then, it invokes $\mathcal{A}$ with input $\mathsf{pk}$ and simulates Game $\ell$-$\mathsf{TTO}$ for $\mathcal{A}$ by providing access to the following oracles:

- $\mathcal{RG}_{\mathsf{S}}(C, \pi)$. If $\pi$ is a valid proof of knowledge of an opening for $C$ and $C \notin$ **registered**, store $C$ in **registered** and output 1, 0 otherwise.
- $\mathsf{S1}(C)$. Output $\perp$ if $C \notin$ **registered**. Query the challenger's oracle $(\mathsf{sid}, a_{\mathsf{sid}}, b_{1,\mathsf{sid}}, b_{2,\mathsf{sid}}, \mathrm{rnd}_{\mathsf{sid}}) \leftarrow_{\$} \mathcal{P}_1(\perp)$, store $\mathsf{sid}$ in $\mathsf{open}$, compute $\mathrm{rnd}'_{\mathsf{sid}} \coloneqq \mathrm{rnd}_{\mathsf{sid}} - \mathrm{dlog}_g C$ (this is possible due to Corollary 3). Output $(\mathsf{sid}, a_{\mathsf{sid}}, b_{1,\mathsf{sid}}, b_{2,\mathsf{sid}}, \mathrm{rnd}'_{\mathsf{sid}})$.
- $\mathsf{S2}(\mathsf{sid}, e_{\mathsf{sid}})$. Output $\perp$ if $\mathsf{sid} \notin \mathsf{open}$. Store $\mathsf{sid}$ in $\mathsf{closed}$, query the challenger's oracle $(c_{\mathsf{sid}}, d_{\mathsf{sid}}, r_{\mathsf{sid}}, s_{1,\mathsf{sid}}, s_{2,\mathsf{sid}}) \leftarrow \mathcal{P}_2(\mathsf{sid}, e_{\mathsf{sid}})$, and output $(c_{\mathsf{sid}}, d_{\mathsf{sid}}, r_{\mathsf{sid}}, s_{1,\mathsf{sid}}, s_{2,\mathsf{sid}})$.
- $\mathsf{H}_1(Q)$. Via lazy sampling.
- $\mathsf{H}_3(Q)$. If $\mathsf{T}_3[Q] \neq \perp$, output $\mathsf{T}_3[Q]$. Otherwise, parse $(\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta, m)$, compute $\gamma \coloneqq \mathrm{dlog}_z \zeta$ using the representation of $\zeta$ if possible (if this query is used later to generate a signature, this is possible due to Corollary 4), otherwise respond to the query with a random value. Compute $z'_1 \coloneqq \zeta_1^{1/\gamma}$, and find a signing session $\mathsf{sid}$ such that $z_{1,\mathsf{sid}} = z'_1$ and set $\mathsf{rnd}^* = \mathrm{rnd}_{\mathsf{sid}}$ (if this query is used later to generate a signature, this is possible due to Corollary 5). If no such signing session exists, respond with a uniformly random value. Next, set $A \coloneqq \alpha, B_1 \coloneqq \beta_1^{1/\gamma}$, $B_2 \coloneqq \beta_2^{1/\gamma}$, make a query $(\mathsf{vid}, \varepsilon_{\mathsf{vid}}) \leftarrow_{\$} \mathcal{V}_1(\mathcal{A}, B_1, B_2, \mathsf{rnd}^*)$. Finally, set $\mathsf{T}_{\mathcal{V}}[Q] \coloneqq \mathsf{vid}$, $\mathsf{T}_3[Q] \coloneqq \varepsilon$, and output $\varepsilon$.
- $\mathsf{H}_{\mathsf{pp}}(Q)$. Via lazy sampling.

When $\mathcal{A}$ outputs a signature $(\zeta, \zeta_1, \omega, \rho, \sigma_1, \sigma_2, \delta, \mu)$ satisfying the condition $\mathrm{dlog}_z \zeta \neq \mathrm{dlog}_{z_{\mathsf{sid}1}} \zeta_1$ for all signing sessions $\mathsf{sid}$, $\mathcal{R}$ retrieves $\mathsf{vid} \leftarrow \mathsf{T}_3[(\zeta, \zeta_1, g^\rho \cdot y^\omega, g^{\sigma_1} \cdot \zeta_1^\delta, h^{\sigma_2} \cdot \zeta_2^\delta, z^\mu \cdot \zeta^\delta)]$ for $\zeta_2 \coloneqq \zeta/\zeta_1$. If $\mathsf{vid} = \perp$, $\mathcal{R}$ aborts and outputs $\perp$. This abort event does not occur because we know from game $\mathbf{G}_9$ that each signature has a corresponding hash query, and the responses of such queries were generated via $\mathcal{V}$ queries., hence each query is linked to a valid verifier session $\mathsf{vid}$. Otherwise, it closes the session $\mathsf{vid}$ by sending $(\omega, \delta, \rho, \sigma_1/\gamma, \sigma_2/\gamma)$.

*Analysis.* First, we note that $\mathcal{R}$'s simulation for the signer's oracles is perfectly indistinguishable from an honest run of the protocol. To see this, note that the only difference between the prover's oracles of the challenger and the honest signer in $\mathsf{ACL}$ is the way $z_1$ is computed. More specifically, $z_1 = g^{\mathrm{rnd}}$ in $\mathsf{ID}_{\mathsf{ACL}}$ while $z_1 = g^{\mathrm{rnd}} \cdot C$ in $\mathsf{ACL}$. To avoid this inconsistency, $\mathcal{R}$ forwards an updated $\mathrm{rnd}' = \mathrm{rnd} - \mathrm{dlog}_g C$ to $\mathcal{A}$ ensuring that $g^{\mathrm{rnd}'} \cdot C = g^{\mathrm{rnd}} \cdot C^{-1} \cdot C = g^{\mathrm{rnd}}$. The resulting $z_1$ (if computed honestly) at the adversary side would be identical to $z_1$ at the prover side, hence forwarding the rest of the values from the prover to $\mathcal{A}$ would preserve the correctness and ensure a perfect simulation.

Second, we note that, if the reduction $\mathcal{R}$ does not abort and $\mathcal{A}$ outputs a valid signature allowing it to win the $\mathsf{RB}$ lemma, $\mathcal{R}$ would use this signature to close a verifier session successfully and satisfying the condition required to win the $\ell$-$\mathsf{RT}$ game. Once $\mathcal{A}$ outputs a valid signature $(\zeta, \zeta_1, \omega, \rho, \sigma_1, \sigma_2, \delta, \mu)$ satisfying the condition $\mathrm{dlog}_z \zeta \neq \mathrm{dlog}_{z_{\mathsf{sid}1}} \zeta_1$ for all signing sessions $\mathsf{sid}$, $\mathcal{R}$ searches for the verifier session $\mathsf{vid}$ that was opened to generate the challenge $\varepsilon$ for this signature by searching $\mathsf{T}_{\mathcal{V}}[(\zeta, \zeta_1, g^{\rho} \cdot y^{\omega}, g^{\sigma_1} \cdot \zeta_1^{\delta}, h^{\sigma_2} \cdot \zeta_2^{\delta}, z^{\mu} \cdot \zeta^{\delta}, m)]$. Then, it sets $\omega' \coloneqq \omega, \delta' \coloneqq \delta, \rho' \coloneqq \rho, \sigma_1' \coloneqq \sigma/\gamma, \sigma_2' \coloneqq \sigma 2/\gamma)$ and closes the session $\mathsf{vid}$ by sending $(\mathsf{vid}, \omega', \rho', \delta', \sigma_1', \sigma_2')$ in a $\mathcal{V}_2$ query. The verifier outputs 1 iff the values $(A, B_1, B_2, \mathsf{rnd}^*)$ and $(\omega', \rho', \delta', \sigma_1', \sigma_2')$ satisfy the equalities $\varepsilon = \omega + \delta$, $\alpha' = g^{\rho'} \cdot y^{\omega'}, B_1 = g^{\sigma_1'} \cdot z_1^{\delta'}, B_2 = h^{\sigma_2'} \cdot z_2^{\delta'}$, where $z_1 = g^{\mathsf{rnd}^*}$ and $z_2 \coloneqq z_1/z$. Indeed, these equalities hold because

- $\omega' + \delta' = \varepsilon$ follows directly from the validity of the signature since $\omega = \omega'$, $\delta = \delta'$, and $\varepsilon = \mathsf{H}_3(\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta, m)$,
- $\alpha' = g^{\rho'} \cdot y^{\omega'}$ follows directly from the validity of the signature since $\alpha = \alpha'$, $\rho = \rho'$, and $\omega = \omega'$.
- Since the signature is valid, it holds that $\beta_1 = g^{\sigma_1} \cdot \zeta_1^{\delta} \Rightarrow \beta_1^{1/\gamma} = (g^{\sigma_1} \cdot \zeta_1^{\delta})^{1/\gamma} \Rightarrow \beta_1^{1/\gamma} = g^{\sigma_1/\gamma} \cdot g^{\mathsf{rnd}^* \cdot \delta} \Rightarrow B_1 = g^{\sigma_1'} \cdot z_1^{\delta'}$.
- Per the validity of the signature, we have $\beta_2 = h^{\sigma_2} \cdot \zeta_2^{\delta} \Rightarrow \beta_2^{1/\gamma} = (h^{\sigma_2} \cdot \zeta_2^{\delta})^{1/\gamma} \Rightarrow \beta_2^{1/\gamma} = h^{\sigma_2/\gamma} \cdot (z/g^{\mathsf{rnd}^*})^{\delta} \Rightarrow \beta_2^{1/\gamma} = h^{\sigma_2/\gamma} \cdot (z/z_1)^{\delta} \Rightarrow B_2 = h^{\sigma_2'} \cdot z_2^{\delta'}$.

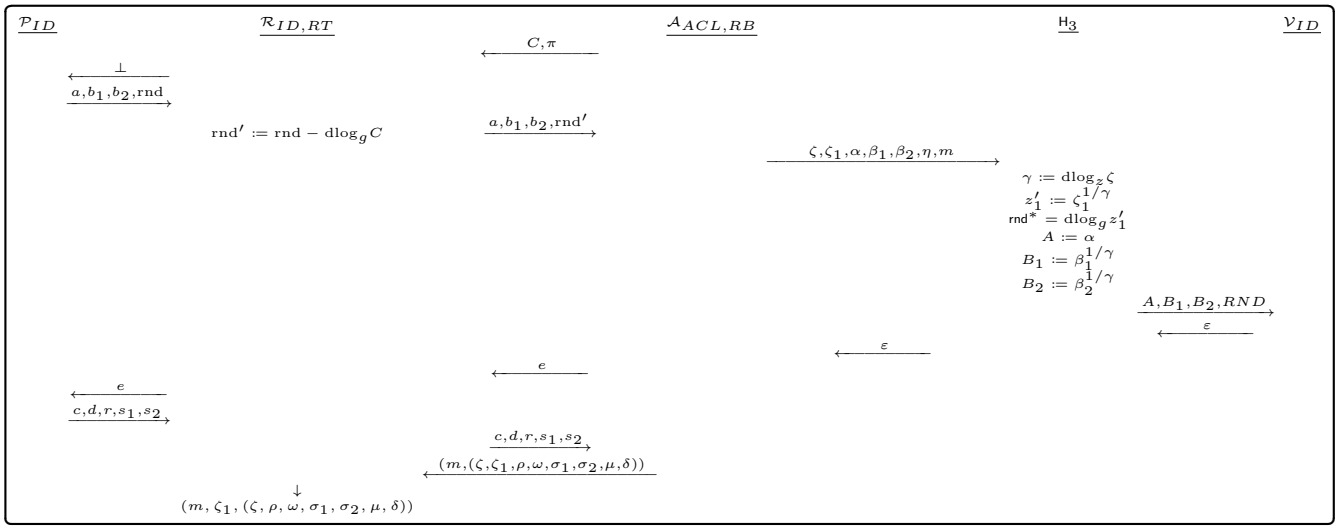


Fig. 6: A reduction from RB of $\mathsf{ACL}$to the RT of $\mathsf{ID}_{\mathsf{ACL}}$.

# I Deferred Figures

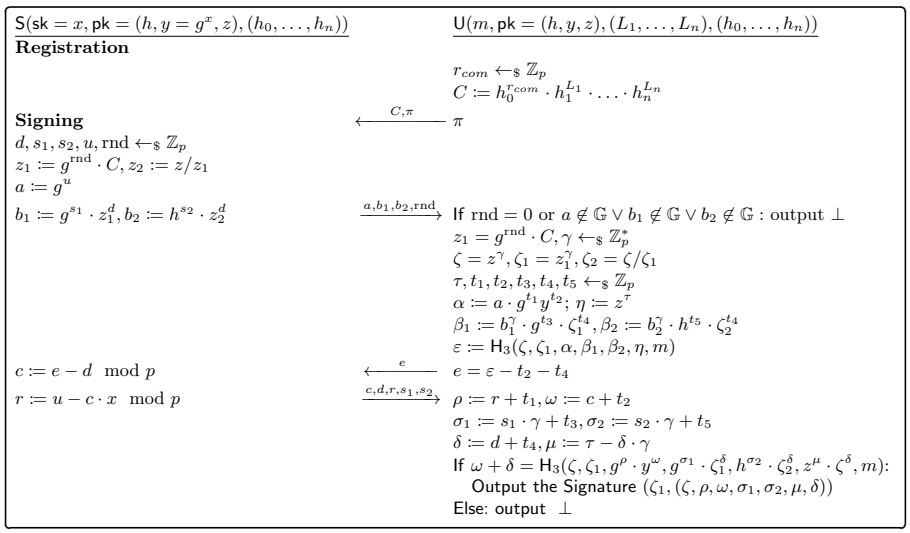


Fig. 7: The ACL scheme [7] depicted as an interactive protocol. The check $\omega+\delta = \mathsf{H}_3(\zeta, \zeta_1, g^\rho \cdot y^\omega, g^{\sigma_1} \cdot \zeta_1^\delta, h^{\sigma_2} \cdot \zeta_2^\delta, z^\mu \cdot \zeta^\delta, m)$ also constitutes the verification equation for a resulting signature.

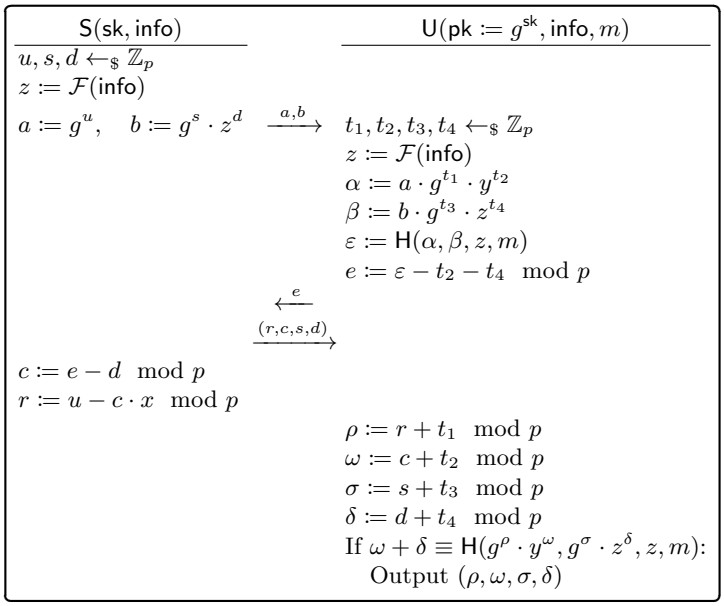


Fig. 8: The Abe-Okamoto (partially) blind signature scheme [3].