Concurrent Security of Anonymous Credentials Light, Revisited

Julia Kastner
Department of Computer Science,
ETH Zurich
Zurich, Switzerland
julia.kastner@inf.ethz.ch

Julian Loss
CISPA Helmholtz Center for
Information Security
Saarbrucken, Germany
loss@cispa.de

Omar Renawi
CISPA Helmholtz Center for
Information Security
Saarland University
Saarbrucken, Germany
omar.renawi@cispa.de

ABSTRACT

We revisit the concurrent security guarantees of the well-known Anonymous Credentials Light (ACL) scheme (Baldimtsi and Lysyanskaya, CCS'13). This scheme was originally proven secure when executed sequentially, and its concurrent security was left as an open problem. A later work of Benhamouda et al. (EUROCRYPT'21) gave an efficient attack on ACL when executed concurrently, seemingly resolving this question once and for all.

In this work, we point out a subtle flaw in the attack of Benhamouda et al. on ACL and show, in spite of popular opinion, that it *can be proven concurrently secure*. Our modular proof in the algebraic group model uses an ID scheme as an intermediate step and leads to a major simplification of the complex security argument for Abe's Blind Signature scheme by Kastner et al. (PKC'22).

CCS CONCEPTS

• **Security and privacy → Digital signatures**; **Digital signatures**; • **Theory of computation → Cryptographic protocols**.

KEYWORDS

Blind Signatures, Anonymous Credentials

1 INTRODUCTION

Blind signatures [21, 22, 51] are a fundamental cryptographic primitive which lies at the heart of privacy-preserving applications such as anonymous credentials [5, 7, 13, 15, 16, 18, 20] and eCash [21]. Informally, a blind signature constitutes an interactive protocol between a signer S and a user U, with the goal of S signing messages of U's choice. The crucial property of a blind signature is its *blindness*: S should obtain no information about U's message m that it is signing. Conversely, security from S's perspective states that it should be infeasible for U to sign messages without the help of S. Fuelled, in part, by developments in the blockchain space [40], blind signatures have recently seen renewed interest [19, 25, 33, 38, 39, 43, 55].

In this work, we revisit the security of the Anonymous Credentials Light (ACL) blind signature scheme due to Baldimtsi and Lysyanskaya [5]. Their simple and elegant construction is derived from Abe's blind signature scheme and avoids computationally heavy zero-knowledge proofs and pairing operations. As such, ACL can be turned into a very efficient one-show credential that allows users to embed and later show arbitrary attributes, e.g., age or gender. Credentials consist of only 9 group elements and can be issued and verified using roughly a dozen group operations. One issue that has limited the practicality of their scheme is the fact that their scheme is only proven secure when signing sessions are executed *sequentially*. This is not a theoretical issue: as shown in a more recent work of Benhamouda et al. [12], there exists an efficient attack on the ACL scheme if even a logarithmic number of sessions are run *concurrently*. It would seem that this leaves no room for further questions regarding ACL's security guarantees. To the contrary, however, we show that the provable security properties of ACL are *all but resolved:*

- We begin by pointing out a subtle flaw in the work of Benhamouda et al. which renders their attack inefficient. (We stress that this does not affect the correctness of their attacks on various other schemes.)
- Motivated by these findings, we give the first concurrent security proof for the ACL scheme in the algebraic group model (AGM) [32]. In order to tame the complexity of our analysis, we follow a modular approach that explains the ACL scheme as being derived from an appropriate identification scheme. Our techniques also lead to a greatly simplified proof of Abe's scheme, which was recently proven secure by Kastner et al. [42].

1.1 Our Contribution

We now elaborate on our contributions in more detail. We begin with a recap of ACL, followed by a detailed description of the issue with the ROS-style attack on ACL. We conclude with a brief overview of our techniques to prove ACL concurrently secure.

ACL*: A Recap.* We begin by giving a brief overview of the ACL scheme, which is depicted in Figure 1. We follow the notation of Baldimtsi and Lysyanskaya and describe their scheme as a blind signature protocol that allows the User to embed attributes into the resulting signature. To keep the attributes private from the signer during signing, the scheme assumes a common uniform random string (or, alternatively, a random oracle) that contains the values $h_0, \ldots, h_n$. This ensures that the attributes can later be shown without linking to a particular session.

The scheme begins with a registration phase during which the user forms a commitment C to its attributes $L_1, \ldots, L_n$ via a generalized Pedersen commitment [47] over some group $\mathbb{G}$ of prime order q. It also forms a proof π of knowledge of these attributes and sends C, π to the Signer. The Signer then verifies these values and stops the issuing process in case they are not well-formed.

Signing then consists of a three-move protocol similar to Abe's blind signature scheme [2]. The scheme proves knowledge of a discrete logarithm to one of two public keys y or z. This is achieved via an OR-style protocol, where two proofs of discrete logarithm are run in parallel. In the first step, the Signer commits to random

$\mathbb{Z}_q$ values d, s_1, s_2, u, and rnd via $z_1 = g^{\text{rnd}} \cdot C, a = g^u$ and $b_1 = g^{s_1} \cdot z_1^d, b_2 = h^{s_2} \cdot z_2^d$, where $z_2 = z/z_1$.

In the second step, the user verifies and blinds these commitments into $\zeta, \zeta_1, \zeta_2, \alpha, \beta_1, \beta_2$ and hashes them into a challenge ε together with its message m. To ensure the unforgeability of the signature, the user also needs to provide a proof of knowledge of the blindings that it used via the value η. From the way that z_1 is formed, the user can convince itself that the signer does not know the discrete logarithm of z_1. This turns out to be crucial toward the blinding guarantees of the scheme. Namely, the blinded values $\zeta := z^\gamma, \zeta_1 := z_1^\gamma$ are computed by raising z and z_1 to the *same* random blinding $\gamma \in \mathbb{Z}_q$. Hence, if the signer knew the discrete logarithm of z_1 (to base z), it could later link the resulting signature to the session during which it was issued by finding the signature satisfying $\zeta^{\log_z}(z_1) = \zeta_1$. The User derives a blinded challenge e that it sends to the signer.

In the third step, the Signer splits the challenge e into two subchallenges c and d (i.e., one per side of the OR protocol), where d was sampled in the first step. It replies with a 'real' answer $u - cx$ and a simulated answer s_1, s_2 which, in the view of the user, proves knowledge of either $x = \log_g(y)$ or $w = \log_h(z)$. The user can suitably unblind these values to derive its final signature.

Observe that the user knows γ and rnd. Hence, it can efficiently prove knowledge of its attributes with regards to the blinded commitment ζ_1 using its signature.

1.1.1 ROS Attack against ACL. We now revisit the ROS-Style Attack [52, 56] of Benhamouda et al. on ACL. In this attack, the attacker $\mathcal{A}$ breaks one-more unforgeability by outputting $\ell + 1$ signatures, in spite of interacting with the signer in only ℓ many signing sessions.

For simplicity, we assume $\gamma = 1$, hence $\zeta = z$ and $\zeta_1 = z_1$. We further assume that $t_i = 0$ for $i = 1, \ldots, 5$. The attacker $\mathcal{A}$ outputs $\ell + 1$ forgeries on the commitment C as follows:

(1) $\mathcal{A}$ starts ℓ concurrent sessions with the signer and finishes the Registration Phase of these executions following the protocol.
(2) During the Signing Phase of session i, $\mathcal{A}$ receives the first signer message $(a_i, b_{1,i}, b_{2,i}, \text{rnd}_i)$ (denoted in [12] as A_i, $A'_{i,1}, A'_{i,2}, d_i$) for $i \in [\ell]$. Using these commitments, it then computes $e_i^{(b)} = 2^{-b} H(z, Cg^{\text{rnd}_i}, a_i^{2^b}, b_{1,i}^{2^b}, b_{2,i}^{2^b}, z^{\tau_i}, m_i)$ (denoted as c_i^b in [12]) for all $i \in [\ell], b \in \{0, 1\}$ with random messages $m_i, \tau_i \leftarrow_\$ \mathbb{Z}_q$. If $e_{i^*}^{(0)} = e_{i^*}^{(1)}$ for some i^*, the algorithm closes all sessions using the $e_i^{(0)}$ challenges. It outputs the message-signature pairs $(m_i, (z, Cg^{\text{rnd}_i}, c_i, d_i, r_i, s_{1,i}, s_{2,i}, \mu_i = \tau_i - d_i))_{i=1}^{\ell}$ and $(m_{i^*}, (z, Cg^{\text{rnd}_{i^*}}, 2c_{i^*}, 2d_{i^*}, 2r_{i^*}, 2s_{1,i^*}, 2s_{2,i^*}, \mu_{i^*} = \tau_{i^*} - d_{i^*}))$. If $e_i^{(0)} \neq e_i^{(1)}$ for all i, it then defines the polynomial $\rho(x_1, \ldots, x_\ell) := \sum_{i \in [\ell]} 2^{i-1}(x_i - e_i^{(0)})/(e_i^{(1)} - e_i^{(0)})$. This is a linear function that can be evaluated in the exponent. By slight abuse of notation, we write for a linear function $\rho(x_1, \ldots, x_\ell) = \rho_0 + \sum_{i=1}^{\ell} \rho_i x_i$ and group elements $\alpha_1, \ldots \alpha_\ell$ the evaluation in the exponent as follows $\rho(\alpha_1, \ldots, \alpha_\ell) := g^{\rho_0} \prod_{i=1}^{\ell} \alpha_i^{\rho_i}$.
(3) Next, it computes the values $a_{\ell+1}, b_{1,\ell+1}$, and $b_{\ell+1}$ as

$$a_{\ell+1} := \rho(a_1, \ldots, a_\ell) y^{\rho_0},$$
$$b_{1,\ell+1} := \rho(b_{1,1}, \ldots, b_{1,\ell}) C^{\rho_0}, b_{2,\ell+1} := \rho(b_{2,1}, \ldots, b_{2,\ell})(z/C)^{\rho_0}. \tag{1}$$

(4) $\mathcal{A}$ then sends the challenges $e_i^{b_i}$ to the signer according to the bit representation $\sum_{i=1}^{\ell} b_i 2^{i-1} = H(z, C, a_{\ell+1}, b_{1,\ell+1}, b_{2,\ell+2}, z^{\tau_{\ell+1}}, m_{\ell+1}) - \rho_0$ and receives the responses $(c_i, r_i, d_i, s_{1,i}, s_{2,i})$ for $i \in [\ell]$.
(5) Finally, it computes:

$$c_{\ell+1} := \rho(\mathbf{c}) = \sum_{i=1}^{\ell} \rho_i c_i + \rho_0,\ d_{\ell+1} := \rho(\mathbf{d}) = \sum_{i=1}^{\ell} \rho_i d_i + \rho_0,$$
$$r_{\ell+1} := \rho(\mathbf{r}) = \sum_{i=1}^{\ell} \rho_i r_i + \rho_0,$$
$$s_{1,\ell+1} := \rho(\mathbf{s_1} + \mathbf{d} \circ \mathbf{rnd}) = \sum_{i=1}^{\ell} \rho_i (s_{1,i} + \text{rnd}_i d_i) + \rho_0,$$
$$s_{2,\ell+1} := \rho(\mathbf{s_2} - \mathbf{d} \circ \mathbf{rnd}) = \sum_{i=1}^{\ell} \rho_i (s_{2,i} - \text{rnd}_i d_i) + \rho_0,$$

where $\mathbf{d} \circ \mathbf{rnd} = (d_1 \text{rnd}_1, \ldots, d_\ell \text{rnd}_\ell)$.

For $i = 1, \ldots, \ell$, the adversary generates the forgery as $(m_i, \sigma_i) := (m_i, (z, Cg^{\text{rnd}_i}, 2^{b_i} c_i, 2^{b_i} d_i, 2^{b_i} r_i, 2^{b_i} s_{1,i}, 2^{b_i} s_{2,i}, \mu_i = \tau_i - 2^{b_i} d_i))$. In this attack, the User does not blind any of these values, and hence, these are valid signatures since

$$\begin{aligned}
2^{b_i} c_i + 2^{b_i} d_i = 2^{b_i} e_i^{(b_i)} &= \mathrm{H}(z, g^{\text{rnd}_i} C, a_i^{2^{b_i}}, b_{1,i}^{2^{b_i}}, b_{2,i}^{2^{b_i}}, z^{\tau_i}, m_i), \\
g^{2^{b_i} r_i} y^{2^{b_i} c_i} &= a_i^{2^{b_i}}, \\
g^{2^{b_i} s_{1,i}} (g^{\text{rnd}_i} C)^{2^{b_i} d_i} &= b_{1,i}^{2^{b_i}}, \\
h^{2^{b_i} s_{2,i}} (z/(g^{\text{rnd}_i} C))^{2^{b_i} d_i} &= b_{2,i}^{2^{b_i}}, \\
z^{\tau_i - 2^{b_i} d_i} z^{2^{b_i} d_i} &= z^{\tau_i}.
\end{aligned}$$

In addition, the adversary computes $(m_{\ell+1}, \sigma_{\ell+1})$, where $\sigma_{\ell+1} := (z, C, c_{\ell+1}, d_{\ell+1}, r_{\ell+1}, s_{1,\ell+1}, s_{2,\ell+1}, \mu_{\ell+1} = \tau_{\ell+1} - d_{\ell+1})$.

We give the verification equations from [12] in our notation:

$$\begin{aligned}
g^{r_{\ell+1}} y^{c_{\ell+1}} &= \prod_{i=1}^{\ell} (g^{r_i} y^{c_i})^{\rho_i} (gy)^{\rho_0} = a_{\ell+1}, \\
g^{s_{1,\ell+1}} C^{d_{\ell+1}} &= \prod_{i=1}^{\ell} (g^{s_{1,i}} (Cg^{\text{rnd}_i})^{d_i})^{\rho_i} (gC)^{\rho_0} = b_{1,\ell+1}, \\
g^{s_{2,\ell+1}} (z/C)^{d_{\ell+1}} &= \prod_{i=1}^{\ell} \left(g^{s_{2,i}} \left(z / \left(Cg^{\text{rnd}_i} \right) \right)^{d_i} \right)^{\rho_i} (gz/C)^{\rho_0} \qquad (2) \\
&\neq \prod_{i=1}^{\ell} \left(h^{s_{2,i}} \left(z / \left(Cg^{\text{rnd}_i} \right) \right)^{d_i} \right)^{\rho_i} (gz/C)^{\rho_0} \qquad (3) \\
&= \prod_{i=1}^{\ell} b_{2,i}^{\rho_i} (gz/C)^{\rho_0} = \rho(b_{2,1}, \ldots, b_\ell) = b_{2,\ell+1}, \\
c_{\ell+1} + d_{\ell+1} &= \rho(c_0 + d_0, \ldots, c_\ell + d_\ell) + \rho_0 \\
&= H(z, C, a_{\ell+1}, b_{1,\ell+1}, b_{2,\ell+1}, m_{\ell+1}).
\end{aligned}$$

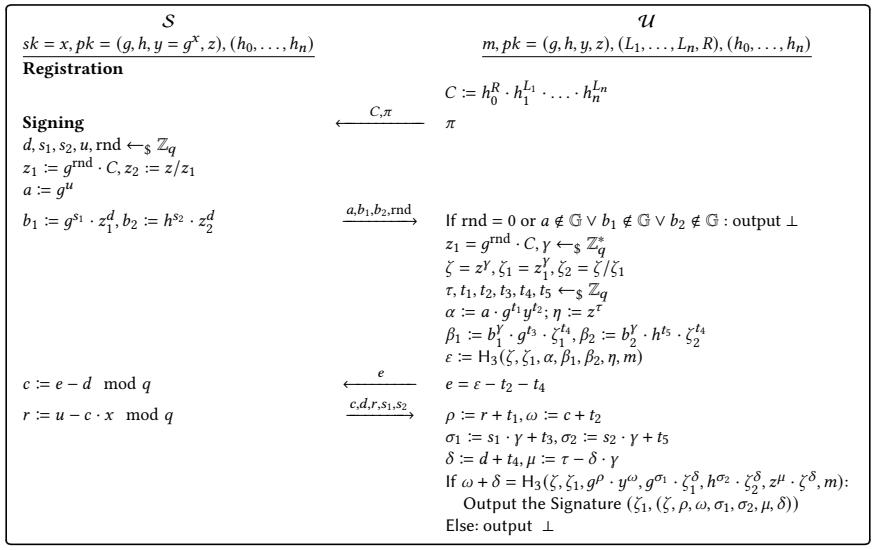

Figure 1: The ACL scheme [5] depicted as an interactive protocol. The check $\omega + \delta = \mathsf{H}_3(\zeta, \zeta_1, g^\rho \cdot y^\omega, g^{\sigma_1} \cdot \zeta_1^\delta, h^{\sigma_2} \cdot \zeta_2^\delta, z^\mu \cdot \zeta^\delta, m)$ also constitutes the verification equation for a resulting signature.

The two sides $g^{s_{2,\ell+1}}(z/C)^{d_{\ell+1}}$ and $b_{2,\ell+1}$ of the third verification equality are translated as written above directly from [12] As indicated, the problem lies with (3), which shows that the attack, as written, does not produce a verifying signature.

However, we point out that this is not even the correct verification check in the first place. Namely, the actual verification procedure of ACL for signature $\sigma_{\ell+1}$ checks whether $c_{\ell+1} + d_{\ell+1} = \mathsf{H}(z, C, g^{r_{\ell+1}} y^{c_{\ell+1}}, g^{s_{1,\ell+1}} z_{1,\ell+1}^{d_{\ell+1}}, h^{s_{2,\ell+1}}(z/C)^{d_{\ell+1}}, z^{\tau_{\ell+1} - d_{\ell+1}} z^{d_{\ell+1}}, m_{\ell+1})$. In particular, the third to last component of H checks that $h^{s_{2,\ell+1}} \cdot (z/C)^{d_{\ell+1}} = b_{2,\ell+1}$, whereas the left-hand side of the verification equation, as written, is of the form $g^{s_{2,\ell+1}}(z/C)^{d_{\ell+1}}$.

Why the Attack Cannot be Fixed. We stress that this is not simply a fixable typo. (Of course, this is clear from the fact that we give a proof to the contrary, see below). It might be tempting to instead define the evaluation of polynomial ρ on group elements $\alpha_1, \ldots, \alpha_\ell$ as $\rho(\alpha_1, \ldots, \alpha_\ell) := h^{\rho_0} \prod_{i=1}^{\ell} \alpha_i^{\rho_i}$. Then, in Equation 1 one would start from $h^{s_{2,\ell+1}} \cdot (z/C)^{d_{\ell+1}}$ rather than from the incorrect $g^{s_{2,\ell+1}}(z/C)^{d_{\ell+1}} = b_{2,\ell+1}$. However, this also fails. First, it would falsify two of the remaining three verification equations. However, this might be fixable by defining two distinct ways of evaluating ρ on group elements, i.e., one using g and one using h as the base for ρ_0. However, even this approach does not see any success. Note that Equation 2 is true *precisely because* the left hand side of the equation uses $g^{s_{2,\ell+1}}(z/C)^{d_{\ell+1}}$. Indeed, if it didn't, then one could not split $g^{s_{2,\ell+1}}$ into $\prod_{i=1}^{\ell} g^{s_{2,i}}/g^{\text{rnd}_i \cdot \rho_i}$. However, this is critically needed, because, in the next steps, we wish to substitute the resulting expression with $\prod_{i=1}^{\ell} b_{2,i}^{\rho_i} g^{\rho_0}$, where for all $i \in [\ell]$ $b_{2,i} = h^{s_{2,i}}(z/(g^{\text{rnd}_i} C))^{d_i}$. Hence, it is unclear how to carry out the attack efficiently from the description of Benhamhouda et al.

Intuition for ROS-Resistance of ACL. We give a brief intuition for why ACL is immune to the ROS attack. Recall that in Abe's blind signature, the signer uses an individual session key z_1 (where z_1 comes out of a random oracle) for each session, and a signature contains a proof that the blinding of this session key happened honestly. Therefore, an attacker cannot "mix and match" sessions in the way that is required by ROS. In the ACL scheme, this session key is replaced by a commitment to the user's attributes, offset by g^{rnd} where the signer chooses a fresh rnd in each round. However, as these offsets are w.r.t. g and one of the proofs of DL-knowledge in the signature is w.r.t. the public key part h, it is still not possible to combine sessions in a ROS-like way.

1.1.2 *Concurrent Security for* ACL.

We now give an overview of our techniques for proving the concurrent security of the ACL scheme.

Challenges with Adapting Kastner et al.'s Proof. At a high level, our proofs follow the techniques of Kastner et al., who provided a proof for the concurrent security of Abe's scheme in the AGM. However, there are several key differences between Abe's scheme and ACL, that make things significantly more complicated when proving the latter secure. The most notable difference lies with the fact that in ACL, the signer picks rnd uniformly at random from $\mathbb{Z}_q$ and computes $z_1 = g^{\text{rnd}} \cdot C$. This stands in stark contrast to Abe's scheme, which sets $z_1 = \mathsf{H}(\text{rnd})$, where H is a random oracle. As a result, the user in ACL knows a discrete logarithm relation between all values of z_1 that are created throughout all its interactions with the signer. Indeed, it is precisely this knowledge that Benhamouda et al. (incorrectly) leveraged for their attack.

Therefore, to adapt the proof of Kastner et al., we need to account for several additional cases in terms of how the adversary can form its outputs from its inputs. Unfortunately, this comes on top of an already extremely involved analysis comprised of almost thirty pages of complex case distinctions in Kastner et al.'s proof. Roughly speaking, their analysis consists of a case distinction over

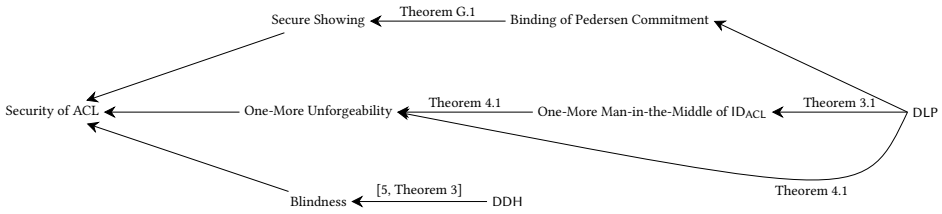

Figure 2: Overview over our theorems and proofs.

all possible ways that an algebraic adversary can combine elements from previous signing sessions into a one-more forgery. At a high level, the crux of their reduction strategy is to define a *preliminary algebraic representation* consisting of the coefficients that the adversary returns together with the hash query (modeled as a random oracle query) corresponding to one of its forgeries. The term 'preliminary' refers to the fact that the adversary could use a different representation for some of the corresponding group elements when it returns its forgery. In this case, however, one obtains an equality from which one can solve a suitably embedded discrete logarithm challenge. If, on the other hand, the adversary keeps the representation the same, then with high probability, it will lead to a non-trivial inequality, because it could not have predicted the random oracle's output at the time it made the query. Unfortunately, making this intuition work turns out to be extremely subtle, as some of the dependencies in the scheme allow the adversary to influence these preliminary representations even *after* first returning them.

It is unclear whether a similarly complicated argument would be feasible for ACL or of any use to a reader attempting to verify it. Therefore, to tame our proof's complexity, we seek a more modular security argument that breaks the proof down into multiple digestible steps while simultaneously minimizing some of the redundancy that is present in Kastner et al.'s proof. Our main tool for doing so is to reduce the security of ACL to that of an underlying identification $\mathsf{ID}_{\mathsf{ACL}}$ scheme with a suitable structure.

A Modular Proof for ACL. To this end, we first deconstruct ACL to find the ID scheme $\mathsf{ID}_{\mathsf{ACL}}$ at its core. In $\mathsf{ID}_{\mathsf{ACL}}$, the Signer is replaced with a prover and analogously, the User is replaced with a verifier. While the prover in $\mathsf{ID}_{\mathsf{ACL}}$ carries out the same steps as the Signer in the ACL scheme, the verifier of $\mathsf{ID}_{\mathsf{ACL}}$ follows a far simpler structure than the User in the ACL scheme. Most notably, the verifier does not apply any blinding and samples the challenge ε directly from $\mathbb{Z}_q$ as opposed to deriving it from a random oracle H. This declutters the scheme considerably and allows us to compactly prove $\mathsf{ID}_{\mathsf{ACL}}$ secure with respect to the notion of *one-more-man-in-the-middle (*OMMIM*)* that was put forth by Hauck, Kiltz, and Loss [38] (also in the context of blind signatures). Informally, this notion ensures that a man-in-the-middle M should not be able to close more than ℓ sessions with the verifier if it closes at most ℓ with the honest prover. The crux of this notion is that M is allowed to interleave sessions with the honest prover and the verifier concurrently. This makes OMMIM a much stronger notion than the standard notion of active security [10, 27], where M is forced to close all (concurrent) interactions with the prover *before* interacting with the verifier in a designated challenge session. On the other hand, OMMIM is a weaker notion than the standard man-in-the-middle (MIM) security notion [8, 36]. Namely, the latter simply asks that M can close a session with the verifier using a fresh transcript (i.e., one that did not come from the honest prover).

We prove OMMIM of $\mathsf{ID}_{\mathsf{ACL}}$ by following a similar proof strategy as Kastner et al. However, we introduce several new tools which allow to group together many of the cases that can occur. This allows us to keep the presentation clear and the complexity of the proof in check.

In a second step, we then reduce the one-more unforgeability of ACL directly to the OMMIM-security of $\mathsf{ID}_{\mathsf{ACL}}$. While this reintroduces the blinding steps on the user's side, our reduction again manages to keep the complexity of the resulting case distinction low. It does so by reducing any attack in which U behaves honestly in all interactions with the signer directly to OMMIM security of $\mathsf{ID}_{\mathsf{ACL}}$. Now, our case distinction only has to ensure that dishonest behaviour can not give an efficient one-more forgery. We achieve the latter by relying, again, on the techniques we developed for proving OMMIM security of $\mathsf{ID}_{\mathsf{ACL}}$. A high-level overview of our proof strategy is depicted in fig. 2.

1.2 Related Work

We give a brief overview of some important related works in the area.

1.2.1 Blind Signatures. Blind Signatures have been the subject of intense study for nearly four decades. After first being proposed by Chaum [21], blindness and one-more unforgeability were first rigorously defined as security properties in works of Juels, Luby, and Ostrovsy [41] and Pointcheval and Stern [49]. Many works have since explored blind signatures from various angles. We have already cited many relevant works in the random oracle model that achieve efficient constructions from well-understood hardness assumptions, some examples being discrete logarithm [4, 51] and factoring and RSA [50] based schemes. Early examples of such schemes are known to be vulnerable to the ROS attack [12, 52, 56] and thus inherently achieve security only for logarithmically many concurrently issued signatures per public key. This problem was addressed in a series of works initiated by Pointecheval [48], and continued by Katz, Loss, and Rosenberg [43], Chairattana-Aiprom et al. [19], and Hanzlik, Loss, and Wagner [37] who give transforms from schemes for logarithmically many signatures to fully secure ones. Schemes from post-quantum secure assumptions have also been proposed in the random oracle model by Hauck et al. [39] and del Pino and Katsumata [25]. It is also possible to achieve efficient

blind signatures from strong interactive assumptions in the random oracle model or standard model directly [9, 14, 31, 45].

Blind signatures have also been studied in other idealized models such as the generic group model [53] and the algebraic group model. Examples of such analyses include the works of Schnorr [52] and Abe and Ohkubo [3] in the GGM and the works of Fuchsbauer, Plouviez, and Suerrin [33], Tessaro and Zhu [55] and Kastner, Loss, and Xu [42]. In the standard model, round-optimal constructions of blind signatures [34, 35] have also been constructed, but are of mostly theoretical interest.

Finally works by Fischlin [29] and Juels et al. [41] give generic constructions of blind signatures from basic primitives such as commitments, zero knowledge, and general two-party computation.

Impossibility of blind signatures under various conditions has also been studied in the literature. Fischlin and Schröder show impossibility of proving the security of two-round blind signatures from a natural class of protocols in the standard model [30]. Impossibility of two-round schemes with unique secret keys has also been considered with respect to the random oracle model by Pass [46] and Baldimtsi and Lysyanskaya [6].

1.2.2 Anonymous Credentials. Anonymous credentials are a key ingredient for privacy-preserving systems in their own right, and many constructions exist in the literature. Typically, one distinguishes two types of credentials: the first type can be shown *multiple times* without being able to link any of these showings to one another. This usually requires some type of zero-knowledge, as it prohibits the owner of the credential from actually revealing it during showing. Examples of multi-show credentials are the constructions of Camenisch and Lysyanskaya [17, 18], the constructions of Belenkiy et al. [7], and the construction of Chase et al. [20]. Recently, some works have also considered decentralized anonymous (multi-show)credentials, i.e., where there is no single authority that issues the credential. Examples are the works of Sonnino et al. [54] and the very recent work of Doerner et al. [26].

The second type of credential can be shown only *a single time*. While this comes at the expense of some functionality, single-show credentials can often be constructed directly from blind signature schemes and can bypass heavy tools such as zero-knowledge. Therefore, single-show credentials are usually far more efficient than multi-show credentials. On the other hand, they offer a light-weight alternative to multi-show credentials which still offers a reasonable degree of anonymity. Examples of one-show credentials are the systems by Brands [15] as well as the ACL system of Baldimtsi and Lysyanskaya, which is the subject of this work. Constructions of anonymous tokens have also been considered in the literature by Dadvison et al. [24] and Kreuter et al. [44], but these constructions do not support the embedding of attributes.

2 PRELIMINARIES

We denote as "$r \leftarrow_{\$} \mathcal{D}$" that a value r is sampled uniformly at random from a distribution $\mathcal{D}$. Unless otherwise stated, we consider probabilistic polynomial-time algorithms and write out $\leftarrow_{\$} \mathsf{A}(x)$ to state that A outputs out on input x. If A is deterministic, we instead write this as out $\leftarrow \mathsf{A}(x)$. We write A^{B} to denote that A has oracle access to algorithm B at runtime. For $n \in \mathbb{Z}^+$, we denote as $[n]$ the numbers in $\{1, .., n\}$.

2.0.1 The Random Oracle Model (ROM).. In the ROM [11], we assume the existence of an idealized hash function H with the following behaviour. Initially, $H[x] = \perp$ for all x in the domain of x. When H is queried on input x, it checks whether $H[x] := \perp$. If so, it samples a uniformly random value y from the codomain of H and sets $H[x] := y$. Then it returns $H[x]$.

2.1 Group-Based Cryptography

We briefly recall some background on groups and idealized models that are relevant to this paper. In this paper, we work with a cyclic group $\mathbb{G}$ of prime order q with known representation and generator g. Thus, we implicitly assume throughout this paper that group parameters $\mathsf{pp} := (\mathbb{G}, g, q)$ have been distributed and are known to all parties.

2.1.1 The Algebraic Group Model (AGM). The AGM [32] is an idealized model which lies in between the generic group model and the standard model, in which all algorithms are considered *algebraic*. An *algebraic algorithm* is defined as follows. Let $\mathbb{G}$ be a cyclic group, and let $\mathcal{A}$ be an algorithm that takes as input the group elements $(x_1, \ldots, x_n)$. We say that $\mathcal{A}$ is algebraic, if for each group element $Z \in \mathbb{G}$ it outputs, it submits a vector $\vec{z} = (z_1, \ldots, z_n) \in \mathbb{Z}_q^n$ with

$$\prod_{i=1}^{n} x_i^{z_i} = Z.$$

When n is large, proofs in the AGM can quickly become very complex. To mitigate this issue, we use *reduced representations* [42] as follows: Assume there exists $\{x'_1, \ldots, x'_m\} \subset \{x_1, \ldots, x_n\}$ such that, for all x_i there exists a representation $\vec{z_i}' = (z'_{i,1}, \ldots, z'_{i,m})$ with

$$x_i = \prod_{j=1}^{m} {x'_j}^{z'_{i,j}}.$$

Given any representation $(z_1, \ldots, z_n)$ to the basis $(x_1, \ldots, x_n)$ of a group element v, we construct a reduced representation $(z^*_1, \ldots, z^*_m)$ to the basis $(x'_1, \ldots, x'_m)$ by computing

$$z^*_j := \sum_{i=1}^{n} z_i \cdot z'_{i,j}$$

for all $j \in [m]$. It is worth noting that if the representation $\{z'_{i,1}, \ldots, z'_{i,m}\}$ is not known to the reduction, it cannot compute the reduced representations efficiently because this would require solving the discrete logarithm problem.

2.1.2 The Discrete Logarithm Assumption. The security of our schemes is based on the standard discrete logarithm problem:

Definition 2.1 (The Discrete Logarithm Problem DLP*).* For a group $\mathbb{G}$ of prime order q, and for an adversary $\mathcal{A}$, we define the DLP game as follows:

- **Setup.** The game samples $x \leftarrow_{\$} \mathbb{Z}_q$, computes $U := g^x$, and runs $\mathcal{A}$ with input U.
- **Output Determination.** When $\mathcal{A}$ terminates and outputs x', the game outputs 1 if $g^{x'} = U$, otherwise it outputs 0.

We define $\mathcal{A}$'s advantage in winning this game as

$$\mathbf{Adv}^{\mathsf{DLP}}(\mathcal{A}) := \Pr[\mathsf{DLP}^{\mathcal{A}} = 1].$$

2.2 Identification Schemes

An identification scheme is an interactive protocol between a prover and a verifier that enables the prover to prove possession of a private key. In this section, we introduce the syntax and security definitions for three-move canonical identification schemes [1, 38].

2.2.1 Definition (Canonical Three-Move Identification Schemes). A *three-move identification scheme* ID is a triple of polynomial-time algorithms ID =$(\mathcal{G}, \mathcal{P}, \mathcal{V})$ with the following properties.

- The *key generation algorithm* $\mathcal{G}$ takes as input public parameter pp, and outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.
- The *prover algorithm* $\mathcal{P}$ is split into two algorithms $\mathcal{P}_1$ and $\mathcal{P}_2$:
  - $\mathcal{P}_1$ takes as input a secret key sk. It returns a commitment R and a prover state st_p.
  - $\mathcal{P}_2$ is deterministic and takes as input a secret key sk, a prover state st_p, a commitment R, and a challenge ε. It returns a response S.
- The *verifier algorithm* $\mathcal{V}$ is split into two algorithms ($\mathcal{V}_1$, $\mathcal{V}_2$):
  - $\mathcal{V}_1$ takes as input a public key pk, and a commitment R. It outputs a challenge ε and a state st_v.
  - $\mathcal{V}_2$ is deterministic and takes in a public key pk, a verifier state st_v, a commitment R, a challenge ε, and a response S. It outputs 1 (Accept) or 0 (Reject).

Definition 2.2 (Perfect Correctness for Identification Schemes). A canonical identification scheme $\mathsf{ID} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ is perfectly correct if the following holds:

$$\Pr\left[b = 1 : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathcal{G}(pp) \\ (R, st_p) \leftarrow_{\$} \mathcal{P}_1(\mathsf{sk}) \\ (\varepsilon, st_v) \leftarrow_{\$} \mathcal{V}_1(\mathsf{pk}, R) \\ S \leftarrow \mathcal{P}_2(\mathsf{sk}, st_p, R, \varepsilon) \\ b \leftarrow \mathcal{V}_2(\mathsf{pk}, st_v, R, \varepsilon, S) \end{array}\right] = 1.$$

*Definition 2.3 (One-More-Man In-The-Middle (*OMMIM*) [38]).* For a positive integer $\ell \in \mathbb{Z}^+$, an identification scheme $\mathsf{ID} := (\mathcal{G}, \mathcal{P}, \mathcal{V})$, and an adversary $\mathcal{A}$, we define the game ℓ-OMMIM as follows:

- **Setup.** Generate a pair of keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathcal{G}(pp)$, and invoke $\mathcal{A}^{\mathcal{P}(\mathsf{sk},\cdot),\mathcal{V}(\cdot,\mathsf{pk})}(\mathsf{pk})$.
- **Online Phase.** $\mathcal{A}$ may query its oracles arbitrarily in an interleaved fashion as long as it completes at most ℓ of those executions with $\mathcal{P}$.
- **Output Determination.** The game outputs 1, if $\mathcal{A}$ successfully completes at least $\ell + 1$ executions with $\mathcal{V}$. Otherwise, the game outputs 0.

Intuitively, the prover answers at most ℓ challenges to close verifier sessions successfully, while $\ell + 1$ verifier sessions must be closed successfully; therefore, $\mathcal{A}$ must close at least one verifier session without the prover's involvement. We define $\mathcal{A}$'s advantage in winning the game ℓ-OMMIM against an identification scheme ID as

$$\mathbf{Adv}^{\ell\text{-OMMIM}}_{\mathsf{ID}}(\mathcal{A}) := \Pr[\ell\text{-OMMIM}^{\mathcal{A}}_{\mathsf{ID}} = 1].$$

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

2.3 Blind Signatures with Attributes

Baldimtsi and Lysyanskaya [5] proposed the notion of *blind signatures with attributes*, an extension for standard blind signatures that allows attaching the user's attributes to the resulting signature via a commitment. We remark that our syntax differs slightly from that of Baldimtsi and Lysyanskaya. In particular, the verifier in our notion only checks for the validity of a signature σ resulting from an interaction between the signer and the user, but not whether it is tied to a valid commitment C to the user's attributes $\vec{L}$. Instead, we define a designated pair of algorithms $\mathcal{SH} = (\mathcal{SH}_U, \mathcal{SH}_V)$ for the user to present (a subset of) its attributes and a proof of knowledge of an opening of the commitment C tying it to σ. This allows us to split and simplify our security definitions when compared to Baldimtsi and Lysyanskaya.

More precisely, our first notion, captured in definition 2.7, closely mirrors the standard (strong) one-more unforgeability notion for blind signatures and deals with adversaries that output more signatures than they requested from the signer. Our second notion, captured in definition 2.8, deals with adversaries that try to show a signature (i.e., their anonymous credential) alongside attributes that they didn't previously commit to during registration. We also capture adversaries that attempt to 'reuse' committed attributes with signatures that didn't originally contain those attributes.

For the latter security property, other definitions may be possible. For example, we require the adversary to reveal its complete attribute vectors. A definition in which the adversary reveals only parts of them would also be possible, although more complex. However, we believe that this definition captures the corresponding property in [5] most closely.

In the following, we describe our syntax of blind signatures with attributes. A blind signature with attributes has three types of interaction: The first is registration of users with signers, where a user registers a commitment of his attributes and proves knowledge of an opening. The second type of interaction is the issuing of signatures, where the user and signer interact so that the user will obtain a signature from the signer on a message of its choice that is linked to the user's attributes. The last type of interaction is verification and showing of attributes and signatures. This is split into a *signature verification algorithm* that checks if the signature is valid, and a pair of *show algorithms*. The latter allows the user to reveal a subset of its attributes to a verifier who can check that they link to the signature.

In the following we call a vector $\vec{L}$ of length n a *partial vector* if some of its entries are a special empty symbol $\perp$.

Definition 2.4. A *blind signature with attributes* is a tuple of algorithms $\mathsf{BSA} := (\mathcal{G}, \mathcal{R}, \mathcal{S}, \mathcal{U}, \mathcal{V}, \mathcal{SH})$, with the following properties.

- The *key generation algorithm* $\mathcal{G}$ takes as input public parameter pp and outputs a pair of keys $(\mathsf{pk}, \mathsf{sk})$.
- The *registration protocol* $\mathcal{R}$ between the signer and the user consists of two algorithms $\mathcal{R}_{\mathcal{U}}, \mathcal{R}_{\mathcal{S}}$:
  - $\mathcal{R}_{\mathcal{U}}$ takes as input a public key pk, and a vector of attributes $\overrightarrow{L}$. It outputs a commitment C to $\overrightarrow{L}$, a registration state st_r, and a proof π.
  - $\mathcal{R}_{\mathcal{S}}$ is deterministic and takes as input a secret key sk, a commitment C, and a proof π. It outputs 1 (Accept) or 0 (Reject).

- The *signer algorithm* $\mathcal{S}$ is split into two algorithms $\mathcal{S}_1$ and $\mathcal{S}_2$:
  - $\mathcal{S}_1$ takes as input a secret key sk and a commitment C. It returns a commitment R and a signer state st_s.
  - $\mathcal{S}_2$ is deterministic and takes as input a secret key sk, a singer state st_s, a commitment R, and a challenge e. It returns a response S.
- The *user algorithm* $\mathcal{U}$ is split into two algorithms $\mathcal{U}_1$ and $\mathcal{U}_2$:
  - $\mathcal{U}_1$ takes as input a public key pk, commitments R and C, a message m, and a registration state st_r. It returns a challenge e and a user state st_u.
  - $\mathcal{U}_2$ is deterministic and takes as input a public key pk, a user state st_u, a commitment R, a challenge e, a response S, and a message m. It returns a signature $\hat{\sigma}$ and an opening r.
- The *show algorithm* $\mathcal{SH}$ is split into two algorithms $\mathcal{SH}_U$ and $\mathcal{SH}_V$:
  - The *show generation algorithm* $\mathcal{SH}_U$ takes as input a public key pk, a signature σ, an attribute vector $\vec{L}$, an opening r, a registration state st_r, a partial attribute vector $\vec{L}' \subset \vec{L}$ and outputs a proof $\bar{\pi}$.
  - The *show verification algorithm* $\mathcal{SH}_V$ is deterministic and takes as input a public key pk, a signature σ, a (partial) attribute vector $\overrightarrow{L'}$, and a proof $\bar{\pi}$. It outputs 1 (Accept) or 0 (Reject).
- The *verification algorithm* $\mathcal{V}$ is deterministic and takes as input a public key pk, a signature $\hat{\sigma}$, and a message m. It outputs 1 (Accept) or 0 (Reject).

Definition 2.5 (Correctness of Blind Signatures with Attributes). We say that a blind signature scheme with attributes BSA =$(\mathcal{G}, \mathcal{R}, \mathcal{S}, \mathcal{U}, \mathcal{V}, \mathcal{SH})$ is perfectly correct if for all public parameters pp, all vectors of attributes $\overrightarrow{L}$, all partial vectors $\overrightarrow{L'}$, and all messages m, it holds that

$$\Pr\left[\begin{array}{c} \mathcal{V}(pk,\sigma,m)=1 \wedge \\ \mathcal{SH}_V(\mathsf{pk},\sigma,\overrightarrow{L'},\bar{\pi})=1 \end{array} : \begin{array}{c} (\mathsf{pk},\mathsf{sk}) \leftarrow_\$ \mathcal{G}(\mathsf{pp}) \\ (C,\pi,\mathsf{st}_R) \leftarrow_\$ \mathcal{R}_{\mathcal{U}}(pk,\overrightarrow{L}) \\ 1 \leftarrow_\$ \mathcal{R}_{\mathcal{S}}(\mathsf{sk},C,\pi) \\ (R,st_s) \leftarrow_\$ \mathcal{S}_1(\mathsf{sk},C) \\ (e,st_u) \leftarrow_\$ \mathcal{U}_1(\mathsf{pk},R,C,m,\mathsf{st}_R) \\ S \leftarrow \mathcal{S}_2(\mathsf{sk},st_s,R,e) \\ (\sigma,r) \leftarrow \mathcal{U}_2(\mathsf{pk},st_u,R,e,S,m) \\ \bar{\pi} \leftarrow \mathcal{SH}_U(\mathsf{pk},\sigma,\vec{L},r,st_r,\vec{L}') \end{array}\right] = 1.$$

Definition 2.6 (Blindness). For the blindness definition and the blindness proof of ACL, we refer the reader to the original work of Baldimtsi and Lysyanskaya [5].

We recall the definition of strong One-More Unforgeability, where an adversary wins if it is able to output $\ell+1$ valid message-signature pairs after completing only ℓ signing sessions. In the setting of blind signatures with attributes, this game is augmented with a registration oracle.

*Definition 2.7 (Strong One-More Unforgeability (*OMUF*)).* For a blind signature with attributes BSA = $(\mathcal{G}, \mathcal{R}, \mathcal{S}, \mathcal{V}, \mathcal{SH})$, a positive integer $\ell \in \mathbb{Z}^+$, and an adversary $\mathcal{A}$, we define the game ℓ-OMUF as follow:

- **Setup.** Generate a pair of keys $(\mathsf{pk},\mathsf{sk})$, and invoke $\mathcal{A}^{\mathcal{R}(\mathsf{sk},.),\mathcal{S}(\mathsf{sk},.)}$.
- **Online Phase.** $\mathcal{A}$ may query its oracles arbitrarily and in an interleaved fashion as long as it completes at most ℓ sessions with $\mathcal{S}$.
- **Output Determination.** The game outputs 1 iff $\mathcal{A}$ outputs $k \geq \ell+1$ pairwise-distinct tuples $(\sigma_1,m_1),\ldots(\sigma_k,m_k)$ such that for all $i \in [k]$, $\mathcal{V}(pk,\hat{\sigma_i},m_i)=1$.

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

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

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

We present the definition of secure showing. Intuitively, this security definition captures that the adversary cannot re-link received signatures to other attribute vectors.

Definition 2.8 (Secure Showing). For a blind signature with attributes BSA = $(\mathcal{G}, \mathcal{R}, \mathcal{S}, \mathcal{V}, \mathcal{SH})$, a positive integer $\ell \in \mathbb{Z}^+$, and an adversary $\mathcal{A}$, we define the game ℓ-SH as follows.

- **Setup.** Generate key pair $(\mathsf{pk},\mathsf{sk})$ and invoke $\mathcal{A}^{\mathcal{R}(\mathsf{sk},\cdot),\mathcal{S}(\mathsf{sk},\cdot)}$. Oracles $\mathcal{R}(\mathsf{sk},\cdot)$ and $\mathcal{S}(\mathsf{sk},\cdot)$ share state and we require that session i with $\mathcal{S}(\mathsf{sk},\cdot)$ uniquely identifies some prior session j with $\mathcal{R}(\mathsf{sk},\cdot)$.
- **Online Phase.** The adversary may query its oracles arbitrarily and in an interleaved fashion as long as it completes at most ℓ sessions with $\mathcal{S}$. For $i \in [\ell]$, let C'_i denote the commitment corresponding to the i_{th} session with $\mathcal{S}(\mathsf{sk},\cdot)$. We define $C'_i = \perp$ if there is no session i.
- **Output Determination.** The adversary outputs up to ℓ pairwise distinct tuples of the form $(m_i,\sigma_i,C_i,\vec{L}_i,r_i,\pi_i)$ as well as ℓ pairs $(\vec{L}'_i,r'_i)$. The game outputs 1 if all of the following hold and 0 otherwise:
  - For all $i \in [\ell]$, either r'_i is a valid opening of C'_i to $\vec{L}'_i$ or $C'_i = \perp$.
  - For all tuples $(m_i,\sigma_i,\vec{L}_i,\pi_i)$ in the adversary's output, it holds that $\mathcal{SH}_V(\mathsf{pk},\sigma_i,\vec{L}_i,\pi_i)=1$.
  - For all tuples $(m_i,\sigma_i,\vec{L}_i,\pi_i)$ in the adversary's output, it holds that $\mathcal{V}(\mathsf{pk},m_i,\sigma_i)=1$.
  - All $\vec{L}_i,\vec{L}'_i$ contain an entry for each attribute, i.e., they do not contain the $\perp$ symbol anywhere.
  - The multiset of all attribute vectors $\vec{L}_i$ is not a subset of the multiset of all attribute vectors $\vec{L}'_i$ that correspond to $C'_i \neq \perp$.

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

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

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

Remark 2.1 (On Modularizing the Security Properties of Blind Signatures with Attributes). We note that compared to [5], we have split up the definition of security of blind signatures with attributes into definition 2.7 and definition 2.8. This is on the one hand to modularize the two different security properties, which may require

very different proof strategies, and on the other hand to clarify the Secure Showing Property which was ambiguous in [5].

Remark 2.2 (Limitations of the Secure Showing Definition). We note that definition 2.8 does not cover all possible attacks on the Secure Showing algorithm that one might exclude. For example, one might want to exclude an attack where the adversary only has to provide partial vectors such that there is no mapping to the full vectors he requested signatures for. However, this winning condition is likely inefficient to check as the challenger would have to verify that no such mapping can exists, for example by iterating over all possible mappings. We note that letting the adversary provide a mapping that violates security is not a good fix for this, as the adversary could provide a bad mapping even if a good mapping exists, e.g. if L_1', L_2' are subvectors of L_a and L_3' is a subvector of L_b, but L_2 is also a shared subvector of L_a and L_b, the adversary could claim in the mapping that he got two signatures for L_b and one for L_a, when the signatures are in fact two signatures for L_a and one for L_b.

Therefore, we require the adversary to open the full vectors as well.

3 AN IDENTIFICATION SCHEME FOR ANONYMOUS CREDENTIALS LIGHT

Towards a modular OMUF proof of Anonymous Credentials Light ACL, we propose an identification scheme, called $\mathsf{ID}_{\mathsf{ACL}}$, from which the scheme ACL can be derived, and from its OMMIM-security the OMUF-security of ACL can be proven. Particularly, following a similar methodology of Hauck et al. [38], we show in this section that $\mathsf{ID}_{\mathsf{ACL}}$ is OMMIM-secure and use this fact in section 4 to show via reduction that ACL is OMUF-secure.

3.1 Construction

For the reader's convenience, we depict this scheme as an interactive protocol in fig. 4. We define the scheme $\mathsf{ID}_{\mathsf{ACL}} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ as follows:

- $\mathcal{G}(pp)$: Parse pp to obtain $\mathbb{G}, q$, and g, sample $h \leftarrow_\$ \mathbb{G}$ and $x, v_{0,1}, v_{0,2} \leftarrow_\$ \mathbb{Z}_q$, compute $y = g^x, z := g^{v_{0,1}} \cdot h^{v_{0,2}}$, set $\mathsf{sk} = (g, h, z, x)$ and $\mathsf{pk} = (g, h, z, y)$, 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 $d, s_1, s_2, u, \mathrm{rnd} \leftarrow_\$ Z_q$, compute $z_1 := g^{\mathrm{rnd}}, z_2 := z/z_1, a := g^u, b_1 := g^{s_1} \cdot z_1^d, b_2 := h^{s_2} \cdot z_2^d$, set $R := (a, b_1, b_2, \mathrm{rnd}), st_p := (d, s_1, s_2, u)$ 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 $c := \varepsilon - d, r := u - c \cdot x$, set and output $S := (c, d, r, s_1, s_2)$.
- The verifier algorithms $\mathcal{V} = (\mathcal{V}_1, \mathcal{V}_2)$ are defined as follows:
  - $\mathcal{V}_1(\mathsf{pk}, R)$: Parse pk, and R, sample $\varepsilon \leftarrow_\$ Z_q$, set $st_v := (a, b_1, b_2, z_1, z_2)$, and output (ε, 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, it outputs 0.

3.2 Perfect Correctness.

$\mathsf{ID}_{\mathsf{ACL}}$ is perfectly correct. The verifier verification equation checks that $g^r \cdot y^c = a \wedge g^{s_1} \cdot z_1^d = b_2 \wedge c + d = \varepsilon$ holds. In an honest run of the protocol, this always holds, because

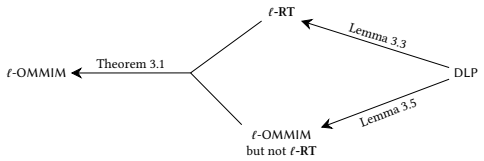

Figure 3: Proof overview for theorem 3.1.

- $g^r \cdot y^c = g^{u-cx} \cdot g^{cx} = g^u = a$,
- $g^{s_1} \cdot z_1^d = b_1$, and $h^{s_2} \cdot z_2^d = b_2$,
- $c + d = \varepsilon - d + d = \varepsilon$.

3.3 One-More Man-In-The-Middle (OMMIM)

Theorem 3.1. *Let $\mathbb{G}$ be a group of prime order q where the* DLP *is (t, ε)-hard. Then $\mathsf{ID}_{\mathsf{ACL}}$ is (t', ε')-ℓ-OMMIM-secure in the AGM + ROM with $t' \approx t$ and $\varepsilon' \leq 4 \cdot \varepsilon + \frac{2 \cdot \binom{q_h}{2} + 11}{q}$*

Notation and Proof Strategy. When $\mathcal{A}$ runs the ℓ-OMMIM game, it acts as a Man-In-The-Middle between the prover and the verifier and may open sessions with both. We examine the elements exchanged among the prover, the verifier, and $\mathcal{A}$. To differentiate between the elements sent to the verifier by $\mathcal{A}$ and those sent by the prover, we keep the names of the elements sent by the prover as $(z_1, a, b_1, b_2, c, d, r, s_1, s_2)$, and correspondingly, $z_2 := z/z_1$, and rename the elements sent by $\mathcal{A}$ as $\zeta_1 := z_1, \alpha := a, \beta_1 := b_1, \beta_2 := b_2, \omega := c, \delta := d, \rho := r, \sigma_1 := s_1$ and $\sigma_2 := s_2$, and correspondingly, $\zeta_2 := \zeta_1/z$.

Our proof strategy consists of two steps. In the first step, we show via reduction under the DL hardness assumption that an adversary $\mathcal{A}$ cannot win the game ℓ-OMMIM while using *a fresh tag* ζ_1 in one of the accepting sessions. A fresh tag ζ_1 is a tag that is not generated by the prover in one of the closed prover sessions, i.e., for all closed prover sessions $1 \leq i \leq \ell$, we have $\zeta_1 \neq z_{1,i}$. We formalize this requirement in a game called *the Restrictive Tagging (ℓ-RT)* and provide a reduction of the hardness of this game to the hardness of the discrete logarithm problem. We turn to the second step: If $\mathcal{A}$ wins the game ℓ-OMMIM without using a fresh tag, then by the pigeonhole principle, there are at least two verifier sessions k and l, such that, $\zeta_{1,k} = \zeta_{1,l}$, because $\mathcal{A}$ closes $\ell + 1$ sessions, while the prover closes only ℓ sessions. In the second step, we show via reduction that $\mathcal{A}$ cannot win the game ℓ-OMMIM if two verifier sessions have the same tag ζ_1 because otherwise $\mathcal{A}$ wins the DLP game.

While interacting with the reduction, the algebraic adversary $\mathcal{A}$ provides a representation for each group element it outputs or uses in its queries to $\mathcal{V}_1$. These representations must be expressible using a basis consisting of the group elements $\mathcal{A}$ acquires during the ℓ-RT game. More specifically, this includes the group elements of the public key (g, y, h, z), and the group elements that it receives by interacting with $\mathcal{P}_1$. Assume $\mathcal{A}$ has invoked $\mathcal{P}_1$ k times for $1 \leq k$, the basis in the k_{th} verifier session is $Z_k := (g, y, h, z, a_1, ..., a_k, b_{1,1}, ..., b_{1,k}, b_{2,1}, ..., b_{2,k}, z_{1,1}, ..., z_{1,k})$.

Recall that the group elements a, b_1 and b_2 can be expressed as $a = g^r \cdot y^c, b_1 = g^{s_1} \cdot z_1^d$, and $b_2 = h^{s_2} \cdot z_2^d = h^{s_2} \cdot (z/z_1)^d$. Additionally, the discrete logarithms of h, z, and z_1 are known for

the reduction. Thus, the reduction can represent each element in Z_k using the reduced basis $I := (g, y, h)$. While $g, y,$ and h remain unchanged, we rewrite the other elements in Z_k as follows:

$z = g^{v_{0,1}} \cdot h^{v_{0,2}}, z_{1,i} = g^{\mathrm{rnd}_i}$, $a_i = g^{r_i} \cdot y^{c_i}, b_{1,i} = g^{s_{1,i}+\mathrm{rnd}_i \cdot d_i}$, and $b_{2,i} = h^{s_{2,i}+v_{0,2} \cdot d} \cdot g^{d \cdot (v_{0,1}-\mathrm{rnd}_i)}$, where $1 \le i \le k$. For simplicity, we often reduce given representations into the basis I. We can rewrite any representation of a group element o in its reduced form (w.r.t. basis I) as $g_{[o]}+x \cdot y_{[o]}+w \cdot h_{[o]}$, where $x := \mathrm{dlog}_g\, y$ and $q := \mathrm{dlog}_g\, h$. The notation $g_{[o]}, y_{[o]},$ or $h_{[o]}$ refers to the respective component of the group element o.

In addition to the reduced representations, we need to argue about the representations that $\mathcal{A}$ submits before we transform these representations into their reduced form. Using the notation $\overline{a_i(o)}$, we refer to the exponent of a_i in the representation of the group element o, where a_i is the group element a that is sent to $\mathcal{A}$ by the prover in response to a $\mathcal{P}_1$ query in session i. If no a_i component occurs in the representation of o for all opened prover sessions i, then $\overline{a_i(o)} = 0$. Similarly, we use the notation $\overline{b_{k,i}(o)}$ for $k \in \{1, 2\}$ to refer to the exponent of $b_{k,i}$ in the group element o, where b_k is sent in the response to $\mathcal{P}_1$ query in the session i. If no $b_{k,i}$ component occurs in the representation of o for all opened prover sessions i, then $\overline{b_{k,i}(o)} = 0$.

We say that a prover session i is linked to verifier session j if $\overline{a_i(\alpha_j)} \neq 0$, $\overline{b_{2,i}(\beta_{1,j})} \neq 0$, $\overline{b_{2,i}(\beta_{2,j})} \neq 0$, $\overline{b_{2,i}(\zeta_{1,j})} \neq 0$, or $\overline{b_{2,i}(\zeta_{2,j})} \neq 0$.

Proof. Firstly, we define the game ℓ-*Restrictive Tagging* (ℓ-**RT**) game that captures the event in which $\mathcal{A}$ wins the ℓ-OMMIM game while using a fresh tag ζ_1 to close one of the verifier accepting sessions.

*Definition 3.2 (***Game ℓ-*Restrictive Tagging* (ℓ-**RT**)*).* For a positive integer $\ell \in \mathbb{Z}^+$, we define the game ℓ-**RT** for an identification scheme $\mathsf{ID} := (\mathcal{G}, \mathcal{P}, \mathcal{V})$, a public parameter pp, and an adversary $\mathcal{A}$ as follows.

Initialization. The same as ℓ-OMMIM.

Online Phase. The same as ℓ-OMMIM.

Output Determination. The game outputs 1 if ℓ-OMMIM game outputs 1 and there is an accepting session i with the verifier, such that, for all prover sessions j that are closed with $\mathcal{P}_2$ invocation, it holds that $\zeta_{1,i} \neq z_{1,j}$. We assume w.l.o.g. that there is a single verifier session with this property and call it *the special session.*

We define the advantage of an adversary $\mathcal{A}$ in winning the game for an identification scheme ID as

$$\mathbf{Adv}^{\ell\text{-RT}}_{\mathsf{ID}}(\mathcal{A}) := \Pr[\ell\text{-}\mathbf{RT}^{\mathcal{A}}_{\mathsf{ID}} = 1].$$

Lemma 3.3 (**Restrictive Tagging Lemma**). *For all adversaries $\mathcal{A}$ that run the game ℓ-**RT** against $\mathsf{ID}_{\mathsf{ACL}}$, there is a reduction $\mathcal{B}$ running against the* DLP *game, such that*

$$\mathbf{Adv}^{\mathsf{DLP}}(\mathcal{B}) \ge \frac{1}{2} \cdot \mathbf{Adv}^{\ell\text{-RT}}_{\mathsf{ID}_{\mathsf{ACL}}}(\mathcal{A}) - \frac{9}{2 \cdot q}.$$

Proof. Assuming $\mathcal{A}$ wins the game ℓ-**RT**, it follows that it wins the game ℓ-OMMIM while using a fresh tag ζ_1 in one of the accepting verifier sessions (i.e., the special session). We show in the following that this enables the reduction $\mathcal{B}$ to win the game DLP, by embedding the DLP challenge U in either h or y, and extracting its discrete logarithm from the representations of the group elements of the special session. We illustrate this strategy in the following via a sequence of games.

Game $\mathbf{G_1}$. The game samples a bit $b \leftarrow_{\$} \{0, 1\}$. If $b = 0$, it behaves the same as ℓ-**RT** (i.e., it uses the y-side simulator C.2.1). If $b = 1$, it uses the z-side simulator C.2.2. Note that the game $\mathbf{G_1}$ and the game ℓ-**RT** are identically distributed due to the witness indistinguishability of the OR-proof [23].

Game $\mathbf{G_2}$. The same as the game $\mathbf{G_1}$, except that it aborts if an event E_1 occurs, which we define in the following.

Definition 3.4 (The Preliminary Values). Assume that during the game $\mathbf{G_1}$, $\mathcal{A}$ opens n prover sessions and links k sessions of them with the special session for $0 \le k \le n$. If a variable v describes a value that is generated in a prover session that is linked to the special session, we mark v with an asterisk, e.g., a^*, b^*_2. Let $\overrightarrow{a^*} = (a^*_1, ..., a^*_k)$, $\overrightarrow{b^*_2} = (b^*_{2,1}, ..., b^*_{2,k})$, $\overrightarrow{c^*} = (c^*_1, ..., c^*_k)$, and $\overrightarrow{d^*} = (d^*_1, ..., d^*_k)$ be the values generated by the k prover sessions linked to the special session (ordered by the time each session is opened). The reduction $\mathcal{B}$ transforms the representations of the group elements $\alpha, \beta_1, \beta_2, \zeta_1, \zeta_2$ of the special session to their reduced form w.r.t. the basis I. It then uses these reduced representations to compute new group elements $\alpha', \beta'_1, \beta'_2, \zeta'_1$, and ζ'_2 as follows: it computes α' from the representation of α after eliminating any a^*_i components in this representation. That is, if a^*_i occurs in the representation of α, i.e., $\overline{a^*_i(\alpha)} \neq 0$, for some session i, then we have $\alpha' := \alpha / a_i^{*\,\overline{a^*_i(\alpha)}}$. For example, assume that $\overline{a^*_i(\alpha)} \neq 0$ and the y-side simulator is used, it follows that $a_i^{*\,\overline{a^*_i(\alpha)}} = g^{u^*_i \cdot \overline{a^*_i(\alpha)}}$, and thus, we have $g_{[\alpha']} := g_{[\alpha]} - u^*_i \cdot \overline{a^*_i(\alpha)}$, $y_{[\alpha']} := y_{[\alpha]}$, and $h_{[\alpha']} := h_{[\alpha]}$. Analogously, we construct $\beta'_1, \beta'_2, \zeta'_1,$ and ζ'_2 from the representations of $\beta_1, \beta_2, \zeta_1,$ and ζ_2, respectively, after eliminating all $b^*_{2,i}$ components from these representations.

We define the following functions for the special session:

$$\omega'(\overrightarrow{c^*}) := y_{[\alpha']} + \sum_{j=1}^{k} \overline{a^*_i(\alpha)} \cdot c^*_i,$$

$$\delta''(\overrightarrow{d^*}) := \frac{g_{[\beta'_2]} + x \cdot y_{[\beta'_2]} + \sum_{i=1}^{k} \overline{b^*_{2,i}(\beta_2)} \cdot d^*_i \cdot (v_{0,1} - v^*_{1,i})}{g_{[\zeta'_2]} + x \cdot y_{[\zeta'_2]} + \sum_{i=1}^{k} \overline{b^*_{2,i}(\zeta_2)} \cdot d^*_i \cdot (v_{0,1} - v^*_{1,i})},$$

where $u^*_{2,i} := \mathrm{dlog}_h\, b^*_{2,i}$.

We prove the following properties of the preliminary values:

Claim 3.1. *Let $\alpha, \beta_1, \zeta_1, \beta_2, \zeta_2$ be the group elements from the special sessions in lemma 3.3, and let $\alpha', \beta'_1, \beta'_2, \zeta'_1$, and ζ'_2 be the values described in definition 3.4. It holds that*

(1) $y_{[\alpha]} = y_{[\alpha']} + \sum_{i=1}^{k} c^*_i \cdot \overline{a^*_i(\alpha)}$

(2) $h_{[\beta_2]} = h_{[\beta'_2]} + \sum_{i=1}^{k} \overline{b^*_{2,i}(\beta_2)} \cdot d^*_i \cdot (v_{0,1} - v^*_{1,i})$

(3) $h_{[\zeta_2]} = h_{[\zeta'_2]} + \sum_{i=1}^{k} \overline{b^*_{2,i}(\zeta_2)} \cdot d^*_i \cdot (v_{0,1} - v^*_{1,i})$

where k is the number of prover sessions linked to the special session.

The proof is given in appendix B.1.

Remark 3.1. We note that if the special session is not linked to any prover session, the arguments $\overrightarrow{c^*}$ and $\overrightarrow{d^*}$ are empty, and thus, the sums in the function definitions are also empty. Furthermore, in case the special session is linked to unclosed prover sessions, the simulator closes these sessions by itself which sets the values for all previously undefined $c_i, r_i, d_i, s_{1,i}, s_{2,i}$.

Observation 3.1. We note that the prover's view of the reduced representations w.r.t. I of the group elements a_i^* and $b_{2,i}^*$ are not fixed. In contrast to the other group elements in $Z_k = (g, y, h, a_1, \ldots, a_k, b_{1,1}, \ldots, b_{1,k}, b_{2,1}, \ldots, b_{2,k}, z_1, \ldots, z_k, z_{1,1}, \ldots, z_{1,k})$, the reduced representations of a_i^* and $b_{2,i}^*$ may change after $\mathcal{P}_2$ query. For example, if the y-side simulator is used (see appendix C.2.1), the prover's view of a_i^* changes from $g^{u_i^*}$ to $g^{r_i^*} \cdot y^{c_i^*}$, hence its g and y components get modified. Similarly, the representation of $b_{2,i}^*$ changes when the z-side simulator (see appendix C.2.2) is used. Therefore, using a_i^* or $b_{2,i}^*$ in the representation of another group element makes the reduced representation of the latter prone to modifications after $\mathcal{P}_2$ query. However, note that the group elements $\alpha', \zeta_1', \zeta_2', \beta_1'$ and β_2' are (partially) resilient to these changes when $\mathcal{P}_2$ is queried. In particular, as α' does not contain a_i^* component, $y_{[\alpha']}$ is fixed. Similarly, as $\zeta_1', \zeta_2', \beta_1'$ and β_2' do not have $b_{2,i}^*$ components, $h_{[\beta_1']}, h_{[\zeta_1']}, g_{[\beta_2']} + x \cdot y_{[\beta_2']}$, and $g_{[\zeta_2']} + x \cdot y_{[\zeta_2']}$ are fixed.

Observation 3.2. Per observation 3.1, we know that $\mathcal{A}$ fixes the values $y_{[\alpha']}, h_{[\beta_1']}, h_{[\zeta_1']}, g_{[\beta_2']} + x \cdot y_{[\beta_2']}$, and $g_{[\zeta_2']} + x \cdot y_{[\zeta_2']}$ at the time of $\mathcal{V}_1$ query and it cannot influence these values afterward. After these values are fixed, the value ε is sampled uniformly at random. Finally, only after the value ε is fixed, the values c_i^* and d_i^* get fixed (before $\mathcal{V}_2$ query, c_i^* (d_i^*) is uniformly random and information-theoretically hidden from $\mathcal{A}$ when the z-side simulator from appendix C.2.2 (y-side simulator from appendix C.2.1) is used).

The functions ω' and δ'' are crucial for the reduction $\mathcal{B}$, because it uses them to extract the solution of the DLP instance from the special session. However, this is only possible if the function $\delta''(\overrightarrow{d^*})$ is defined. We define the event E_1 to be the event in which $\delta''(\overrightarrow{d^*})$ is undefined. For simplicity, we know per claim 3.1 that $\delta''(\overrightarrow{d^*}) = \frac{g_{[\beta_2]} + x \cdot y_{[\beta_2]}}{g_{[\zeta_2]} + x \cdot y_{[\zeta_2]}}$, thus we define E_1 as

$$\mathsf{E}_1 \coloneqq (g_{[\zeta_2]} + x \cdot y_{[\zeta_2]} = 0).$$

Claim 3.2. $\Pr[\mathsf{E}_1] \leq \frac{1}{q}$.

Proof. Recall that z is sampled as $z \coloneqq g^{v_{0,1}} \cdot h^{v_{0,2}}$ for $v_{0,1}, v_{0,2} \in \mathbb{Z}^*$, and $\zeta_2 \coloneqq z/\zeta_1$, hence $g_{[\zeta_2]} + x \cdot y_{[\zeta_2]} = g_{[z]} + x \cdot y_{[z]} - (g_{[\zeta_1]} + x \cdot y_{[\zeta_1]}) = v_{0,1} - \text{rnd}$ and $h_{[\zeta_2]} = h_{[z]} - h_{[\zeta_1]} = h_{[z]}$. Thus, in order for E_1 to occur, the adversary must choose rnd, such that $\text{rnd} = v_{0,1}$. However, as $v_{0,1}$ is information-theoretically hidden from the adversary at all times, the probability that $\text{rnd} = v_{0,1}$ is $\frac{1}{q}$. □

By this claim, it follows that $\mathbf{Adv}^{\mathbf{G}_2}(\mathcal{A}) \geq \mathbf{Adv}^{\mathbf{G}_1}(\mathcal{A}) - \frac{1}{q}$; therefore, we assume that $\mathcal{A}$ wins the game $\mathbf{G_2}$.

Game $\mathbf{G_3}$. The same as the game $\mathbf{G_2}$, except that it aborts if an event E_2 occurs for the special session.

Define $\mathsf{E}_2 \coloneqq (\omega'(\overrightarrow{c^*}) = \omega) \wedge (\delta''(\overrightarrow{d^*}) = \delta) \wedge \neg\mathsf{E}_1$.

Claim 3.3. $\Pr[\mathsf{E}_2] \leq \frac{8}{q}$.

The proof is given in appendix B.2. By this claim, we have that $\mathbf{Adv}^{\mathbf{G}_3}(\mathcal{A}) \geq \mathbf{Adv}^{\mathbf{G}_2}(\mathcal{A}) - \frac{8}{q}$; therefore, we assume that $\mathcal{A}$ wins $\mathbf{G_3}$.

Next, we show that if $\mathcal{A}$ wins $\mathbf{G_3}$, then $\mathcal{B}$ can use $\mathcal{A}$ to extract the discrete logarithm of y or h using the representations of the group elements of the special session.

Simulating $\mathbf{G_3}$: $\mathcal{B}$ simulates $\mathbf{G_3}$ by simulating the oracles $\mathcal{P}_1, \mathcal{P}_2$, $\mathcal{V}_1$,and $\mathcal{V}_2$. While the simulation of $\mathcal{V}_1$ and $\mathcal{V}_2$ is done as in $\mathsf{ID}_{\mathsf{ACL}}$ (see fig. 4), the simulation of $\mathcal{P}_1$ and $\mathcal{P}_2$ can be done in two ways. Given a DLP challenge U, $\mathcal{B}$ flips a coin and does the following:

(1) On heads, it simulates the prover algorithms using the simulator (C.2.1), but instead of sampling h at random, it sets $h \coloneqq U$.
(2) On tails, it simulates the prover using the simulator (C.2.2), but instead of sampling y randomly, it sets $y \coloneqq U$.

$\mathcal{B}$ maintains a per-session storage to store all the group elements used during the simulation alongside the representations submitted by the algebraic adversary $\mathcal{A}$. We show in the following claim the technique used by $\mathcal{B}$ to extract $\text{dlog}_g U$.

Claim 3.4. $\mathbf{Adv}^{\mathsf{DLP}}(\mathcal{B}) \geq \frac{1}{2} \cdot \mathbf{Adv}^{\mathbf{G}_3}(\mathcal{A})$.

Proof. Recall that $\mathcal{A}$ has to query the verifier oracle $\mathcal{V}_1$ $\ell + 1$ times, and, as $\mathcal{A}$ is algebraic, it submits representations for the group elements in the queries; therefore, the reduction $\mathcal{B}$ has access to the representations of the group elements $\alpha, \beta_2,$ and ζ_2, which it can transform into their reduced form to the basis I and compute the values ω', and δ'' for the special session.

Furthermore, since $\mathcal{A}$ wins the game $\mathbf{G_3}$, the events E_1 and E_2 do not occur. Per observation 3.3, we have that

$$\neg\mathsf{E}_1 \wedge \neg\mathsf{E}_2 \equiv$$
$$y_{[\alpha]} \neq \omega \vee g_{[\zeta_2]} + x \cdot y_{[\zeta_2]} \neq 0 \wedge \frac{g_{[\beta_2]} + x \cdot y_{[\beta_2]}}{g_{[\zeta_2]} + x \cdot y_{[\zeta_2]}} \neq \delta.$$

$\mathcal{B}$'s strategies to extract the discrete logarithm of U depends on which of the following cases occurs (the following cases are not mutually exclusive; therefore, $\mathcal{B}$ follows the strategy of the first matching case):

Case 1: $\omega \neq y_{[\alpha]}$. $\mathcal{B}$ extracts the solution of the DLP instance from y, i.e., $x = \text{dlog}_g U = \text{dlog}_g y$. This is possible if the simulator C.2.2 is used since $\mathcal{B}$ embeds U in y.
We know from the verification equation that $\alpha = g^\rho \cdot y^\omega$, hence we can write its representation as $\rho + x \cdot \omega$, where $x = \text{dlog}_g y$. From the representation of α that $\mathcal{A}$ submits to the verifier, $\mathcal{B}$ computes the reduced representation of α as $g_{[\alpha]} + x \cdot y_{[\alpha]} + w \cdot h_{[\alpha]}$.
It follows that $\rho + x \cdot \omega = g_{[\alpha]} + x \cdot y_{[\alpha]} + w \cdot h_{[\alpha]}$, which yields the equation $x = \frac{g_{[\alpha]} + w \cdot h_{[\alpha]} - \rho}{\omega - y_{[\alpha]}}$.
This equation is solvable because $\omega \neq y_{[\alpha]}$.

Case 2: $g_{[\zeta_2]} + x \cdot y_{[\zeta_2]} \neq 0 \wedge \frac{g_{[\beta_2]} + x \cdot y_{[\beta_2]}}{g_{[\zeta_2]} + x \cdot y_{[\zeta_2]}} \neq \delta$: In this case, $\mathcal{B}$ extracts $w = \text{dlog}_g U = \text{dlog}_g h$. This is possible if the simulator C.2.1 is used since $\mathcal{B}$ embeds U in h.

From the verification equation, we have that $\beta_2 := h^{\sigma_2} \cdot \zeta_2^{\delta}$, hence the representation of β_2 can be expressed as

$$w \cdot \sigma_2 + \delta \cdot (g_{[\zeta_2]} + x \cdot y_{[\zeta_2]} + w \cdot h_{[\zeta_2]}), \tag{4}$$

where $w = \mathrm{dlog}_g h$. Recall that $\mathcal{A}$ submits the representation of β_2 to the verifier. This allows $\mathcal{B}$ to compute its reduced form as

$$g_{[\beta_2]} + x \cdot y_{[\beta_2]} + w \cdot h_{[\beta_2]}. \tag{5}$$

We set $\psi := w, A := \sigma_2, B := \delta, C := g_{[\zeta_2]} + x \cdot y_{[\zeta_2]}, D := h_{[\zeta_2]}, E := g_{[\beta_2]} + x \cdot y_{[\beta_2]}, F := h_{[\beta_2]}, \mathcal{V}_1 := \psi \cdot A + B \cdot (C + \psi \cdot D)$, and $\mathcal{V}_2 := E + \psi \cdot F$. By claim D.3, w can be recovered since the condition $\mathcal{V}_1 = \mathcal{V}_2 \wedge C \neq 0 \wedge B \neq \frac{E}{C}$ holds.

For each of the cases, the solution to the DLP instance can be extracted if the appropriate simulator is used, which happens with probability $\frac{1}{2}$. Thus, $\mathcal{B}$ can compute $\mathrm{dlog}_g U$ successfully with probability $\geq \frac{1}{2} \cdot \mathbf{Adv}^{\mathbf{G}_3}(\mathcal{A})$. □

Consequently, $\mathbf{Adv}^{\mathsf{DLP}}(\mathcal{B}) \geq \frac{1}{2} \cdot \mathbf{Adv}^{\mathbf{G}_3}(\mathcal{A}) \geq \frac{1}{2} \cdot \mathbf{Adv}^{\mathbf{G}_2}(\mathcal{A}) - \frac{8}{2 \cdot q} \geq \frac{1}{2} \cdot \mathbf{Adv}^{\mathbf{G}_1}(\mathcal{A}) - \frac{1}{2 \cdot q} - \frac{8}{2 \cdot q} = \frac{1}{2} \cdot \mathbf{Adv}^{\ell\text{-}\mathbf{RT}}_{\mathsf{ID}_{\mathsf{ACL}}}(\mathcal{A}) - \frac{1}{2 \cdot q} - \frac{8}{2 \cdot q}$, hence $\mathbf{Adv}^{\mathsf{DLP}}(\mathcal{B}) \geq \frac{1}{2} \cdot \mathbf{Adv}^{\ell\text{-}\mathbf{RT}}_{\mathsf{ID}_{\mathsf{ACL}}}(\mathcal{A}) - \frac{9}{2 \cdot q}$. □

By this lemma, $\mathcal{A}$ cannot win the game ℓ-**RT**, hence if $\mathcal{A}$ wins the game ℓ-OMMIM, it may not use a fresh tag ζ_1 that was not generated by the prover.

Lemma 3.5. *For all adversaries $\mathcal{A}$, define E_3 to be the event in which $\mathcal{A}$ wins the ℓ-OMMIM game but loses the ℓ-**RT** game. There exists a reduction $\mathcal{B}$, such that*

$$\mathbf{Adv}^{\mathsf{DLP}}(\mathcal{B}) \geq \frac{1}{2} \cdot \Pr[\mathsf{E}_3] - \frac{1}{q}.$$

Proof. As $\mathcal{A}$ loses the game ℓ-**RT** while winning the game ℓ-OMMIM, it only uses tags ζ_1 that were generated by the prover. We show in the following via a sequence of games that this leads to solving the DLP.

Game $\mathbf{G_1}$. The same as the ℓ-OMMIM game, except that it aborts if a fresh tag ζ_1 is used in any of the accepting sessions (recall that *fresh* means that it was not generated by the prover as z_1 in a closed session). Since the prover only closes ℓ sessions, while $\mathcal{A}$ closes $\ell+1$ sessions, $\mathcal{A}$ must use the same tag ζ_1 in at least two different sessions. We assume w.l.o.g. that only two sessions share the same tag ζ_1. Formally, there are two protocol runs between $\mathcal{A}$ and the verifier with transcripts $(R^{(1)}, \varepsilon^{(1)}, S^{(1)})$, and $(R^{(2)}, \varepsilon^{(2)}, S^{(2)})$ with $R^{(1)} := (\alpha^{(1)}, \beta_1^{(1)}, \beta_2^{(1)}, \mathrm{rnd}), R^{(2)} := (\alpha^{(2)}, \beta_1^{(2)}, \beta_2^{(2)}, \mathrm{rnd}), \varepsilon^{(1)} \neq \varepsilon^{(2)}$, and $S^{(1)} \neq S^{(2)}$. We call these sessions *the special sessions*. We note that, if $R^{(1)} = R^{(2)}$, the combination of the two transcripts directly yields one of the witnesses regardless of the algebraic representations submitted by the adversary. We therefore assume in the following that $R^{(1)} \neq R^{(2)}$.

Game $\mathbf{G_2}$. The same as game $\mathbf{G_1}$, except that it aborts if $g_{[\zeta_2]} + x \cdot y_{[\zeta_2]} = 0$.

Claim 3.5. $\Pr[g_{[\zeta_2]} + x \cdot y_{[\zeta_2]} = 0] \leq \frac{1}{q}$.

Proof. Recall that $\zeta_2 = z/\zeta_1$, hence $g_{[\zeta_2]} + x \cdot y_{[\zeta_2]} = g_{[z]} + x \cdot y_{[z]} - g_{[\zeta_1]} + x \cdot y_{[\zeta_1]}$; therefore, if $g_{[\zeta_2]} + x \cdot y_{[\zeta_2]} = 0$, $g_{[z]} + x \cdot y_{[z]} = g_{[\zeta_1]} + x \cdot y_{[\zeta_1]}$. Furthermore, z is chosen by the reduction as $z = g^{v_{0,1}} \cdot h^{v_{0,2}}$, and ζ_1 is chosen by the prover as $z_1 := g^{\mathrm{rnd}}$ (recall that $\mathcal{A}$ loses the game ℓ-**RT**, hence it only chooses tags that are generated by the prover). Consequently $g_{[z]} + x \cdot y_{[z]} = g_{[z_1]} + x \cdot y_{[z_1]}$ only occurs if $v_{0,1} = \mathrm{rnd}$, which occurs with a probability at most $\frac{1}{q}$. □

By this claim, we have that $\mathbf{Adv}^{\mathbf{G}_2} \geq \mathbf{Adv}^{\mathbf{G}_1} - \frac{1}{q}$, and thus, we assume that $\mathcal{A}$ wins the game $\mathbf{G_2}$.

Game $\mathbf{G_3}$. The same as game $\mathbf{G_2}$, except that it aborts if an event E_4 happens. We define E_4 in the following.

Definition 3.6 (The Preliminary Values). Given the commitments $R^{(j)} = (\alpha^{(j)}, \beta_1^{(j)}, \beta_2^{(j)}, \mathrm{rnd})$ for $j = 1, 2$ from the transcripts of the special sessions, the reduction transforms the representations of $\alpha^{(j)}, \beta_2^{(j)}$, and $\zeta_2 = z/\zeta_1$ to their reduced form w.r.t. the basis I. Using these representations, the reduction computes the values $\alpha'^{(j)}, \beta_2'^{(j)}$, and ζ_2' in a similar fashion to definition 3.4, and defines the following preliminary values:

$$\omega_j'(\overrightarrow{c_j^*}) := y_{[\alpha'^{(j)}]} + \sum_{i \in S_j} \overline{a_i^*(\alpha^{(j)})} \cdot c_i^*,$$

$$\delta_j'(\overrightarrow{d_j^*}) := \frac{g_{[\beta_2'^{(j)}]} + x \cdot y_{[\beta_2'^{(j)}]} + \sum_{i \in S_j} \overline{b_{2,i}^*(\beta_2^{(j)})} \cdot d_i^* \cdot (v_{0,1} - \mathrm{rnd}_i^*)}{g_{[\zeta_2']} + x \cdot y_{[\zeta_2']}},$$

where S_j is a list containing the session identifiers of the prover sessions linked to the special session j sorted by the time the sessions are closed in an ascending order, $\overrightarrow{c_j^*} := (c_{j_1}^*, \ldots, c_{j_k}^*), \overrightarrow{d_j^*} := (d_{j_1}^*, \ldots, d_{j_k}^*)$, and j_k is the number of prover sessions linked to the special session j.

Observation 3.3. Note that observation 3.1 applies on the current definitions of α', β_2', and ζ_2. Particularly, these values get fixed by $\mathcal{V}_1$ query and cannot be influenced by $\mathcal{A}$ afterward. Furthermore, observation 3.2 applies on this lemma: first, $\mathcal{A}$ fixes the values α', β_2', and ζ_2 in $\mathcal{V}_1$ query, then ε is fixed and revealed to $\mathcal{A}$ in the response to $\mathcal{V}_1$ query, and finally, c_i^* and d_i^* are revealed in the response to $\mathcal{P}_2$ query. Additionally, one of the values c_i^* and d_i^* is uniformly random, and it holds that $c_i^* + d_i^* = \varepsilon_i$.

Let ω_j and δ_j be the field elements sent by $\mathcal{A}$ in the $\mathcal{V}_2$ query of the special session j. We define E_4 as

$$\mathsf{E}_4 := (\omega_j'(\overrightarrow{c_j^*}) = \omega_j) \wedge (\delta_j'(\overrightarrow{d_j^*}) = \delta_j)$$

for both $j = 1, 2$.

Claim 3.6. *For a polynomially-large integer $q_h \in \mathbb{Z}^+$,*

$$\Pr[\mathsf{E}_4] \leq \frac{\binom{q_h}{2} + 1}{q}.$$

Proof. Deifne the varibles $C'_{j,0} = \sum_{i=1}^{j_k - 1} c_i^* \cdot \overline{a_i^*(\alpha^{(j)})}, C'_{j,2} = \sum_{i=1}^{j_k - 1} \overline{b_{2,i}^*(\beta_2^{(j)})} \cdot d_i^* \cdot (v_{0,1} - \mathrm{rnd}_i^*), \xi^{(j)} := \varepsilon_{j_k}, \xi' := \varepsilon_{j_k}, \Delta^{(j)} := d_{j_k}^*, \Omega^{(j)} := c_{j_k}^*, C_0{}^{(j)} := y_{[\alpha'^{(j)}]} + C_0'^{(j)}, C_1{}^{(j)} := \overline{a_k^*(\alpha^{(j)})}, C_2{}^{(j)} := g_{[\beta_2'^{(j)}]} + x \cdot y_{[\beta_2'^{(j)}]} + C_2'^{(j)}, C_3{}^{(j)} := g_{[\zeta_2']} + x \cdot y_{[\zeta_2']} = g_{[\zeta']} +$

$x \cdot y_{[\zeta']} - g_{[\zeta_1']} + x \cdot y_{[\zeta_1']} = v_{0,1} - \text{rnd}$ where $\text{rnd} = \text{dlog}_g \zeta_1'$, ${C_4}^{(j)} := (v_{0,1} - \text{rnd}_k^*) \cdot \overline{b_{2,k}^*(\beta_2^{(j)})}$, and ${C_5}^{(j)} := 0$.

If the event E_4 occurs, we have the following overdetermined system of linear equations

$$ {C_0}^{(j)} + {C_1}^{(j)} \cdot \Omega^{(j)} + \frac{{C_2}^{(j)} + {C_4}^{(j)} \cdot (\xi - \Omega^{(j)})}{{C_3}^{(j)}} = {\xi'}^{(j)}. $$

for $j = 1, 2$.

Rearranging the equations yields

$$ \xi = \frac{1}{{C_4}^{(j)}} \cdot (-{C_3}^{(j)} \cdot ({C_0}^{(j)} + {C_1}^{(j)} \cdot \Omega^{(j)} - {\xi'}^{(j)}) \\ -{C_2}^{(j)} + {C_4}^{(j)} \cdot \Omega^{(j)}), $$

and thus

$$ {\xi'}^{(j)} = (\xi \cdot {C_4}^{(j)} + ({C_3}^{(j)} \cdot ({C_0}^{(j)} + {C_1}^{(j)} \cdot \Omega^{(j)}) \\ +{C_2}^{(j)} - {C_4}^{(j)} \cdot \Omega^{(j)}))/{C_3}^{(j)}. $$

We construct two function H_V and H_{ros} which behave as random oracles as follows:

- $H_V(Q)$: If $T_V[Q] \neq \perp$, return $T_V[Q]$. Otherwise, parse $\alpha, \beta_1, \beta_2, \text{rnd} \leftarrow Q$, make $\mathcal{V}1(\alpha, \beta_1, \beta_2, \text{rnd})$ query to generate ξ', store $T_V[Q] := \xi'$ and return ξ'.
- $H_{ros}({C_4}^j, a\vec{u}x_j)$: Return ${C_3}^{(j)} \cdot H_V(a\vec{u}x) - ({C_3}^{(j)} \cdot ({C_0}^{(j)} + {C_1}^{(j)} \cdot \Omega^{(j)}) + {C_2}^{(j)} - {C_4}^{(j)} \cdot \Omega^{(j)})$,

where $a\vec{u}x_j = (g, h, y, z, \{a_k, b_{1,k}, b_{2,k}, \text{rnd}_k\}_{k \in [\ell]})$.

It holds that

$$ H_{ros}({C_4}^{(1)}, a\vec{u}x_1) = {C_4}^{(1)} \cdot \xi $$

$$ H_{ros}({C_4}^{(2)}, a\vec{u}x_2) = {C_4}^{(2)} \cdot \xi. $$

This is the one-dimensional ROS (i.e., ROS with $\ell = 1$) problem [33], which is information-theoretically hard and holds with a probability at most $\frac{\binom{q_h}{2}+1}{q}$, where q_h is the number of queries $\mathcal{A}$ makes to H_{ros}. Therefore, we assume that E_4 occurs with probability at most $\frac{\binom{q_h}{2}+1}{q}$. □

By the claim, we have that $\mathbf{Adv}^{\mathbf{G}_3}(\mathcal{A}) \geq \mathbf{Adv}^{\mathbf{G}_2}(\mathcal{A}) - \frac{\binom{q_h}{2}+1}{q}$. Thus, we assume that $\mathcal{A}$ wins.

Simulating $\mathbf{G}_3$: We simulate $\mathbf{G}_3$ by simulating the oracles $\mathcal{P}_1$,$\mathcal{P}_2$,$\mathcal{V}_1$, and $\mathcal{V}_2$. While the simulation of $\mathcal{V}_1$ and $\mathcal{V}_2$ is done as in $\mathsf{ID}_{\mathsf{ACL}}$ (see fig. 4), the simulation of $\mathcal{P}_1$ and $\mathcal{P}_2$ can be done in two ways as follows. Given a DLP challenge U, the reduction $\mathcal{B}$ flips a coin and does the following:

(1) On heads, it simulates the prover using the simulator C.2.2. Instead of sampling y randomly, it sets $y := U$.
(2) On tails, it uses the simulator C.2.1 to simulate the prover. Instead of sampling h randomly, it sets $h := U$.

Claim 3.7. $\mathbf{Adv}^{\mathsf{DLP}}(\mathcal{B}) \geq \frac{1}{2} \cdot \mathbf{Adv}^{\mathbf{G}_3}(\mathcal{A})$.

Proof. Analogous to claim 3.4. Due to claim 3.1, we can write the preliminary values as

$$ \omega_j'(\vec{c_j^*}) := y_{[\alpha^{(j)}]}, \delta_j'(\vec{d_j^*}) := \frac{g_{[\beta_2^{(j)}]} + x \cdot y_{[\beta_2^{(j)}]}}{g_{[\zeta_2]} + x \cdot y_{[\zeta_2]}} $$

for the special sessions $j = 1, 2$. Combinging these equations with the fact that $\mathcal{A}$ wins $\mathbf{G}_3$ implies that the equation

$$ y_{[\alpha^{(j)}]} \neq \omega_j \vee g_{[\zeta_2]} + x \cdot y_{[\zeta_2]} \neq 0 \wedge \frac{g_{[\beta_2^{(j)}]} + x \cdot y_{[\beta_2^{(j)}]}}{g_{[\zeta_2]} + x \cdot y_{[\zeta_2]}} \neq \delta_j. $$

holds for at least one of the special sessions $j \in \{1, 2\}\}$, from which $\mathcal{B}$ can extract the solution to the discrete logarithm challenge.

Since $\mathcal{B}$ does not know for which session the argument above holds, it attempts to extract the solution from both sessions using the following technique. For simplicity, we omit the index j in the following.

$\mathcal{B}$ extracts the discrete logarithm of U depending on which of the following cases occurs (as the cases are not mutually exclusive, the reduction follows the technique from the first matching case).

Case 1: $y_{[\alpha]} \neq \omega'$: Analogous to case 1 in claim 3.4.
Case 2: $g_{[\zeta_2]} + x \cdot y_{[\zeta_2]} \neq 0 \wedge \frac{g_{[\beta_2]} + x \cdot y_{[\beta_2]}}{g_{[\zeta_2]} + x \cdot y_{[\zeta_2]}} = \delta$: Analogous to case 2 in claim 3.4.

The discrete logarithm can be extracted successfully if the correct simulator is used, which happens with probability $\frac{1}{2}$. Thus, we can extract the DLP solution with probability $\geq \frac{1}{2} \cdot \mathbf{Adv}^{\mathbf{G}_3}(\mathcal{A})$. □

Thus, if the event E_3 occurs, the reduction $\mathcal{B}$ solves the DLP problem. More specifically, $\mathbf{Adv}^{\mathsf{DLP}}(\mathcal{B}) \geq \frac{1}{2} \cdot \mathbf{Adv}^{\mathbf{G}_3}(\mathcal{A}) \geq \frac{1}{2} \cdot \mathbf{Adv}^{\mathbf{G}_2}(\mathcal{A}) - \frac{2 \cdot \binom{q_h}{2}+1}{2 \cdot q} = \frac{1}{2} \cdot \mathbf{Adv}^{\mathbf{G}_1}(\mathcal{A}) - \frac{2 \cdot \binom{q_h}{2}+1}{2 \cdot q} - \frac{1}{2 \cdot q} = \frac{1}{2} \cdot \Pr[\mathsf{E}_3] - \frac{\binom{q_h}{2}+1}{q}$. □

Overall, the reduction $\mathcal{B}$ works as follows: it flips two coins, the first coin to decide which lemma between lemmas 3.3 and 3.5 to simulate, and the second coin to decide which simulator to use. It follows that $\mathbf{Adv}^{\mathsf{DLP}}(\mathcal{B}) \geq \frac{1}{2} \cdot (\frac{1}{2} \cdot \mathbf{Adv}^{\ell\text{-RT}}_{\mathsf{ID}_{\mathsf{ACL}}}(\mathcal{A}) - \frac{9}{2 \cdot q} + \frac{1}{2} \cdot \Pr[\mathsf{E}_3] - \frac{\binom{q_h}{2}+1}{q}) = \frac{1}{2} \cdot (\frac{1}{2} \cdot (\mathbf{Adv}^{\ell\text{-RT}}_{\mathsf{ID}_{\mathsf{ACL}}}(\mathcal{A}) + \Pr[\mathsf{E}_3]) - \frac{2 \cdot \binom{q_h}{2}+11}{2 \cdot q}) = \frac{1}{4} \cdot \mathbf{Adv}^{\ell\text{-OMMIM}}_{\mathsf{ID}_{\mathsf{ACL}}}(\mathcal{A}) - \frac{2 \cdot \binom{q_h}{2}+11}{4 \cdot q}$. □

4 ANONYMOUS CREDENTIALS LIGHT

In this section, we study the security of the blind signature scheme Anonymous Credentials Light (ACL) proposed by Baldimtsi and Lysyanskaya [5]. ACL is an adaption of the blind signature of Abe $\mathsf{BS}_{\mathsf{Abe}}$ [2], which they slightly modified to allow the user to embed (a commitment of) its attributes in the signature. Concretely, they extend the protocol with a registration phase, in which the user commits to its attributes, and they change the way z_1 is computed. Instead of generating z_1 by calling a random oracle on a randomly-generated string rnd, it is generated as $z_1 := g^{\text{rnd}} \cdot C$, where C is a commitment of the user's attributes. The user then blinds z_1 using a blinding factor γ, i.e., $\zeta_1 := z_1^{\gamma}$, and outputs ζ_1 as part of the signature.

For the reader's convenience, we illustrate the scheme as an interactive protocol in fig. 1 and give a full description of the scheme (including the show algorithms) in appendix E.1. We provide a proof sketch of the secure show property of ACL in appendix G.

We refer the reader to [5] for proofs of correctness and blindness of the scheme.

4.1 One-More Unforgeability (OMUF)

Theorem 4.1. *Let $\mathbb{G}$ be a group of prime order q with generator g where* DLP *is (t, ε)-hard and such that* $\mathsf{ID}_{\mathsf{ACL}}$ *is (t, ε')-ℓ-*OMMIM *secure. Then* ACL *is (t, ε'')-one-more unforgeable in the AGM+ROM with*

$$\varepsilon'' = \varepsilon' + 13 \cdot \varepsilon + \frac{22 + 3 \cdot \xi + \ell + 1}{q},$$

where ξ is the number of registration queries the adversary makes.

Proof Overview. We split the proof into two main cases: In the first case, the adversary participates in the registration of commitments, as well as the signing sessions somewhat honestly, i.e., the representations it outputs match which group elements an honest user would use to generate the corresponding values. In this case, we provide a reduction to the OMMIM security of the underlying ID scheme. This reduction simulates the signer using the prover oracles provided by the challenger. It simulates the hash oracle by sending the requests to the verifier of the ID scheme. Finally, it closes verifier sessions using the signatures it obtains from the adversary.

The other case is that the adversary behaves dishonestly in one of the following ways:

- It generates commitments during the registration phase using group elements that should not be used for this purpose in honest runs of the protocol, such as using parts of the public key in the representations of the commitments.
- It generates the value ζ not as a multiplicative blinding of the public key element z.
- It submits a representation of ζ_1 that contains group elements from the public key instead of the commitment parameters $h_0, \ldots, h_n$.

In each of these cases of "dishonest" behaviour, we provide a reduction that solves the discrete logarithm problem DLP.

We provide full proof in appendix E.2.

Corollary 4.2. *Let $\mathbb{G}$ be a group of prime order q with generator g where* DLP *is (t, ε)-hard. Then* ACL *is (t, ε'')-one-more unforgeable in the AGM+ROM with*

$$\varepsilon'' = 17 \cdot \varepsilon + \frac{28 + 3 \cdot \xi + \ell + 1}{q}$$

where ξ is the number of registration queries the adversary makes.

Proof. Follows from theorems 3.1 and 4.1. □

4.2 Application to Abe's Blind Signature Scheme

In this section, we provide a new modular OMUF security proof for $\mathsf{BS}_{\mathsf{Abe}}$. We omit the blindness proof since it is already proved in other works [2, 42].

Theorem 4.3. *If* ACL *is (t, ε)-*OMUF*-secure in the AGM + ROM, then* $\mathsf{BS}_{\mathsf{Abe}}$ *is $(t, \varepsilon + \frac{q_S(q_S+q_{\mathsf{H}_2})}{q})$-*OMUF*-secure in the AGM+ROM where q_S is the number of opened signing sessions and q_{H_2} is the number of queries made to hash oracle H_2 by an adversary against* OMUF*-security of* $\mathsf{BS}_{\mathsf{Abe}}$.

The proof is given in appendix F.

REFERENCES

[1] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. 2002. From Identification to Signatures via the Fiat-Shamir Transform: Minimizing Assumptions for Security and Forward-Security. In *EUROCRYPT 2002 (LNCS, Vol. 2332)*, Lars R. Knudsen (Ed.). Springer, Heidelberg, 418–433. https://doi.org/10.1007/3-540-46035-7_28

[2] Masayuki Abe. 2001. A Secure Three-Move Blind Signature Scheme for Polynomially Many Signatures. In *EUROCRYPT 2001 (LNCS, Vol. 2045)*, Birgit Pfitzmann (Ed.). Springer, Heidelberg, 136–151. https://doi.org/10.1007/3-540-44987-6_9

[3] M Abe and M Ohkubo. 2003. Security of Some Three-move Blind Signature Schemes Reconsidered. In *The 2003 Symposium on Cryptography and Information Security*.

[4] Masayuki Abe and Tatsuaki Okamoto. 2000. Provably Secure Partially Blind Signatures. In *CRYPTO 2000 (LNCS, Vol. 1880)*, Mihir Bellare (Ed.). Springer, Heidelberg, 271–286. https://doi.org/10.1007/3-540-44598-6_17

[5] Foteini Baldimtsi and Anna Lysyanskaya. 2013. Anonymous credentials light. In *ACM CCS 2013*, Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung (Eds.). ACM Press, 1087–1098. https://doi.org/10.1145/2508859.2516687

[6] Foteini Baldimtsi and Anna Lysyanskaya. 2013. On the Security of One-Witness Blind Signature Schemes. In *ASIACRYPT 2013, Part II (LNCS, Vol. 8270)*, Kazue Sako and Palash Sarkar (Eds.). Springer, Heidelberg, 82–99. https://doi.org/10.1007/978-3-642-42045-0_5

[7] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. 2009. Randomizable Proofs and Delegatable Anonymous Credentials. In *CRYPTO 2009 (LNCS, Vol. 5677)*, Shai Halevi (Ed.). Springer, Heidelberg, 108–125. https://doi.org/10.1007/978-3-642-03356-8_7

[8] Mihir Bellare, Marc Fischlin, Shafi Goldwasser, and Silvio Micali. 2001. Identification Protocols Secure against Reset Attacks. In *EUROCRYPT 2001 (LNCS, Vol. 2045)*, Birgit Pfitzmann (Ed.). Springer, Heidelberg, 495–511. https://doi.org/10.1007/3-540-44987-6_30

[9] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. 2003. The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme. *Journal of Cryptology* 16, 3 (June 2003), 185–215. https://doi.org/10.1007/s00145-002-0120-1

[10] Mihir Bellare and Adriana Palacio. 2002. GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks. In *CRYPTO 2002 (LNCS, Vol. 2442)*, Moti Yung (Ed.). Springer, Heidelberg, 162–177. https://doi.org/10.1007/3-540-45708-9_11

[11] Mihir Bellare and Phillip Rogaway. 1993. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM CCS 93*, Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby (Eds.). ACM Press, 62–73. https://doi.org/10.1145/168588.168596

[12] Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. 2021. On the (in)security of ROS. In *EUROCRYPT 2021, Part I (LNCS, Vol. 12696)*, Anne Canteaut and François-Xavier Standaert (Eds.). Springer, Heidelberg, 33–53. https://doi.org/10.1007/978-3-030-77870-5_2

[13] Johannes Blömer, Jan Bobolz, Denis Diemert, and Fabian Eidens. 2019. Updatable Anonymous Credentials and Applications to Incentive Systems. In *ACM CCS 2019*, Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz (Eds.). ACM Press, 1671–1685. https://doi.org/10.1145/3319535.3354223

[14] Alexandra Boldyreva. 2003. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In *PKC 2003 (LNCS, Vol. 2567)*, Yvo Desmedt (Ed.). Springer, Heidelberg, 31–46. https://doi.org/10.1007/3-540-36288-6_3

[15] Stefan Brands. 1994. Untraceable Off-line Cash in Wallets with Observers (Extended Abstract). In *CRYPTO'93 (LNCS, Vol. 773)*, Douglas R. Stinson (Ed.). Springer, Heidelberg, 302–318. https://doi.org/10.1007/3-540-48329-2_26

[16] Jan Camenisch and Anna Lysyanskaya. 2001. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *EUROCRYPT 2001 (LNCS, Vol. 2045)*, Birgit Pfitzmann (Ed.). Springer, Heidelberg, 93–118. https://doi.org/10.1007/3-540-44987-6_7

[17] Jan Camenisch and Anna Lysyanskaya. 2001. An Identity Escrow Scheme with Appointed Verifiers. In *CRYPTO 2001 (LNCS, Vol. 2139)*, Joe Kilian (Ed.). Springer, Heidelberg, 388–407. https://doi.org/10.1007/3-540-44647-8_23

[18] Jan Camenisch and Anna Lysyanskaya. 2004. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *CRYPTO 2004 (LNCS, Vol. 3152)*, Matthew Franklin (Ed.). Springer, Heidelberg, 56–72. https://doi.org/10.1007/978-3-540-28628-8_4

[19] Rutchathon Chairattana-Apirom, Lucjan Hanzlik, Julian Loss, Anna Lysyanskaya, and Benedikt Wagner. 2022. PI-Cut-Choo and Friends: Compact Blind Signatures via Parallel Instance Cut-and-Choose and More. In *CRYPTO 2022, Part III (LNCS, Vol. 13509)*, Yevgeniy Dodis and Thomas Shrimpton (Eds.). Springer, Heidelberg, 3–31. https://doi.org/10.1007/978-3-031-15982-4_1

[20] Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. 2014. Algebraic MACs and Keyed-Verification Anonymous Credentials. In *ACM CCS 2014*, Gail-Joon Ahn, Moti Yung, and Ninghui Li (Eds.). ACM Press, 1205–1216. https://doi.org/10.1145/2660267.2660328

[21] David Chaum. 1982. Blind Signatures for Untraceable Payments. In *CRYPTO'82*, David Chaum, Ronald L. Rivest, and Alan T. Sherman (Eds.). Plenum Press, New York, USA, 199–203.

[22] David Chaum. 1983. Blind Signature System. In *CRYPTO'83*, David Chaum (Ed.). Plenum Press, New York, USA, 153.

[23] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. 1994. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In *CRYPTO'94 (LNCS, Vol. 839)*, Yvo Desmedt (Ed.). Springer, Heidelberg, 174–187. https://doi.org/10.1007/3-540-48658-5_19

[24] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. 2018. Privacy Pass: Bypassing Internet Challenges Anonymously. *PoPETs* 2018, 3 (July 2018), 164–180. https://doi.org/10.1515/popets-2018-0026

[25] Rafaël del Pino and Shuichi Katsumata. 2022. A New Framework for More Efficient Round-Optimal Lattice-Based (Partially) Blind Signature via Trapdoor Sampling. In *CRYPTO 2022, Part II (LNCS, Vol. 13508)*, Yevgeniy Dodis and Thomas Shrimpton (Eds.). Springer, Heidelberg, 306–336. https://doi.org/10.1007/978-3-031-15979-4_11

[26] J. Doerner, Y. Kondi, E. Lee, a. shelat, and L. Tyner. 2023. Threshold BBS+ Signatures for Distributed Anonymous Credential Issuance. In *2023 2023 IEEE Symposium on Security and Privacy (SP) (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 2095–2111. https://doi.org/10.1109/SP46215.2023.00120

[27] Uriel Feige, Amos Fiat, and Adi Shamir. 1988. Zero-Knowledge Proofs of Identity. *Journal of Cryptology* 1, 2 (June 1988), 77–94. https://doi.org/10.1007/BF02351717

[28] Amos Fiat and Adi Shamir. 1987. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO'86 (LNCS, Vol. 263)*, Andrew M. Odlyzko (Ed.). Springer, Heidelberg, 186–194. https://doi.org/10.1007/3-540-47721-7_12

[29] Marc Fischlin. 2006. Round-Optimal Composable Blind Signatures in the Common Reference String Model. In *CRYPTO 2006 (LNCS, Vol. 4117)*, Cynthia Dwork (Ed.). Springer, Heidelberg, 60–77. https://doi.org/10.1007/11818175_4

[30] Marc Fischlin and Dominique Schröder. 2010. On the Impossibility of Three-Move Blind Signature Schemes. In *EUROCRYPT 2010 (LNCS, Vol. 6110)*, Henri Gilbert (Ed.). Springer, Heidelberg, 197–215. https://doi.org/10.1007/978-3-642-13190-5_10

[31] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. 2015. Practical Round-Optimal Blind Signatures in the Standard Model. In *CRYPTO 2015, Part II (LNCS, Vol. 9216)*, Rosario Gennaro and Matthew J. B. Robshaw (Eds.). Springer, Heidelberg, 233–253. https://doi.org/10.1007/978-3-662-48000-7_12

[32] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. 2018. The Algebraic Group Model and its Applications. In *CRYPTO 2018, Part II (LNCS, Vol. 10992)*, Hovav Shacham and Alexandra Boldyreva (Eds.). Springer, Heidelberg, 33–62. https://doi.org/10.1007/978-3-319-96881-0_2

[33] Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. 2020. Blind Schnorr Signatures and Signed ElGamal Encryption in the Algebraic Group Model. In *EUROCRYPT 2020, Part II (LNCS, Vol. 12106)*, Anne Canteaut and Yuval Ishai (Eds.). Springer, Heidelberg, 63–95. https://doi.org/10.1007/978-3-030-45724-2_3

[34] Sanjam Garg and Divya Gupta. 2014. Efficient Round Optimal Blind Signatures. In *EUROCRYPT 2014 (LNCS, Vol. 8441)*, Phong Q. Nguyen and Elisabeth Oswald (Eds.). Springer, Heidelberg, 477–495. https://doi.org/10.1007/978-3-642-55220-5_27

[35] Sanjam Garg, Vanishree Rao, Amit Sahai, Dominique Schröder, and Dominique Unruh. 2011. Round Optimal Blind Signatures. In *CRYPTO 2011 (LNCS, Vol. 6841)*, Phillip Rogaway (Ed.). Springer, Heidelberg, 630–648. https://doi.org/10.1007/978-3-642-22792-9_36

[36] Rosario Gennaro. 2004. Multi-trapdoor Commitments and Their Applications to Proofs of Knowledge Secure Under Concurrent Man-in-the-Middle Attacks. In *CRYPTO 2004 (LNCS, Vol. 3152)*, Matthew Franklin (Ed.). Springer, Heidelberg, 220–236. https://doi.org/10.1007/978-3-540-28628-8_14

[37] Lucjan Hanzlik, Julian Loss, and Benedikt Wagner. 2023. Rai-Choo! Evolving Blind Signatures to the Next Level. In *Advances in Cryptology – EUROCRYPT 2023*, Carmit Hazay and Martijn Stam (Eds.). Springer Nature Switzerland, Cham, 753–783.

[38] Eduard Hauck, Eike Kiltz, and Julian Loss. 2019. A Modular Treatment of Blind Signatures from Identification Schemes. In *EUROCRYPT 2019, Part III (LNCS, Vol. 11478)*, Yuval Ishai and Vincent Rijmen (Eds.). Springer, Heidelberg, 345–375. https://doi.org/10.1007/978-3-030-17659-4_12

[39] Eduard Hauck, Eike Kiltz, Julian Loss, and Ngoc Khanh Nguyen. 2020. Lattice-Based Blind Signatures, Revisited. In *CRYPTO 2020, Part II (LNCS, Vol. 12171)*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer, Heidelberg, 500–529. https://doi.org/10.1007/978-3-030-56880-1_18

[40] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. 2017. TumbleBit: An Untrusted Bitcoin-Compatible Anonymous Payment Hub. In *NDSS 2017*. The Internet Society.

[41] Ari Juels, Michael Luby, and Rafail Ostrovsky. 1997. Security of Blind Digital Signatures (Extended Abstract). In *CRYPTO'97 (LNCS, Vol. 1294)*, Burton S. Kaliski Jr. (Ed.). Springer, Heidelberg, 150–164. https://doi.org/10.1007/BFb0052233

[42] Julia Kastner, Julian Loss, and Jiayu Xu. 2022. On Pairing-Free Blind Signature Schemes in the Algebraic Group Model. In *PKC 2022, Part II (LNCS, Vol. 13178)*, Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe (Eds.). Springer, Heidelberg, 468–497. https://doi.org/10.1007/978-3-030-97131-1_16

[43] Jonathan Katz, Julian Loss, and Michael Rosenberg. 2021. Boosting the Security of Blind Signature Schemes. In *ASIACRYPT 2021, Part IV (LNCS, Vol. 13093)*, Mehdi Tibouchi and Huaxiong Wang (Eds.). Springer, Heidelberg, 468–492. https://doi.org/10.1007/978-3-030-92068-5_16

[44] Ben Kreuter, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. 2020. Anonymous Tokens with Private Metadata Bit. In *CRYPTO 2020, Part I (LNCS, Vol. 12170)*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer, Heidelberg, 308–336. https://doi.org/10.1007/978-3-030-56784-2_11

[45] Tatsuaki Okamoto. 2006. Efficient Blind and Partially Blind Signatures Without Random Oracles. In *TCC 2006 (LNCS, Vol. 3876)*, Shai Halevi and Tal Rabin (Eds.). Springer, Heidelberg, 80–99. https://doi.org/10.1007/11681878_5

[46] Rafael Pass. 2011. Limits of provable security from standard assumptions. In *43rd ACM STOC*, Lance Fortnow and Salil P. Vadhan (Eds.). ACM Press, 109–118. https://doi.org/10.1145/1993636.1993652

[47] Torben P. Pedersen. 1992. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *CRYPTO'91 (LNCS, Vol. 576)*, Joan Feigenbaum (Ed.). Springer, Heidelberg, 129–140. https://doi.org/10.1007/3-540-46766-1_9

[48] David Pointcheval. 2001. Self-Scrambling Anonymizers. In *FC 2000 (LNCS, Vol. 1962)*, Yair Frankel (Ed.). Springer, Heidelberg, 259–275.

[49] David Pointcheval and Jacques Stern. 1996. Provably Secure Blind Signature Schemes. In *ASIACRYPT'96 (LNCS, Vol. 1163)*, Kwangjo Kim and Tsutomu Matsumoto (Eds.). Springer, Heidelberg, 252–265. https://doi.org/10.1007/BFb0034852

[50] David Pointcheval and Jacques Stern. 1997. New Blind Signatures Equivalent to Factorization (Extended Abstract). In *ACM CCS 97*, Richard Graveman, Philippe A. Janson, Clifford Neuman, and Li Gong (Eds.). ACM Press, 92–99. https://doi.org/10.1145/266420.266440

[51] David Pointcheval and Jacques Stern. 2000. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology* 13, 3 (June 2000), 361–396. https://doi.org/10.1007/s001450010003

[52] Claus-Peter Schnorr. 2001. Security of Blind Discrete Log Signatures against Interactive Attacks. In *ICICS 01 (LNCS, Vol. 2229)*, Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou (Eds.). Springer, Heidelberg, 1–12.

[53] Victor Shoup. 1997. Lower Bounds for Discrete Logarithms and Related Problems. In *EUROCRYPT'97 (LNCS, Vol. 1233)*, Walter Fumy (Ed.). Springer, Heidelberg, 256–266. https://doi.org/10.1007/3-540-69053-0_18

[54] Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, Sarah Meiklejohn, and George Danezis. 2019. Coconut: Threshold Issuance Selective Disclosure Credentials with Applications to Distributed Ledgers. In *NDSS 2019*. The Internet Society.

[55] Stefano Tessaro and Chenzhi Zhu. 2022. Short Pairing-Free Blind Signatures with Exponential Security. In *EUROCRYPT 2022, Part II (LNCS, Vol. 13276)*, Orr Dunkelman and Stefan Dziembowski (Eds.). Springer, Heidelberg, 782–811. https://doi.org/10.1007/978-3-031-07085-3_27

[56] David Wagner. 2002. A Generalized Birthday Problem. In *CRYPTO 2002 (LNCS, Vol. 2442)*, Moti Yung (Ed.). Springer, Heidelberg, 288–303. https://doi.org/10.1007/3-540-45708-9_19

A ADDITIONAL PRELIMINARIES

A.1 Commitment Schemes

Commitment schemes are cryptographic protocols between a sender and a receiver that allow the sender to commit to a message while keeping it secret from the receiver. The receiver cannot inspect the message before the sender permits them to open the commitment, and the sender cannot send a different message from the one they committed to.

Let Commit be a deterministic algorithm that takes as input a message m and a random r and outputs a commitment C. A commitment scheme consists of the following phases:

- **Setup.** A public parameter pp is generated.
- **Committing.** The sender commits to a message m by sampling a random r, generating $C \leftarrow \mathsf{Commit}(m; r)$, and sending C to the receiver.
- **Opening.** The sender allows the receiver to open the commitment C by sending them (m, r). The receiver checks if $C = \mathsf{Commit}(m; r)$.

Definition A.1 (Perfectly Hiding). A scheme Comm is perfectly hiding if for all pairs of messages m_0, m_1 and all commitments C in the span of Commit it holds that

$$\Pr_r[\mathsf{Commit}(m_0; r) = C] = \Pr_r[\mathsf{Commit}(m_1; r) = C].$$

Definition A.2 (Binding). Let Comm be a commitment scheme with a committing algorithm Commit, and let $\mathcal{A}$ an adversary. We define the game Bind as follows:

- **Online Phase.** Generate a public parameter pp, and invoke $\mathcal{A}(pp)$.
- **Output Determination.** Output 1 iff $\mathcal{A}$ outputs two (m_1, r_1) and (m_2, r_2) with $m_1 \neq m_2$ and $C = \mathsf{Commit}(m_1, r_1) = \mathsf{Commit}(m_2, r_2)$, 0 otherwise.

We define the advantage of $\mathcal{A}$ in winning the game Bind against the scheme Comm as

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

We say that Comm is binding if for all adversary $\mathcal{A}$, $\mathbf{Adv}^{\mathsf{Bind}}_{\mathsf{Comm}}(\mathcal{A})$ is negligible.

A.1.1 The Generalized Pedersen Commitment. We recall the Generalized Pedersen Commitment[47], with which a sender can commit to multiple messages $(m_1, \ldots, m_n)$ for $n \in \mathbb{Z}^+$. The Generalized Pedersen Commitment over a group $\mathbb{G}$ of prime order q with generator g is defined as follows:

- **Setup.** Sample random values $a_0, a_1, \ldots, a_n \leftarrow_\$ \mathbb{Z}_q$, define $h_i \coloneqq g^{a_i}$ for $i \in [n] \cup \{0\}$, set and output $\mathsf{pp} \coloneqq (h_0, h_1, \ldots, h_n)$.
- **Committing.** To commit to messages $(m_1, \ldots, m_n)$, where $m_i \in \mathbb{Z}_q$ for $i \in [n]$, the sender samples a random value $r \leftarrow_\$ \mathbb{Z}_q$, computes $C \coloneqq h_0^r \prod_{i=1}^n h_i^{m_i}$ and sends C to the receiver.
- **Opening.** To open a commitment C, the sender sends $(r, m_1, \ldots, m_n)$ to the receiver. The receiver verifies the opening by checking $C \stackrel{?}{=} h_0^r \prod_{i=1}^n h_i^{m_i}$

A.1.2 Proof of Knowledge of an Opening for Generalized Pedersen Commitments. We recall the following (folklore) sigma protocol for proving knowledge of an opening of a Generalized Pedersen commitment. As above, let $\mathsf{pp} = (h_0, \ldots, h_n)$ be the public parameters. Let $(m_1, \ldots, m_n)$ be a message vector, and let r be the randomness used during committing. To prove knowledge of $r, (m_1, \ldots, m_n)$, do the following: Pick random values $r_0, \ldots, r_n \leftarrow_\$ \mathbb{Z}_q$. The prover sends the first message $M = \prod_{i=0}^n h_i^{r_i}$. The verifier sends a challenge c. The prover computes $s_0 = r_0 - c \cdot r$, $s_i = r_i - c \cdot m_i$ for $i \in [n]$ and outputs the last message $(s_0, \ldots, s_n)$. To verify, check that $C^c \prod_i h_i^{s_i} = M$. It is straightforward to turn this proof into a non-interactive one using the Fiat-Shamir heuristic [28].

B DEFERRED PROOFS FROM SECTION 3

B.1 Proof of Claim 3.1

We recall the claim

Claim 3.1. *Let $\alpha, \beta_1, \zeta_1, \beta_2, \zeta_2$ be the group elements from the special sessions in lemma 3.3, and let $\alpha', \beta_1', \beta_2', \zeta_1'$, and ζ_2' be the values described in definition 3.4. It holds that*

(1) $y_{[\alpha]} = y_{[\alpha']} + \sum_{i=1}^k c_i^* \cdot \overline{a_i^*(\alpha)}$

(2) $h_{[\beta_2]} = h_{[\beta_2']} + \sum_{i=1}^k \overline{b_{2,i}^*(\beta_2)} \cdot d_i^* \cdot (v_{0,1} - v_{1,i}^*)$

(3) $h_{[\zeta_2]} = h_{[\zeta_2']} + \sum_{i=1}^k \overline{b_{2,i}^*(\zeta_2)} \cdot d_i^* \cdot (v_{0,1} - v_{1,i}^*)$

where k is the number of prover sessions linked to the special session.

Proof. Since these equalities are similar, we only prove the claim for equalities (1) and (2). For simplicity, we assume w.l.o.g. $k = 1$.

(1) Per definition 3.4, we have that

$$y_{[\alpha']} = y_{[\alpha]} - y_{[a_1^*]} \cdot \overline{a_1^*(\alpha)},$$

and since $a_1^* = g^{r_1^*} \cdot y^{c_1^*}$, we have

$$y_{[\alpha']} = y_{[\alpha]} - c_1^* \cdot \overline{a_1^*(\alpha)} = y_{[\alpha]} - \sum_{i=1}^k c_i^* \cdot \overline{a_i^*(\alpha)}.$$

Thus, we have that

$$y_{[\alpha']} + \sum_{i=1}^k c_i^* \cdot \overline{a_i^*(\alpha)} = y_{[\alpha]} - \sum_{i=1}^k c_i^* \cdot \overline{a_i^*(\alpha)} + \sum_{i=1}^k c_i^* \cdot \overline{a_i^*(\alpha)} = y_{[\alpha]}.$$

(2) Per definition 3.4, we have that

$$h_{[\beta_1']} = h_{[\beta_1]} - h_{[b_{2,1}^*]} \cdot \overline{b_{2,1}^*(\beta_1)},$$

and since $b_{2,1}^* = h^{u_{2,1}^*} = h^{r_1^*} \cdot z_2^{c_1^*} = h^{s_{1,2}^* + v_{0,2} \cdot d_1^*} \cdot g^{d_1^* \cdot (v_{0,1} - v_{1,1}^*)}$, we have

$$h_{[b_{2,1}^*]} = s_{1,2}^* + v_{0,2} \cdot d_1^* = u_{2,1}^* - d_1^* \cdot (v_{0,1} - v_{1,1}^*),$$

and

$$h_{[\beta']} + \sum_{i=1}^k \overline{b_{2,i}^*(\beta_1)} \cdot (u_{2,i}^* - d_i^* \cdot (v_{0,1} - v_{1,i}^*))$$

$$h_{[\beta']} + \overline{b_{2,1}^*(\beta_1)} \cdot (u_{2,1}^* - d_1^* \cdot (v_{0,1} - v_{1,1}^*)) = h_{[\beta_1]}.$$

(3) the correctness of the equalities (3), (4), and (5) can be proven analogously.

□

B.2 Proof of Claim 3.3

Claim 3.3. $\Pr[\mathsf{E}_2] \leq \frac{8}{q}$.

Proof. Assume $\mathcal{A}$ links the special session to k prover sessions, thus we have that $\overrightarrow{c^*} = (c_1^*, \ldots, c_k^*)$ and $\overrightarrow{d^*} = (d_1^*, \ldots, d_k^*)$. We order these prover sessions by the time they are opened. To further simplify the proof, we compute $\mathcal{A}$'s advantage before it terminates and while the k_{th} linked prover session is still open. If $\mathcal{A}$ wants to close any of the first $k-1$ prover sessions, we assume that it has already closed these sessions. Consequently, the field elements c^* and d^* of the first $k-1$ prover sessions linked to the special session $(c_1^*, \ldots, c_{k-1}^*)$ and $(d_1^*, \ldots, d_{k-1}^*)$ are constants. Therefore, we can rewrite the preliminary values as follows: we first define the constants $C_0' = \sum_{i=1}^{k-1} c_i^* \cdot \overline{a_i^*(\alpha)}$, $C_3' = \sum_{i=1}^{k-1} \overline{b_{2,i}^*(\beta_2)} \cdot d_i^* \cdot (v_{0,1} - \mathrm{rnd}_i^*)$, and write

$$\omega'(\overrightarrow{c^*}) = y_{[\alpha']} + C_0' + c_k^* \cdot \overline{a_k^*},$$

$$\delta''(\overrightarrow{d^*}) = \frac{g_{[\beta_2']} + x \cdot y_{[\beta_2']} + C_3' + \overline{b_{2,k}^*(\beta_2)} \cdot d_k^* \cdot (v_{0,1} - \mathrm{rnd}_k^*)}{g_{[\zeta_2']} + x \cdot y_{[\zeta_2']}}.$$

Recall that in order to provoke E_2, $\mathcal{A}$ has to ensure that $\omega'(\overrightarrow{c^*}) + \delta''(\overrightarrow{d^*}) = \varepsilon$.

We want to apply claim D.1 where E_2 is equivalent to E_x with the following assignments of variables $\mathcal{X} := \omega', \mathcal{Y} := \delta'', \xi := \varepsilon_k, \xi' := \varepsilon_k, \Delta := d_k^*, \Omega := c_k^*, C_0 := y_{[\alpha']} + C_0', C_1 := \overline{a_k^*(\alpha)}, C_2 := g_{[\beta_2']} + x \cdot y_{[\beta_2']} + C_3', C_3 := g_{[\zeta_2']} + x \cdot y_{[\zeta_2']} = g_{[\zeta']} + x \cdot y_{[\zeta']} - g_{[\zeta_1']} + x \cdot y_{[\zeta_1']} = v_{0,1} - \mathrm{rnd}$ where $\mathrm{rnd} = \mathrm{dlog}_g \zeta_1'$, $C_4 := (v_{0,1} - \mathrm{rnd}_k^*) \cdot \overline{b_{2,k}^*(\beta_2)}$, and $C_5 := 0$.

As $C_5 = 0$, we need to show that $C_4/C_3 \neq C_1$. To this end, consider the following cases:

(1) $\overline{b_{2,k}^*(\beta_2)} = \overline{a_k^*(\alpha)}$: In this case, we would get $v_{0,1} - \mathrm{rnd}_k^* = v_{0,1} - \mathrm{rnd}$, . This, however, violates "freshness" of the tag ζ_1, because $\zeta_1 = z_{1,k^*}$ (except with probability $\frac{1}{q}$ per the randomness of $v_{0,1}$).

(2) otherwise, we obtain $v_{0,1} = \frac{\overline{b_{2,k}^*(\beta_2)} \cdot \mathrm{rnd}_{k^*} - \overline{a_k^*(\alpha)} \cdot \mathrm{rnd}}{\overline{b_{2,k}^*(\beta_2)} - \overline{a_k^*(\alpha)}}$ which occurs with probability $\frac{1}{q}$ as $v_{0,1}$ is chosen uniformly at random and is information-theoretically hidden from the adversary.

Thus, by claim D.1, this occurs with a probability of at most $\frac{8}{q}$. □

C SIMULATORS

The underlying idea of both schemes BS_ACL and ID_ACL uses the OR-Proof technique [23]. In BS_ACL, the signer can be simulated in two ways. The signer can either use its knowledge of the discrete logarithm of y (the y-side witness) to run the protocol or its knowledge of the discrete logarithm of $h, z,$ and z_1 (the z-side witness). Similarly, the prover in ID_ACL can either be simulated using the knowledge of the discrete logarithm of y or the knowledge of the discrete logarithm of $h, z,$ and z_1. We exploit this fact in our proof to embed the discrete logarithm instance U in one witness and sample the other witness with a known discrete logarithm. Using the witness with the known discrete logarithm, we simulate the protocol. This enables us to extract the discrete logarithm of the witness, in which U is embedded, hence computing the discrete logarithm of U and winning the DLP game. For a group $\mathbb{G}$ of order q and a generator g, we describe in the following the y-side simulator and z-side simulator for both schemes BS_ACL and ID_ACL.

C.1 Simulators for BS_ACL

C.1.1 The y-side Simulator for BS_ACL.

- **Initialization.** Sample $x \leftarrow_\$ \mathbb{Z}_q$ and $z, h \leftarrow_\$ \mathbb{G}$, compute $y := g^x$ and set $pk := (g, h, y, z), sk := x$. Sample $w_0, \ldots, w_n \leftarrow_\$ \mathbb{Z}_q$. Set $H_{\mathsf{pp}}(i) = h_i = g^{w_i}$.
- **Online Phase.** The simulator simulates Registration, $\mathcal{S}_1$ and $\mathcal{S}_2$ as follows:
  - **Registration:** On input of C, π, check that π is a valid proof of knowledge and then save C in a list of registered commitments.
  - $\mathcal{S}_1(C)$: Check that C was previously registered. Sample a fresh session identifier sid, a random $\mathrm{rnd}_{\mathsf{sid}} \leftarrow_\$ \mathbb{Z}_q$, and field elements $u_{\mathsf{sid}}, d_{\mathsf{sid}}, s_{1,\mathsf{sid}}, s_{2,\mathsf{sid}} \leftarrow_\$ \mathbb{Z}_q$, generate $z_{1,\mathsf{sid}} := C g^{\mathrm{rnd}_{\mathsf{sid}}}$, compute $z_{2,\mathsf{sid}} := z/z_{1,\mathsf{sid}}$, $a_{\mathsf{sid}} := g^{u_{\mathsf{sid}}}$, $b_{1,\mathsf{sid}} := g^{s_{1,\mathsf{sid}}} \cdot z_1^{d_{\mathsf{sid}}}, b_{2,\mathsf{sid}} := h^{s_{2,\mathsf{sid}}} \cdot z_2^{d_{\mathsf{sid}}}$, and return $a_{\mathsf{sid}}, b_{1,\mathsf{sid}}, b_{2,\mathsf{sid}}, \mathrm{rnd}_{\mathsf{sid}}$.
  - $\mathcal{S}_2(\mathsf{sid}, e_{\mathsf{sid}})$: compute $c_{\mathsf{sid}} := e_{\mathsf{sid}} - d_{\mathsf{sid}}$, $r_{\mathsf{sid}} := u_{\mathsf{sid}} - c_{\mathsf{sid}} \cdot x$, and return $c_{\mathsf{sid}}, d_{\mathsf{sid}}, r_{\mathsf{sid}}, s_{1,\mathsf{sid}}, s_{2,\mathsf{sid}}$.
- **Finalization.** To fix the values $c_{\mathsf{sid}}, r_{\mathsf{sid}}$ for sessions that have not been closed by the adversary, the simulator chooses values $e_{\mathsf{sid}} \leftarrow_\$ \mathbb{Z}_q$ for these sessions and runs $\mathcal{S}_2(\mathsf{sid}, e_{\mathsf{sid}})$ for these unclosed sessions.

C.1.2 The z-side Simulator for BS_ACL.

- **Initialization.** Sample $w \leftarrow_\$ \mathbb{Z}_q$ and $y \leftarrow_\$ \mathbb{G}$, compute $h := g^w, z := \mathsf{H}_1(\mathbb{G}, g, q, y, h)$, and set $pk := (g, h, y, z)$, $sk := w$.
  Sample $w_0, \ldots, w_n \leftarrow_\$ \mathbb{Z}_q$. Set $H_{\mathsf{pp}}(i) = h_i = g^{w_i}$.
- **Online Phase.** The simulator simulates Registration, $\mathcal{S}_1$ and $\mathcal{S}_2$ as follows:
  - **Registration:** On input of C, π, check that π is a valid proof of knowledge. Compute $\mathrm{dlog}_g C$ from the representation submitted in the proof hash query.[1]
  - $\mathcal{S}_1(C)$: Check that C was previously registered. Generate a fresh session identifier sid, sample $\mathrm{rnd}_{\mathsf{sid}} \leftarrow_\$ \mathbb{Z}_q$ and $c_{\mathsf{sid}}, r_{\mathsf{sid}}, u_{1,\mathsf{sid}}, u_{2,\mathsf{sid}} \leftarrow_\$ \mathbb{Z}_q$, compute $z_{1,\mathsf{sid}} := C^{\mathrm{rnd}_{\mathsf{sid}}}$, $z_{2,\mathsf{sid}} := z/z_{1,\mathsf{sid}}$, $a_{\mathsf{sid}} := g^{r_{\mathsf{sid}}} \cdot y^{c_{\mathsf{sid}}}$, $b_{1,\mathsf{sid}} := g^{u_{1,\mathsf{sid}}}, b_{2,\mathsf{sid}} := h^{u_{2,\mathsf{sid}}}$, define $d_{\mathsf{sid}} := 0, s_{1,\mathsf{sid}} := u_{1,\mathsf{sid}}$, $s_{2,\mathsf{sid}} := u_{2,\mathsf{sid}}$ and return $a_{\mathsf{sid}}, b_{1,\mathsf{sid}}, b_{2,\mathsf{sid}}, \mathrm{rnd}_{\mathsf{sid}}$.
  - $\mathcal{S}_2(\mathsf{sid}, e_{\mathsf{sid}})$: compute $s_{1,\mathsf{sid}} := u_{1,\mathsf{sid}} - d_{\mathsf{sid}} \cdot v_{1,\mathsf{sid}}$, $s_{2,\mathsf{sid}} := u_{2,\mathsf{sid}} - d_{\mathsf{sid}} \cdot w_{2,\mathsf{sid}}$, and $d_{\mathsf{sid}} := e_{\mathsf{sid}} - c_{\mathsf{sid}}$, where $v_{1,\mathsf{sid}} := \mathrm{dlog}_g z_{1,\mathsf{sid}} = \mathrm{dlog}_g C + \mathrm{rnd}_{\mathsf{sid}}$, and $w_{2,\mathsf{sid}} := \frac{\mathrm{dlog}_g z - \mathrm{dlog}_g z_{1,\mathsf{sid}}}{w}$, and return $c_{\mathsf{sid}}, d_{\mathsf{sid}}, r_{\mathsf{sid}}, s_{1,\mathsf{sid}}, s_{2,\mathsf{sid}}$.
- **Finalization.** To fix the $d_{\mathsf{sid}}, s_{1,\mathsf{sid}}, s_{2,\mathsf{sid}}$ for sessions that are not closed by the adversary, the simulator chooses values $e_{\mathsf{sid}} \leftarrow_\$ \mathbb{Z}_q$ for these sessions and runs $\mathcal{S}_2(\mathsf{sid}, e_{\mathsf{sid}})$ for these unclosed sessions.

C.2 Simulators for ID_ACL

C.2.1 The y-side Simulator for ID_ACL.

- **Initialization.** Sample $x, v_{0,1}, v_{0,2} \leftarrow_\$ \mathbb{Z}_q$ and $h \leftarrow_\$ \mathbb{G}$, compute $y := g^x, z := g^{v_{0,1}} \cdot h^{v_{0,2}}$, and set $pk := (g, h, y, z), sk := x$.
- **Online Phase.** The simulator simulates both $\mathcal{P}_1$ and $\mathcal{P}_2$ as follows:
  - $\mathcal{P}_1()$: sample a fresh session identifier sid and field elements $u_{\mathsf{sid}}, d_{\mathsf{sid}}, s_{1,\mathsf{sid}}, s_{2,\mathsf{sid}}, \mathrm{rnd}_{\mathsf{sid}} \leftarrow_\$ \mathbb{Z}_q$, compute $z_{1,\mathsf{sid}} := g^{\mathrm{rnd}_{\mathsf{sid}}}, z_{2,\mathsf{sid}} := z/z_{1,\mathsf{sid}}, a_{\mathsf{sid}} := g^{u_{\mathsf{sid}}}, b_{1,\mathsf{sid}} := g^{s_{1,\mathsf{sid}}} \cdot z_1^{d_{\mathsf{sid}}}, b_{2,\mathsf{sid}} := h^{s_{2,\mathsf{sid}}} \cdot z_2^{d_{\mathsf{sid}}}$, and return $a_{\mathsf{sid}}, b_{1,\mathsf{sid}}, b_{2,\mathsf{sid}}, \mathrm{rnd}_{\mathsf{sid}}$.
  - $\mathcal{P}_2(\mathsf{sid}, \varepsilon_{\mathsf{sid}})$: compute $c_{\mathsf{sid}} := \varepsilon_{\mathsf{sid}} - d_{\mathsf{sid}}$ and $r_{\mathsf{sid}} := u_{\mathsf{sid}} - c_{\mathsf{sid}} \cdot x$, and return $c_{\mathsf{sid}}, d_{\mathsf{sid}}, r_{\mathsf{sid}}, s_{1,\mathsf{sid}}, s_{2,\mathsf{sid}}$.
- **Finalization.** To fix the values $c_{\mathsf{sid}}, r_{\mathsf{sid}}$ for sessions that have not been closed by the adversary, the simulator chooses values $e_{\mathsf{sid}} \leftarrow_\$ \mathbb{Z}_q$ for these sessions and runs $\mathcal{P}_2(\mathsf{sid}, \varepsilon_{\mathsf{sid}})$ for these unclosed sessions.

[1] This only works once we rule out the usage of group elements with unknown discrete logarithms or if all discrete logarithms are known to the simulator.

C.2.2 The z-side Simulator for $\mathsf{ID}_{\mathsf{ACL}}$.

- **Initialization.** Samples $w, v_{0,1}, v_{0,2} \leftarrow_\$ \mathbb{Z}_q$ and $y \leftarrow_\$ \mathbb{G}$, compute $h := g^w, z := g^{v_{0,1}} \cdot h^{v_{0,2}}$. Additionally, it sets $pk := (g, h, y, z)$, and $sk := w$.
- **Online Phase.** It simulates $\mathcal{P}_1$ and $\mathcal{P}_2$ as follows:
  $\mathcal{P}_1()$: generate a fresh session identifier sid, sample $c_{\text{sid}}, r_{\text{sid}}, u_{1,\text{sid}}, u_{2,\text{sid}}, \text{rnd}_{\text{sid}} \leftarrow_\$ \mathbb{Z}_q$, compute $z_{1,\text{sid}} := g^{\text{rnd}_{\text{sid}}}, z_{2,\text{sid}} := z/z_{1,\text{sid}}, a_{\text{sid}} := g^{r_{\text{sid}}} \cdot y^{c_{\text{sid}}}, b_{1,\text{sid}} := g^{u_{1,\text{sid}}}, b_{2,\text{sid}} := h^{u_{2,\text{sid}}}$, and return $a_{\text{sid}}, b_{1,\text{sid}}, b_{2,\text{sid}}, \text{rnd}_{\text{sid}}$.
  $\mathcal{P}_2(\text{sid}, \varepsilon_{\text{sid}})$: compute $s_{1,\text{sid}} := u_{1,\text{sid}} - d_{\text{sid}} \cdot \text{rnd}_{\text{sid}}$, $s_{2,\text{sid}} := u_{2,\text{sid}} - d_{\text{sid}} \cdot w_{2,\text{sid}}$, and $d_{\text{sid}} := \varepsilon_{\text{sid}} - c_{\text{sid}}$, where $w_{2,\text{sid}} := \frac{v_0 - \text{rnd}_{\text{sid}}}{w}$, and $v_0 := \text{dlog}_g z$, and return $c_{\text{sid}}, d_{\text{sid}}, r_{\text{sid}}, s_{1,\text{sid}}, s_{2,\text{sid}}$.
- **Finalization.** To fix the $d_{\text{sid}}, s_{1,\text{sid}}, s_{2,\text{sid}}$ for sessions that are not closed by the adversary, the simulator chooses values $e_{\text{sid}} \leftarrow_\$ \mathbb{Z}_q$ for these sessions and runs $\mathcal{S}_2(\text{sid}, e_{\text{sid}})$ for these unclosed sessions.

D HELPFUL TECHNICAL CLAIMS

We prove some helpful technical claims. First, we define a generic type of event that can occur in multiple places throughout our proofs.

Definition D.1 (Template Proof). Let $\mathcal{X}: \mathbb{Z}_q \to \mathbb{Z}_q$ and $\mathcal{Y}: \mathbb{Z}_q \to \mathbb{Z}_q$ be two functions defined as

$$\mathcal{X}(\Omega) := C_0 + C_1 \cdot \Omega, \ \mathcal{Y}(\Delta) := \frac{C_2 + C_4 \cdot \Delta}{C_3 + C_5 \cdot \Delta},$$

where the values C_i for $i \in \{0, \ldots, 5\}$ are constants. Furthermore, define $\xi, \xi' \in \mathbb{Z}_q$ such that the following conditions hold:

(1) The values C_i for $i \in \{0, \ldots, 5\}$ get fixed first, then, the value ξ' gets sampled uniformly at random from $\mathbb{Z}_q$. Then ξ gets fixed, and then one of the values Ω and Δ is sampled uniformly at random, while the other value is chosen such that $\Omega + \Delta = \xi$.
(2) $(C_3 \neq 0 \wedge C_4/C_3 \neq C_1) \vee C_5 \neq 0$.

We define the event $\mathsf{E_x}$ as $\mathcal{X}(\Omega) + \mathcal{Y}(\Delta) = \xi'$.

We bound the generalized event with the following claim.

Claim D.1. $\Pr[\mathsf{E_x}] \leq \frac{6}{q}$.

Proof. If $\mathsf{E_x}$ occurs, it must hold that

$$\mathcal{X}(\Omega) + \mathcal{Y}(\Delta) = \xi'. \tag{6}$$

We distinguish two cases of the condition (1) depending on which of the values Ω and Δ is chosen uniformly at random. However, these cases are analogous; therefore, we only prove the claim assuming that Ω is random. By substituting $\mathcal{X}$ and $\mathcal{Y}$ in equation 6, we get

$$C_0 + C_1 \cdot \Omega + \frac{C_2 + C_4 \cdot \Delta}{C_3 + C_5 \cdot \Delta} = \xi'.$$

Per condition (2), the denominator $C_3 + C_5 \cdot \Delta$ is non-zero except with a probility at most $\frac{1}{q}$ due to the randomness of Δ. Thus, we assume in the following that $C_3 + C_5 \cdot \Delta \neq 0$.

Next, we make the following case distinction.

(1) $C_1 = 0$. In this case, the event $\mathsf{E_x}$ occurs iff $(C_0 \cdot C_5 + C_4 - \xi' \cdot C_5) \cdot \Delta - \xi' \cdot C_3 + C_2 + C_0 \cdot C_3 = 0$. As $C_{0,\ldots,5}$ and ξ' are fixed before Δ, we construct the polynomial

$$P(X) := (C_0 \cdot C_5 + C_4 - \xi' \cdot C_5) \cdot X - \xi' \cdot C_3 + C_2 + C_0 \cdot C_3. \tag{7}$$

We distinguish two sub-cases: the first case is that the polynomial P is the zero polynomial. This requires both terms $C_0 \cdot C_5 + C_4 - \xi' \cdot C_5$ and $\xi' \cdot C_3 + C_2 + C_0 \cdot C_3$ to be zero. However, due to condition (2), either $C_3 \neq 0$ or $C_5 \neq 0$. Consequently, ξ' occurs in at least one of these terms, and since ξ' is uniformly random, the probability that both terms are 0 is at most $\frac{1}{q}$. The other case is that P is a non-zero polynomial. Since Δ gets sampled uniformly at random ($\Delta = \xi - \Omega$, and Ω is uniformly random), the probability of $P(\Delta) = 0$ holds is at most $\frac{1}{q}$ per the Schwartz-Zippel lemma. Consequently, the probability that $\mathsf{E_x}$ occurs while $C_1 = 0$ is at most $\frac{2}{q}$. We therefore assume in the following that $C_1 \neq 0$.

(2) $C_1 \neq 0 \wedge C_5 = 0$. As $\Delta = \xi - \Omega$, we have

$$C_0 + C_1 \cdot \Omega + \frac{C_2 + C_4 \cdot (\xi - \Omega)}{C_3} = \xi'.$$

In this case, we can simplify the event to the linear equation $(C_1 - C_4/C_3) \cdot \Omega + C_0 + C_2/C_3 + C_4/C_3 \cdot \xi = \xi'$. Since Ω is the value that is fixed last in this equation, we construct the polynomial

$$P(X) := (C_1 - C_4/C_3) \cdot X + C_0 + C_2/C_3 + C_4/C_3 \cdot \xi - \xi'.$$

Recall that condition (2) requires that in case $C_5 = 0, C_1 - C_4/C_3 \neq 0$ holds; therefore, the polynomial P cannot be the zero polynomial. Since P is non-zero, and Ω is sampled uniformly at random, the probability that $P(\Omega) = 0$ is at most $\frac{1}{q}$ per the Schwartz-Zippel lemma. Thus, the event $\mathsf{E_x}$ occurs with a probability at most $\frac{1}{q}$ if $C_5 = 0$.

(3) $C_1, C_5 \neq 0$. We construct the polynomial

$$P(X) := \mathcal{V}_1 \cdot X^2 + \mathcal{V}_2 \cdot X + \mathcal{V}_3,$$

where $\mathcal{V}_1 = -C_1 \cdot C_5, \mathcal{V}_2 = C_1 \cdot (C_3 + C_5 \cdot \xi) - C_4 - C_5 \cdot (C_0 - \xi')$, $\mathcal{V}_3 = (C_0 - \xi') \cdot (C_3 + C_5 \cdot \xi) + C_2 + C_4 \cdot \xi$. Since $C_1 \neq 0$ and $C_5 \neq 0$, it follows $\mathcal{V}_1 \neq 0$, and thus, P cannot be the zero polynomial. Consequently, since P is not a zero-polynomial, and Ω is sampled uniformly at random, $P(\Omega) = 0$ holds with probability $\frac{2}{q}$ per the Schwartz-Zippel lemma.

In summary, this yields that $\mathsf{E_x}$ occurs with probability at most $\frac{6}{q}$. □

The following claims are helpful for extracting the discrete logarithms using the representations along with $\mathcal{V}_2$ queries/signatures.

Claim D.2. *Define $\mathcal{V}_1 := A + \psi \cdot B$ and $\mathcal{V}_2 := C + \psi \cdot D$, where $\mathcal{V}_1, \mathcal{V}_2$, and ψ are unknown while A, B, C, and D are known constants. If $\mathcal{V}_1 = \mathcal{V}_2$ and $B \neq D$, the value ψ can be recovered.*

Proof. As $\mathcal{V}_1 = \mathcal{V}_2$, we have that $\psi = \frac{A-C}{D-B}$. As $D \neq B$, the equation is solvable, and ψ can be recovered. □

Claim D.3. *Define $\mathcal{V}_1 := \psi \cdot A + B \cdot (C + \psi \cdot D)$ and $\mathcal{V}_2 := E + \psi \cdot F$, where $A, \ldots, F$ are known constants, while $\mathcal{V}_1, \mathcal{V}_2$, and ψ are unknown. If the condition $\mathcal{V}_1 = \mathcal{V}_2 \wedge C \neq 0 \wedge B \neq \frac{E}{C}$ holds, the value ψ can be recovered.*

Proof. As $\mathcal{V}_1$=$\mathcal{V}_2$, we have that

$$\psi \cdot (A + B \cdot D - F) = E - B \cdot C, \tag{8}$$

and thus

$$\psi = \frac{E - B \cdot C}{A + B \cdot D - F}. \tag{9}$$

This equation is defined iff $A + B \cdot D - F \neq 0$.

We rearrange equation (8) as

$$\psi \cdot (A + B \cdot D - F) = C \cdot (\frac{E}{C} - B).$$

Since $C \neq 0$ and $\frac{E}{C} \neq B$, it follows that $C \cdot (\frac{E}{C} - B) \neq 0$, and thus, $(A + B \cdot D - F) \neq 0$. We conclude that equation (9) is defined and the value ψ can be recovered. □

Claim D.4. *Define $\mathcal{V}_1 := A+B\cdot(C+\psi\cdot D)$ and $\mathcal{V}_2 := E+\psi\cdot F$, where $A, \ldots, F$ are known constants, while $\mathcal{V}_1, \mathcal{V}_2$, and ψ are unknown. If the condition $\mathcal{V}_1 = \mathcal{V}_2 \wedge D \neq 0 \wedge B \neq \frac{F}{D}$ holds, the value ψ can be recovered.*

Proof. As $\mathcal{V}_1$=$\mathcal{V}_2$, we have that

$$\psi \cdot (B \cdot D - F) = E - A - B \cdot C, \tag{10}$$

and thus

$$\psi = \frac{E - A - B \cdot C}{B \cdot D - F}. \tag{11}$$

It remains to show that $B \cdot D - F \neq 0$ to ensure that equation (11) is defined. We rewrite $B \cdot D - F$ as $D \cdot (B - \frac{F}{D})$. Since $D \neq 0$ and $B \neq \frac{F}{D}$, it follows that $D \cdot (B - \frac{F}{D}) \neq 0$, and thus, $B \cdot D - F \neq 0$ and equation (11) is defined and solvable for ψ. □

E ANONYMOUS CREDENTIALS LIGHT

E.1 Construction

For hash functions H_{pp}, H_1, and H_3 modeled as random oracles, and a public parameter $\mathsf{pp} := (\mathbb{G}, g, q)$, the scheme $\mathsf{ACL} = (\mathcal{G}, \mathcal{R}, \mathcal{S}, \mathcal{U}, \mathcal{V})$ is defined as follows:

- $\mathcal{G}(pp)$: Parse pp to extract $(\mathbb{G}, g, q)$, sample $x \leftarrow_\$ \mathbb{Z}_q$, $h \leftarrow_\$ \mathbb{G}$, compute $y := g^x$, $z := \mathsf{H}_1(g, h, y)$, set $\mathsf{sk} := x$ and $\mathsf{pk} := (g, h, y, z)$, and output a pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathcal{R}_{\mathcal{U}}(\mathsf{pk}, \mathsf{pp}, \overrightarrow{L}, \bar{R})$: Parse $\overrightarrow{L}$ as $L_1, \ldots, L_n$ and pp as $(\mathbb{G}, g, q)$ obtain $h_0, \ldots, h_n$ by querying $0, \ldots, n$ to H_{pp}, compute $C := h_0^{\bar{R}} \cdot h_1^{L_1} \cdot \ldots \cdot h_n^{L_n}$, send C. Compute proof of knowledge π of an opening of C. Output $C, \pi, st_r = R$.
- $\mathcal{R}_{\mathcal{S}}(\mathsf{sk}, C, \pi)$: Verify that π is a proof of knowledge of an opening of C. Output Accept if yes, Reject otherwise.
- $\mathcal{S}$: The signer algorithm. It consists of $\mathcal{S}_1$ and $\mathcal{S}_2$:
  - $\mathcal{S}_1(\mathsf{sk}, C)$: Sample $d, s_1, s_2, u, \mathrm{rnd} \leftarrow_\$ \mathbb{Z}_q$, generate $z_1 := g^{\mathrm{rnd}} C, z_2 := z/z_1$, compute $a := g^u, b_1 := g^{s_1} \cdot z_1^d$, $b_2 := h^{s_2} \cdot z_2^d$, set $st_s := (d, s_1, s_2, u), R := (\mathrm{rnd}, a, b_1, b_2)$, and output (R, st_s).
  - $\mathcal{S}_2(\mathsf{sk}, st_s, R, e)$: Set $d, s_1, s_2, u := st_s, x := sk$, compute $c := e - d \mod q$ and $r := u - c \cdot x \mod q$, and return $S := (c, d, r, s_1, s_2)$.
- $\mathcal{U}$: The user algorithm consists of two algorithms $\mathcal{U}_1$ and $\mathcal{U}_2$:
  - $\mathcal{U}_1(\mathsf{pk}, R, m, C)$: Parse $(\mathrm{rnd}, a, b_1, b_2) := R$, sample $\gamma \leftarrow_\$ \mathbb{Z}_q^*, \tau, t_1, t_2, t_3, t_4, t_5 \leftarrow_\$ \mathbb{Z}_q$, generate $z_1 := g^{\mathrm{rnd}} \cdot C$, $\zeta := z^\gamma, \zeta_1 := z_1^\gamma, \zeta_2 := \zeta/\zeta_1$, compute $\alpha := a \cdot g^{t_1} \cdot y^{t_2}, \beta_1 := b_1^\gamma \cdot g^{t_3} \cdot \zeta_1^{t_4}, \beta_2 := b_2^\gamma \cdot h^{t_5} \cdot \zeta_2^{t_4}, \eta := z^\tau$, generate $\varepsilon := \mathsf{H}_3(\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta, m)$, compute $e := \varepsilon - t_2 - t_4 \mod q$, set $st_u := (\gamma, \tau, t_1, t_2, t_3, t_4, t_5, \mathrm{rnd})$, and return (e, st_u).
  - $\mathcal{U}_2(\mathsf{pk}, st_u, R, e, S, m)$: Parse st_u, and S to extract $(\gamma, \tau, t_1, t_2, t_3, t_4, t_5)$, and $(c, d, r, s_1, s_2, \mathrm{rnd}_1, \mathrm{rnd}_2)$. Compute $\rho := r + t_1, \omega := c + t_2, \sigma_1 := \gamma \cdot s_1 + t_3, \sigma_2 := \gamma \cdot s_2 + t_5, \delta := d + t_4, \mu := \tau - \delta \cdot \gamma$, and return the signature $\hat{\sigma} := (\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu)$ if $\omega + \delta \equiv \mathsf{H}_3(\zeta, \zeta_1, g^\rho \cdot y^\omega, g^{\sigma_1} \cdot \zeta_1^\delta, h \cdot \zeta_2^\delta, z^\mu \cdot \zeta^\delta, m)$, $\perp$ otherwise. Output $r = (\gamma, \mathrm{rnd})$.
- $\mathcal{V}(\mathsf{pk}, m, \hat{\sigma})$: Parse $\hat{\sigma}$ as $(\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu)$, and output 1 if $\zeta \not\equiv 1$ and $\omega + \delta \equiv \mathsf{H}_3(\zeta, \zeta_1, g^\rho \cdot y^\omega, g^{\sigma_1} \cdot \zeta_1^\delta, h \cdot (\zeta/\zeta_1)^\delta, z^\mu \cdot \zeta^\delta)$, 0 otherwise.
- $\mathcal{SH}_U(\mathsf{pk}, \sigma, \vec{L}, r, st_r, \vec{L}')$: Parse $r = (\gamma, \mathrm{rnd})$ and $\sigma = (\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu)$. Compute $\Gamma = g^\gamma$, $h_i' = h_i^\gamma$ for $i \in [n]$, and set $\vec{h}' := (h_0', \ldots, h_n')$. Compute π_{sdl} as a proof of same discrete logarithm that shows that $\mathrm{dlog}_z \zeta = \mathrm{dlog}_g \Gamma = \mathrm{dlog}_{h_0} h_0' = \ldots = \mathrm{dlog}_{h_n} h_n'$. Compute the partial commitment $\zeta_1' = \zeta_1/\vec{h}'^{\vec{L}'}$. Using $st_r = R$, and γ, compute π_{op} as a proof of knowledge of an opening of ζ_1' w.r.t. rnd and the parameters in $\vec{h}'$ that correspond to attributes not in $\vec{L}'$. Output $\pi := (\Gamma, \vec{h}', \pi_{sdl}, \pi_{op})$
- $\mathcal{SH}_V(\mathsf{pk}, \sigma, \vec{L}', \pi)$: Parse $\sigma = (\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu)$, $\pi = (\Gamma, \vec{h}', \pi_{sdl}, \pi_{op})$, and $\vec{h}' = (h_0', \ldots, h_n')$. Check that π_{sdl} is a valid proof of the same discrete logarithm that proves that $\mathrm{dlog}_z \zeta = \mathrm{dlog}_{h_0} h_0' = \ldots = \mathrm{dlog}_{h_n} h_n'$. Compute the partial commitment $\zeta_1' = \zeta_1/\vec{h}'^{\vec{L}}$. Verify that π_{op} is a valid proof of knowledge of an opening of ζ_1' w.r.t. the parameters in $\vec{h}'$ that correspond to attributes, not in $\vec{L}'$.

E.2 Proof of Theorem 4.1

We recall the theorem:

Theorem 4.1. *Let $\mathbb{G}$ be a group of prime order q with generator g where* DLP *is (t, ε)-hard and such that* $\mathsf{ID}_{\mathsf{ACL}}$ *is (t, ε')-ℓ-OMMIM secure. Then* ACL *is (t, ε'')-one-more unforgeable in the AGM+ROM with*

$$\varepsilon'' = \varepsilon' + 13 \cdot \varepsilon + \frac{22 + 3 \cdot \xi + \ell + 1}{q},$$

where ξ is the number of registration queries the adversary makes.

Proof. Before we start with the proof, we give a high-level overview of the proof and introduce simplifying notation.

Proof Sketch. The ultimate goal of this proof is to provide a reduction that transforms an adversary $\mathcal{A}$ against ℓ-$\mathsf{OMUF}_{\mathsf{ACL}}$ into an adversary $\mathcal{B}_{10}$ against ℓ-$\mathsf{OMMIM}_{\mathsf{ID}_{\mathsf{ACL}}}$. However, the reduction we provide requires multiple restrictions on the representations of the group elements output by $\mathcal{A}$ during its interaction with the game ℓ-OMUF. Therefore, before introducing our reduction, we start by showing that $\mathcal{A}$ must adhere to these restrictions via game hop. First, in games $\mathsf{G}_{[1-7]}$ we abort if $\mathcal{A}$ uses group elements from the public key of the ℓ-OMUF challenger to compute its commitments C during the registration phase explicitly (e.g., by using y to compute C) or implicitly (e.g., by using a nonce $a = g^r \cdot y^c$ to

compute C). We proceed in games $\mathbf{G}_{[8-9]}$ to ensure that the group element ζ in the RO queries (that are used later in the signatures) is computed honestly, i.e., its representation only consists of a z component, and abort otherwise. Finally, in game $\mathbf{G_{10}}$ we abort if ζ_1 in the RO queries (that are used later in the signatures) was not computed honestly, i.e. as $g^{\mathrm{rnd}} \cdot C$, where rnd is a nonce received from $\mathcal{S}_1$ query.

Assumptions and Simplifying Notations. We denote the number of opened signing sessions by κ. For the games $\mathbf{G_1}$ to $\mathbf{G_{10}}$ we make the following simplifying assumption: Once the adversary has submitted its message-signature pairs, the game generates values e_i (by hashing the commitments without blinding along with randomly chosen messages) for all sessions that have not been closed and manually closes them using e_i (this also implicitly generates signatures for these sessions). Consequently, the values $e_i, c_i, d_i, r_i, s_{1,i}, s_{2,i}$ are well-defined for each signer session, even if the adversary never closed the session. We now turn to the proof.

Game $\mathbf{G_1}$. This is the OMUF game for ACL.

Game $\mathbf{G_2}$. This game aborts depending on the representations the adversary $\mathcal{A}$ submits for the commitments C. Let κ be the number of signing sessions the adversary opens. Let ξ be the number of commitments C the adversary registers.

Namely, we consider the occurrence of the group elements y, a_i for $i \in 1, \ldots, \kappa$ in the representation of C_j where C_j is the commitment C registered in the jth call to the registration oracle. For any session k that is never closed by the adversary, after the end of the game, sample a random value c_k. Define $r_k := u_k - c_k \cdot x$.

For any group element o_1 in the input of the adversary, and any group element o_2 in its output, we denote by $\overline{o_1(o_2)}$ the exponent of o_1 in the representation of o_2. Correspondingly, for a value C_j submitted by the adversary at registration, $\overline{y(C_j)}$ denotes the exponent of the public key element y in the representation of C_j and $\overline{a_i(C_j)}$ denotes the exponent of the session nonce a_i (for $i \in [\kappa]$) in the representation of C_j.

We are now ready to define the abort condition of $\mathbf{G_2}$.

$$\mathsf{E}_1 := \bigvee_{j=1}^{\xi} \left(\overline{y(C_j)} + \sum_{i=1}^{\kappa} c_i \cdot \overline{a_i(C_j)} \neq 0 \right).$$

We prove the following claim:

Claim E.1.

$$\Pr[\mathsf{E}_1] \leq \mathbf{Adv}^{\mathsf{DLP}}_{\mathcal{B}_1} + \frac{1}{q}.$$

Proof. We provide the reduction $\mathcal{B}_1$. The reduction simulates the OMUF game to the adversary using essentially the z-side witness as follows:

- **Setup.** Sample $w, w', w_0, \ldots, w_n \leftarrow_\$ \mathbb{Z}_q$. Set $h = g^w, z = g^{w'}$, and $h_i = g^{w_i}$ for $i = 0 \ldots n$. Program H_{pp} to output h_i on input i, for $i = 0, \ldots, n$. For a DLP challenge Y, set $y = Y$, and program H_1 to output z on input (g, h, y). Finally, output $\mathsf{pk} = (g, h, y, z)$ to the adversary.
- **Online Phase.** The reduction simulates the oracles as follows:
  - **Queries to** H_1,H_3, H_P,H_{pp}. Answer by lazy sampling for anything that is not yet programmed.
  - **Registration queries.** Upon a query of the form (C_j, π_j), abort if the proof π_j is invalid. Otherwise, E_1 occurs, extract $\mathrm{dlog}\, y$ as described below. Note that the reduction can compute $\mathrm{dlog}_g C_j$ using the representations of C_j the adversary submits alongside the registration query because the discrete logarithm of all the group elements in the adversary's input are known to the reduction.
  - $\mathcal{S}_1$ **Queries.** On input C_j (for some previously registered C_j), increase local session key counter i. Sample $c_i, r_i, u_{1,i}, u_{2,i}, \mathrm{rnd}_i \leftarrow_\$ \mathbb{Z}_q$. Set $z_{1,i} := C_j \cdot g^{\mathrm{rnd}_i}$. Set $a_i := y^{c_i} g^{r_i}$, $b_{1,i} := g^{u_{1,i}}$, $b_{2,i} := h^{u_{2,i}}$. Output $a_i, b_{1,i}, b_{2,i}, \mathrm{rnd}_i$.
  - $\mathcal{S}_2$ **Queries.** On input of a session id i, along with a challenge e_i, compute

$$d_i = e_i - c_i,$$
$$s_{1,i} = u_{1,i} - d_i \cdot (\mathrm{dlog}_g(C_j) + \mathrm{rnd}_i),$$
$$s_{2,i} = u_{2,i} - d_i \cdot \left(\frac{w' - (\mathrm{dlog}_g(C_j) + \mathrm{rnd}_i)}{w} \right).$$

We explain how the reduction can solve for $\mathrm{dlog}_g y$ if the event E_1 occurs.

In the following, we assume the event E_1 occurs for a fixed commitment C_j, call it C. As a proof of knowledge of an opening of the commitment C, the user sends the Fiat-Shamir transform of the protocol described in appendix A.1.2, i.e. it sends $c, (s_0, \ldots, s_n)$ such that $\mathsf{H}_P(C, C^c \cdot h_0^{s_0} \cdots h_n^{s_n}) = c$. If the proof is valid and the event E_1 occurred, the reduction obtains the following equation, where M is the group element submitted to the RO H_P to obtain the proof π:

$$\underbrace{C^c \cdot h_0^{s_0} \cdots h_n^{s_n}}_{:=V} = M. \tag{12}$$

The reduction can compute the representations of both sides of the equation to get

$$\mathrm{dlog}_g y \cdot \left(\overline{y(V)} + \sum_{i=1}^{\kappa} c_i \cdot \overline{a_i(V)} \right) + \not{y}_{[V]}$$
$$= \mathrm{dlog}_g y \cdot \left(\overline{y(M)} + \sum_{i=1}^{\kappa} c_i \cdot \overline{a_i(M)} \right) + \not{y}_{[M]}, \tag{13}$$

where for a group element o, $\not{y}_{[o]}$ is the exponent of g that can be used to compute to the non y part of the group element o. That is, $\not{y}_{[o]} := \overline{g(o)} + w \cdot \overline{h(o)} + w' \cdot \overline{z(o)} + \sum_{i=1}^{\kappa} (w' \cdot u_{2,i} \cdot \overline{b_{2,i}(o)} + u_{1,i} \cdot \overline{b_{1,i}(o)} + r_i \cdot \overline{a_i(o)}) + \sum_{i=1}^{n} w_i \cdot \overline{h_i(o)}$, hence $o = g^{\mathrm{dlog}_g y \cdot \left(\overline{y(o)} + \sum_{i=1}^{\kappa} c_i \cdot \overline{a_i(o)} \right) + \not{y}_{[o]}}$.

If the event E_1 occurs, this yields the following equation:

$$\mathrm{dlog}_g y = \frac{\not{y}_{[M]} - c \cdot \not{y}_{[C]}}{c \cdot \left(\overline{y(C)} + \sum_{i=1}^{\kappa} c_i \cdot \overline{a_i(C)} \right) - \left(\overline{y(M)} + \sum_{i=1}^{\kappa} c_i \cdot \overline{a_i(M)} \right)}$$

which can be solved for $\mathrm{dlog}_g y$ if $c \cdot \left(\overline{y(C)} + \sum_{i=1}^{\kappa} c_i \cdot \overline{a_i(C)} \right) \neq \left(\overline{y(M)} + \sum_{i=1}^{\kappa} c_i \cdot \overline{a_i(M)} \right)$. As the representation is submitted to the random oracle before the challenge c is received, this happens with

probability $1 - \frac{1}{q}$ and thus we can solve for $\text{dlog}_g y$ with probability $1 - \frac{1}{q}$.

Therefore, if the event E_1 occurs, the reduction can solve the DLP with probability $1 - \frac{\xi}{q}$.

It therefore holds that $\Pr[\mathbf{G_2} = 1] \geq \Pr[\mathbf{G_1} = 1] - \mathbf{Adv}^{\mathsf{DLP}}_{\mathcal{B}_1} + \frac{1}{q}$. □

Game $\mathbf{G_3}$. In this game, we want to abort if the adversary manages to use nonces a_i from not yet closed signing sessions in the representation of a commitment C without triggering the event E_1. We define this more formally. For a registration query C, π, let $I_c \subset [\kappa]$ be the set of session ids that have already been closed by the time H_P is queried for the proof π. Let $I_o \subset [\kappa]$ be the set of sessions that are open at this point in time.

We denote by E_2 the event that for some registration query with C, π it holds that:

$$0 \neq \overline{y(C)} + \sum_{i \in I_c} c_i \cdot \overline{a_i(C)} = - \sum_{i \in I_o} c_i \cdot \overline{a_i(C)}.$$

As for any $i \in I_o$, the value c_i is information-theoretically hidden from the adversary at the time the hash query is made, the probability that E_2 occurs for a single C is $\frac{1}{q}$. Overall we get that $\Pr[\mathsf{E}_2] \leq \frac{\xi}{q}$ and

$\Pr[\mathbf{G_3} = 1] \geq \Pr[\mathbf{G_2} = 1] - \frac{\xi}{q}$.

Game $\mathbf{G_4}$. In this game, we want to abort if the adversary manages to use h in the representation of a commitment C and still provide valid proof. We formally define the event E_3 as follows:

$$\mathsf{E}_3 \coloneqq \bigvee_{j=1}^{\xi} \left(\overline{h(C_j)} + \sum_{i=1}^{\kappa} s_{2,i} \cdot \overline{b_{2,i}(C_j)} \neq 0 \right)$$

The game $\mathbf{G_4}$ aborts if the event E_3 occurs. We sketch a reduction $\mathcal{B}_2$. The reduction embeds its discrete logarithm challenge in the h part of the public key. It responds to signing queries using the real secret key. When the adversary submits a commitment C and proof π during registration where the above expression is non-zero, the reduction uses a similar technique as in the hop to $\mathbf{G_2}$ to solve for $\text{dlog}\, h$.

It thus holds that $\Pr[\mathbf{G_4} = 1] \geq \Pr[\mathbf{G_3} = 1] - \mathbf{Adv}^{\mathsf{DLP}}_{\mathcal{B}_2} + \frac{1}{q}$.

Game $\mathbf{G_5}$. In this game, we want to abort if the adversary manages to use nonces $b_{2,i}$ from not yet closed signing sessions in the representation of a commitment C without triggering the event E_3 and still provide valid proof. This game hop works analogously to that to $\mathbf{G_3}$. We say that the event E_4 occurs if for some registration query with C and π it holds that

$$0 \neq \overline{h(C)} + \sum_{i \in I_c} s_{2,i} \cdot \overline{b_{2,i}(C)} = - \sum_{i \in I_o} s_{2,i} \cdot \overline{b_{2,i}(C)}$$

where I_o and I_c are defined as in the game hop to $\mathbf{G_3}$. As the values for $s_{2,i}$ for $i \in I_o$ are information-theoretically hidden from the adversary at time of querying H_P, the probability $\Pr[\mathsf{E}_4] \leq \frac{\xi}{q}$ and $\Pr[\mathbf{G_5} = 1] \geq \Pr[\mathbf{G_4} = 1] - \frac{\xi}{q}$.

Game $\mathbf{G_6}$. In this game, we want to abort if the adversary manages to use z in the representation of a commitment C and still provide valid proof. We define the event E_5 in which the game aborts as

$$\mathsf{E}_5 \coloneqq \bigvee_{j=1}^{\xi} \left(\overline{z(C_j)} + \sum_{i=1}^{\kappa} d_i \cdot \overline{b_{2,i}(C_j)} \neq 0 \right).$$

Using a similar strategy as for $\mathbf{G_2}$ and $\mathbf{G_4}$, we obtain that $\Pr[\mathbf{G_6} = 1] \geq \Pr[\mathbf{G_5} = 1] - \mathbf{Adv}^{\mathsf{DLP}}_{\mathcal{B}_3} + \frac{1}{q}$.

Game $\mathbf{G_7}$. In this game, we want to abort if the adversary manages to use nonces $b_{2,i}$ from not yet closed signing sessions in the representation of a commitment C without triggering the event E_5 and still provide valid proof. This game hop is analogous to $\mathbf{G_5}$. We get $\Pr[\mathbf{G_7} = 1] \geq \Pr[\mathbf{G_6} = 1] - \frac{\xi}{q}$.

Game $\mathbf{G_8}$. This game is the same as the game $\mathbf{G_7}$, except that it aborts if an event E_6 occurs.

E_6 is 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 without making a hash query with a response ε, such that, $\varepsilon = \omega + \delta$.

Claim E.2. $\Pr[\mathsf{E}_6] \leq \frac{\ell+1}{q}$.

Proof. Note that a signature $(\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \mu, \delta)$ on a message m is valid iff $\omega + \delta = \varepsilon = \mathsf{H}_3(\zeta, \zeta_1, g^\rho \cdot y^\omega, g^{\sigma_1} \cdot \zeta_1^\delta, h^{\sigma_2} \cdot \zeta_2^\delta, z^\mu \cdot \zeta^\delta, m)$. Thus, outputting a valid signature without making a hash query to H_3 is equivalent to guessing the output of $\mathsf{H}_3(\zeta, \zeta_1, g^\rho \cdot y^\omega, g^{\sigma_1} \cdot \zeta_1^\delta, h^{\sigma_2} \cdot \zeta_2^\delta, z^\mu \cdot \zeta^\delta, m)$. However, as the output of H_3 is uniformly random in q, $\mathcal{A}$ can output a valid signature (hence guessing ω and δ correctly) without making a hash query for this signature with a probability of at most $\frac{1}{q}$. Since $\mathcal{A}$ outputs $\ell + 1$ signatures, the probability that $\mathcal{A}$ did not make a hash query for one of them is $\frac{\ell+1}{q}$. □

Therefore, we have that $\Pr[\mathbf{G_8} = 1] \geq \Pr[\mathbf{G_7} = 1] - \frac{\ell+1}{q}$; therefore, if $\mathcal{A}$ wins the game $\mathbf{G_7}$, it wins $\mathbf{G_8}$.

For the next game, we introduce the following simplifying notation:

Simplifying Notation. While interacting with the reduction, the algebraic adversary $\mathcal{A}$ provides a representation for each group element it outputs or uses in its queries to the random oracle H_3. These representations must be expressible using a basis consisting of the group elements $\mathcal{A}$ acquires during the OMUF game. More specifically, this includes the group elements of the public key (g, y, h, z), and the group elements that it receives by interacting with $\mathcal{S}_1$. Assume $\mathcal{A}$ has invoked $\mathcal{S}_1$ t times for $1 \leq t$, the basis in the t_{th} $\mathcal{U}$ session is $Z_k \coloneqq (g, y, h, z, a_1, \ldots, a_t, b_{1,1}, \ldots, b_{1,t}, b_{2,1}, \ldots, b_{2,t}, z_{1,1}, \ldots, z_{1,t})$.

Recall that the group elements a, b_1 and b_2 can be expressed as $a = g^r \cdot y^c, b_1 = g^{s_1} \cdot z_1^d = g^{s_1 + v_1 \cdot d}$, and $b_2 = h^{s_2} \cdot z_2^d = h^{s_2} \cdot (z/z_1)^d = g^{w \cdot s_2 - v_1 \cdot d} \cdot z^d$, where $v_1 \coloneqq \text{dlog}_g z_1$ and $w \coloneqq \text{dlog}_g h$. Additionally, the discrete logarithms of h, and z_1 are known to the reduction, hence it can represent each element in Z_t using the reduced basis $J \coloneqq (g, y, z)$. While g, y, and z remain unchanged, we rewrite the other elements in Z_k as follows: $h = g^{w}, z_{1,i} = g^{v_{1,i}}, a_i = g^{r_i} \cdot y^{c_i}, b_{1,i} = g^{s_{1,i} + v_{1,i} \cdot d_i}$, and $b_{2,i} = h^{s_{2,i}} \cdot z_{2,i}^{d_i} = h^{s_{2,i}} \cdot (z/z_{1,i})^d = g^{w \cdot s_{2,i} - v_{1,i} \cdot d_i} \cdot z^{d_i}$, where $1 \leq i \leq t$. We can rewrite any representation of a group element o in its reduced form (w.r.t. basis

J) as $g_{[o]} + x \cdot y_{[o]} + v_0 \cdot z_{[o]}$, where $x \coloneqq \mathrm{dlog}_g\, y$ and $v_0 \coloneqq \mathrm{dlog}_g\, z$. The notation $g_{[o]}, y_{[o]},$ or $z_{[o]}$ refers to the respective component of the group element o.

In the following, we use the reduced representation to the basis $J = (g, y, z)$. Any representations submitted to the adversary can be reduced to this basis because the reduction knows all the discrete logarithms required for this transformation. (Note that all group elements are lumped in the g component, except y and z which are already reduced to the basis J).

As $\mathcal{A}$ has to make a hash query for each signature it outputs, each signature $(\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \mu, \delta)$ can be linked to its corresponding query since $\omega + \delta = \varepsilon$, where ε is the hash response of that query. Furthermore, the queries $\mathcal{A}$ sends to H_3 contain group elements $\zeta, \zeta_1, \alpha, \beta_1, \beta_2,$ and η. Each of those group elements can be generated by $\mathcal{A}$ arbitrarily, which includes using group elements from arbitrary signer sessions that it has received in $\mathcal{S}_1$ queries (i.e. a, b_1, b_2). We say that a signer session i is linked to a hash query $(\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta, m)$, iff α has a_i component, or one of ζ or η has $b_{2,i}$ component in their representations. We say that a signature σ is linked to signer session i, iff its corresponding hash query $(\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta, m)$ is linked to i.

Game $\mathbf{G_9}$. This game is the same as the game $\mathbf{G_8}$, except that it aborts if an event E_7 occurs.

We define E_7 as

$$\mathsf{E}_7 \coloneqq \bigvee_{i=1}^{\ell+1} (z^{z_{[\zeta_i]}} \neq \zeta_i)$$

Intuitively, the adversary $\mathcal{A}$ outputs a signature with a ζ component that was not computed honestly as $z^{z_{[\zeta]}}$, and equivalently, $g_{[\zeta]} + x \cdot y_{[\zeta]} \neq 0$.

We assume, for the sake of contradiction, that $\mathcal{A}$ provokes the event E_7 by outputting a signature with a ζ component, such that, $z^{z_{[\zeta]}} \neq \zeta$, and provide a reduction that solves the discrete logarithm problem DLP.

We call the first signature $\mathcal{A}$ outputs satisfying this property as *the special signature*. As $\mathcal{A}$ wins the game $\mathbf{G_8}$, it makes a hash query for each signature it outputs, and the reduction has access to these queries, including the query of the special signature and the representations of the group elements alongside this query.

Assume the query of the special signature is linked to k signer sessions. We mark the values $\mathcal{A}$ receives from those sessions with an asterisk, such as a^* and d^*.

Definition E.1. Akin to definition 3.4, we define the group elements $\alpha', \eta',$ and ζ' by eliminating all a_i^* components from the representation of α, and all $b_{2,i}^*$ components from the representations of ζ and η. We define the following preliminary values for the special signature[2]:

$$\omega'(\overrightarrow{c^*}) \coloneqq y_{[\alpha']} + \sum_{i=1}^{k} c_i^* \cdot \overline{a_i^*(\alpha)},$$

$$\delta'(\overrightarrow{d^*}) \coloneqq \frac{g_{[\eta']} + x \cdot y_{[\eta']} + \sum_{i=1}^{k} \overline{b_{2,i}^*(\eta)} \cdot (u_{3,i}^* - d_i^* \cdot v_0)}{g_{[\zeta']} + x \cdot y_{[\zeta']} + \sum_{i=1}^{k} \overline{b_{2,i}^*(\zeta)} \cdot (u_{3,i}^* - d_i^* \cdot v_0)},$$

[2] and its respective hash query.

where $u_{3,i}^* \coloneqq \mathrm{dlog}_g\, b_{2,i}^*$ and $v_0 \coloneqq \mathrm{dlog}_g\, z, \overrightarrow{c^*} = (c_1^*, \ldots, c_k^*), \overrightarrow{d^*} = (d_1^*, \ldots, d_k^*)$.

Next, we define an event E_8 as

$$\mathsf{E}_8 \coloneqq \omega \neq \omega'(\overrightarrow{c^*}) \wedge$$
$$g_{[\zeta']} + x \cdot y_{[\zeta']} + \overline{b_{2,i}^*(\zeta)} \cdot (u_{3,i}^* - d_i^* \cdot v_0) = 0 \vee \delta \neq \delta'(\overrightarrow{d^*}).$$

We show that $\mathcal{A}$ cannot provoke this event, except with a probability $\frac{2}{q}$.

Claim E.3. $\Pr[\mathsf{E}_8] \leq \frac{6}{q} + \mathbf{Adv}_{\mathcal{B}_5}^{\mathsf{DLP}}$.

Proof. We define an event $\mathsf{E}_8' \coloneqq \omega'(\overrightarrow{c^*}) + \delta'(\overrightarrow{d^*}) = \varepsilon = \omega + \delta$. Obviously, E_8 implies E_8', hence it suffices to show that E_8' occurs with a probability at most $\frac{6}{q} + \mathbf{Adv}_{\mathcal{B}_5}^{\mathsf{DLP}}$. We note that E_8' is equivalent to the event E_x (see definition D.1) by setting $C_0 = \sum_{i=1}^{k-1} c_i^* \cdot \overline{a_i^*(\alpha)}, C_3 = \sum_{i=1}^{k-1} \overline{b_{2,i}^*(\eta)} \cdot (u_{3,i}^* - d_i^* \cdot v_0)$, and $C_4 = \sum_{i=1}^{k-1} \overline{b_{2,i}^*(\zeta)} \cdot (u_{3,i}^* - d_i^* \cdot v_0), \mathcal{X} \coloneqq \omega', \mathcal{Y} \coloneqq \delta', \xi \coloneqq e_k, \xi' \coloneqq \varepsilon_k, \Delta \coloneqq d_k^*, \Omega \coloneqq c_k^*, C_0 \coloneqq y_{[\alpha']} + C_0, C_1 \coloneqq \overline{a_k^*(\alpha)}, C_2 \coloneqq g_{[\eta']} + x \cdot y_{[\eta']} + C_3, C_3 \coloneqq g_{[\zeta']} + x \cdot y_{[\zeta']} + C_4, C_4 \coloneqq v_0 \cdot \overline{b_{2,k}^*(\eta)},$ and $C_5 \coloneqq v_0 \cdot \overline{b_{2,k}^*(\zeta)}$. To apply claim D.1, we need that $C_4/C_3 \neq C_1$ if $C_5 = 0$. Assume the contrary. Then there exists a reduction $\mathcal{B}_5$ that solves for v_0 by embedding its discrete logarithm challenge in z and simulating using the y-side signer. If the adversray outputs representations with $C_4/C_3 = C_1$, the reduction solves for v_0 and breaks its discrete logarithm challenge. Otherwise, we apply claim D.1 to obtain a bound of $\frac{6}{q} + \mathbf{Adv}_{\mathcal{B}_5}^{\mathsf{DLP}}$. □

To simplify the remainder of the proof, we prove the following claim:

Claim E.4. *Let α, η, ζ be the group elements from the special sessions in game $\mathbf{G_9}$, and let α', η' and ζ' be the values described in definition E.1. It holds that*

(1) $y_{[\alpha]} = y_{[\alpha']} + \sum_{i=1}^{k} c_i^* \cdot \overline{a_i^*(\alpha)}$
(2) $g_{[\eta]} + x \cdot y_{[\eta]} = g_{[\eta']} + x \cdot y_{[\eta']} + \sum_{i=1}^{k} \overline{b_{2,i}^*(\eta)} \cdot (u_{3,i}^* - d_i^* \cdot v_0)$
(3) $g_{[\zeta]} + x \cdot y_{[\zeta]} = g_{[\zeta']} + x \cdot y_{[\zeta']} + \sum_{i=1}^{k} \overline{b_{2,i}^*(\zeta)} \cdot (u_{3,i}^* - d_i^* \cdot v_0)$

where k is the number of prover sessions linked to the special session.

Proof. Analogous to claim 3.1. □

Per claim E.4, we have that $g_{[\zeta]} + x \cdot y_{[\zeta]} = g_{[\zeta']} + x \cdot y_{[\zeta']} + \sum_{i=1}^{k} \overline{b_{2,i}^*(\zeta)} \cdot (u_{3,i}^* - d_i^* \cdot v_0)$. Since $\mathcal{A}$ provokes E_7, $g_{[\zeta]} + x \cdot y_{[\zeta]} \neq 0$ holds and we have that $g_{[\zeta']} + x \cdot y_{[\zeta']} + \overline{b_{2,i}^*(\zeta)} \cdot (u_{3,i}^* - d_i^* \cdot v_0) \neq 0$, hence $\mathsf{E}_8 = y_{[\alpha]} = \omega \wedge \frac{g_{[\eta]} + x \cdot y_{[\eta]}}{g_{[\zeta]} + x \cdot y_{[\zeta]}} = \delta$. Since $\mathcal{A}$ does not provoke the event E_8, it holds that $y_{[\alpha]} \neq \omega \vee \frac{g_{[\eta]} + x \cdot y_{[\eta]}}{g_{[\zeta]} + x \cdot y_{[\zeta]}} \neq \delta$. The reduction $\mathcal{B}_6$ exploits this fact to win the DLP game using the following strategy.

Given a discrete logarithm challenge U, $\mathcal{B}_6$ flips a coin and behaves as follows:

- On head, the reduction sets $y \coloneqq U$, and uses the z-side simulator from appendix C.1.2 to simulate the ℓ-OMUF-game for $\mathcal{A}$.

- On tail, the reduction sets $z := U$, and uses the y-side simulator from appendix C.1.1 to simulate the ℓ-OMUF-game for $\mathcal{A}$.

Using the special signature and the representations of its hash query, the reduction extracts $\mathrm{dlog}_g U$ following the strategy in the first matching case:

- $y_{[\alpha]} \neq \omega$: This case is analogous to case 1 in claim 3.4.
- $\frac{g_{[\eta]}+x\cdot y_{[\eta]}}{g_{[\zeta]}+x\cdot y_{[\zeta]}} \neq \delta$: By outputting the special signature, $\mathcal{A}$ submits two different representations of the group element η. The first representation is derived from the signature $(\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \mu, \delta)$. In particular, we have

$$\eta = z^{\mu} \cdot \zeta^{\delta}.$$

By reducing this representation to the basis J, we get

$$v_0 \cdot \mu + \delta \cdot (g_{[\zeta]} + x \cdot y_{[\zeta]} + v_0 \cdot z_{[\zeta]}).$$

The second representation $\mathcal{A}$ submits is the one it submits alongside the hash query of the special signature, which can be reduced to the basis J as

$$g_{[\eta]} + x \cdot y_{[\eta]} + v_0 \cdot z_{[\eta]}.$$

We note that we can use the technique from claim D.3 to extract $v_0 = \mathrm{dlog}_g z = dlog_g U$. By setting $\psi := v_0, A := \mu, B := \delta, C := g_{[\zeta]} + x \cdot y_{[\zeta]}, D := z_{[\zeta]}, E := g_{[\eta]} + x \cdot y_{[\eta]}, F := z_{[\eta]}, \mathcal{V}_1 := \psi \cdot A + B \cdot (C + \psi \cdot D)$, and $\mathcal{V}_2 := E + \psi \cdot F$, we can recover v_0 because the condition $\mathcal{V}_1 = \mathcal{V}_2 \wedge C \neq 0 \wedge B \neq \frac{E}{C}$ holds.

Consequently, $\mathcal{B}_6$ extracts the discrete logarithm successfully if the correct simulator is used, which happens with probability $\frac{1}{2}$; therefore, we have

$$\mathbf{Adv}^{\mathsf{DLP}}_{\mathcal{B}_6} \geq \frac{1}{2} \cdot \Pr[\neg \mathsf{E}_8].$$

Since

$$\Pr[\mathsf{E}_7] = \Pr[\mathsf{E}_8] + \Pr[\neg \mathsf{E}_8],$$

we have

$$\Pr[\mathsf{E}_7] = \frac{5}{q} + \mathbf{Adv}^{\mathsf{DLP}}_{\mathcal{B}_5} + 2 \cdot \mathbf{Adv}^{\mathsf{DLP}}_{\mathcal{B}_6},$$

hence

$$\Pr[\mathbf{G_9} = 1] \geq \Pr[\mathbf{G_8} = 1] - \frac{5}{q} + \mathbf{Adv}^{\mathsf{DLP}}_{\mathcal{B}_5} - 2 \cdot \mathbf{Adv}^{\mathsf{DLP}}_{\mathcal{B}_6}.$$

We thus assume that $\mathcal{A}$ cannot provoke the event E_7, hence it wins the game $\mathbf{G_9}$, and ζ is computed honestly in all forgeries output by $\mathcal{A}$ if it wins the game ℓ-OMUF.

Game $\mathbf{G_{10}}$. In this game, we want to abort if the adversary provides a representation of ζ_1 in a hash query to H_3 that is later used in a signature but does not yield $\mathrm{dlog}_g \zeta_1$ to the reduction.

We describe the event E_9 that causes $\mathbf{G_{10}}$ to abort as follows. For the ith signing session, denote by C_i the commitment used in that session. Let J_s be the set of hash queries that the adversary eventually uses in signatures. Denote by $\zeta_j, \zeta_{1,j}, \alpha_j, \beta_{1,j}, \beta_{2,j}, \eta_j$ the group elements submitted in the jth hash query.

$$\mathsf{E}_9 := \exists j^* \in J_s : \bigwedge_{i=1}^{\kappa} \left(\mathrm{dlog}_z C_i g^{\mathrm{rnd}_i} \neq \mathrm{dlog}_{\zeta_{j^*}} \zeta_{1,j^*} \right)$$

$$\wedge \left(\left(\overline{y(\zeta_{1,j^*})} + \sum_{i=1}^{\kappa} c_i \cdot \overline{a_i(\zeta_{1,j^*})} \neq 0 \right) \right.$$
$$\vee \left(\overline{h(\zeta_{1,j^*})} + \sum_{i=1}^{\kappa} s_{2,i} \cdot \overline{b_{2,i}(\zeta_{1,j^*})} \neq 0 \right)$$
$$\left. \vee \left(\overline{z(\zeta_{1,j^*})} + \sum_{i=1}^{\kappa} d_i \cdot \overline{b_{2,i}(\zeta_{1,j^*})} \neq 0 \right) \right)$$

Similar to the proof of theorem 3.1, we define so-called *preliminary values* for ω and δ. In particular, let j^* be the smallest hash index that corresponds to a signature that triggers the event E_9. We denote by $\alpha_{j^*}, \beta_{1,j^*}, \beta_{2,j^*}$ the group elements submitted in that hash query. We further denote by $\vec{c}$ and $\vec{d}$ the vectors of c_i and d_i for all opened sessions (recall that the reduction closes all unclosed sessions itself).

$$\omega'(\vec{c}) := \overline{y(\alpha_{j^*})} + \sum_{i=1}^{\kappa} c_i \cdot \overline{a_i(\alpha_{j^*})}$$

$$\delta'(\vec{c}) := \frac{\overline{y(\beta_{1,j^*})} + \sum_{i=1}^{\kappa} c_i \cdot \overline{a_i(\beta_{1,j^*})}}{\overline{y(\zeta_{1,j^*})} + \sum_{i=1}^{\kappa} c_i \cdot \overline{a_i(\zeta_{1,j^*})}}$$

$$\delta''(\vec{d}) := \frac{\overline{h(\beta_{1,j^*})} + \sum_{i=1}^{\kappa} s_{2,i} \cdot \overline{b_{2,i}(\beta_{1,j^*})}}{\overline{h(\zeta_{1,j^*})} + \sum_{i=1}^{\kappa} s_{2,i} \cdot \overline{b_{2,i}(\zeta_{1,j^*})}}$$

$$\delta'''(\vec{d}) := \frac{\overline{z(\beta_{1,j^*})} + \sum_{i=1}^{\kappa} d_i \cdot \overline{b_{2,i}(\beta_{1,j^*})}}{\overline{z(\zeta_{1,j^*})} + \sum_{i=1}^{\kappa} d_i \cdot \overline{b_{2,i}(\zeta_{1,j^*})}}$$

We note that by definition of E_9, at least one of the preliminary values δ', δ'', and δ''' is well-defined. Furthermore, the value ω' is always well-defined.

We define a new event E_{10} as

$$\mathsf{E}_{10} := \mathsf{E}_9 \wedge \left(\left(\delta'(\vec{c}) \neq \bot \wedge \delta'(\vec{c}) + \omega'(\vec{c}) = \varepsilon \right) \right.$$
$$\vee \left(\delta''(\vec{d}) \neq \bot \wedge \delta''(\vec{d}) + \omega'(\vec{c}) = \varepsilon \right)$$
$$\left. \vee \left(\delta'''(\vec{d}) \neq \bot \wedge \delta'''(\vec{d}) + \omega'(\vec{c}) = \varepsilon \right) \right)$$

We bound the event E_{10} as follows:

Claim E.5. $\Pr[\mathsf{E}_{10}] \leq \frac{12}{q} + \mathbf{Adv}^{\mathsf{DLP}}_{\mathbb{G}} \mathcal{B}_8 + \mathbf{Adv}^{\mathsf{DLP}}_{\mathbb{G}} \mathcal{B}_7$.

Proof. We will exclude the event E_{10} case by case. First, consider $\mathsf{E}_{10} \wedge (\delta'(\vec{c}) \neq \bot)$. We note that the game $\mathbf{G_{10}}$ can be simulated using either the z-side simulator or the y-side simulator from appendix C.1.2 and appendix C.1.1 which yield identical distributions. If the z-side simulator is used, the values for $\vec{c}$ are all already fixed by the time the adversary makes its hash query. Thus, the probability of $\mathsf{E}_{10} \wedge (\delta'(\vec{c}) \neq \bot)$ is at most $\frac{1}{q}$.

We now turn to the cases of $\mathsf{E}_{10} \wedge (\delta''(\vec{d}) \neq \bot)$ and $\mathsf{E}_{10} \wedge (\delta'''(\vec{d}) \neq \bot)$. We want to apply claim D.1. We consider the following cases:

(1) There are no sessions whose a_i and $b_{2,i}$ appear in the computation of the preliminary values that are open at the time of the hash query. Then the preliminary values are fixed and $\omega'(\vec{c}) + \delta''(\vec{d}) = \varepsilon$, resp. $\omega'(\vec{c}) + \delta'''(\vec{d}) = \varepsilon$ happens with probability $\frac{1}{q}$.

(2) There are sessions whose a_i and $b_{2,i}$ appear in the computation of the preliminary values that are never closed by the adversary. Then, e_i is sampled uniformly at random and the probability that $\omega'(\vec{c}) + \delta''(\vec{d}) = \varepsilon$ resp. $\omega'(\vec{c}) + \delta'''(\vec{d}) = \varepsilon$ is $\frac{1}{q}$.

(3) There are sessions whose nonces a_i and $b_{2,i}$ are used in the computation of the preliminary values and that are open at the time of the hash query, but all of them are closed by the adversary eventually. Let i^* be the session id of the last of those sessions to be closed. Let $\omega'(c_{i^*})$ and $\delta''(d_{i^*})$ be the values of ω' and δ'' when all sessions except for the i^*th session have been closed. We distinguish the following cases

(a) If the value δ''' is defined, we want to apply claim D.1 using $\Omega = c_{i^*}$, $\Delta = d_{i^*}$, $C_0 = \overline{y(\alpha_{j^*})} + \sum_{i\neq i^*} c_i \cdot \overline{a_i(\alpha_{j^*})}$, $C_1 = \overline{a_{i^*}(\alpha_{j^*})}$, $C_2 = \overline{z(\beta_{1,j^*})} + \sum_{i=1,i\neq i^*}^{\kappa} d_i \cdot \overline{b_{2,i}(\beta_{1,j^*})}$, $C_3 = \overline{z(\zeta_{1,j^*})} + \sum_{i=1,in\neq i^*}^{\kappa} d_i \cdot \overline{b_{2,i}(\zeta_{1,j^*})}$, $C_4 = \overline{b_{2,i}(\beta_{1,j^*})}$,$C_5 = \overline{b_{2,i}(\zeta_{1,j^*})}$ to upper bound the probability of the event $\mathsf{E}_{10} \wedge (\delta'''(\vec{d}) \neq \bot)$ with $\frac{6}{q}$. To this end, we need to show that if $C_5 = 0$, $C_4/C_3 \neq C_1$. Assume this case occurs, and $\delta'' = \bot$. In this case, it must hold that $\sum_{i=1}^{\kappa} s_{2,i} \cdot \overline{b_{2,i}(\beta_{1,j^*})} = 0$, as otherwise there exists a reduction $\mathcal{B}_7$ that can solve for the discrete logarithm of h by embedding its challenge in h and simulating using the y-side simulator. When the adversary submits its signatures, it learns the entire discrete logarithm of β_{1,j^*} to the base of g which it can use to solve for the discrete log of h. As d_{i^*} is revealed after C_4 has been fixed, it only happens with probability $\frac{1}{q}$ that for non-zero C_4 the h-component of β_{2,j^*} is zero. However, if $C_4/C_3 = C_1 = 0$, the value ω' is already independent of c_{i^*} and E_{10} occurs with probability $\frac{1}{q}$. Otherwise we can apply either claim D.1 or the next case.

(b) If the value δ'' is defined we again want to apply claim D.1. However, as δ'' depends on $s_{2,i}$, we first note that $s_{2,i} = \mathrm{dlog}_h\, b_{2,i} - d_i \cdot \mathrm{dlog}_h\, z_{2,i}$ where $z_{2,i} = z/(Cg^{\mathrm{rnd}_i})$. These discrete logarithms are well-defined and fixed before the adversary makes its hash query. We can therefore apply claim D.1 to exclude the event $\omega' + \delta'' = \varepsilon$ where $C_0 = \overline{y(\alpha_{j^*})} + \sum_{i\neq i^*} c_i \cdot \overline{a_i(\alpha_{j^*})}$, $C_1 = \overline{a_{i^*}(\alpha_{j^*})}$, $C_2 = \overline{h(\beta_{1,j^*})} + \sum_{i=1}^{\kappa} \overline{b_{2,i}(\beta_{1,j^*})}\,\mathrm{dlog}_h\, b_{2,i} - \sum_{i=1,i\neq i^*}^{\kappa} d_i \cdot \overline{b_{2,i}(\beta_{1,j^*})} \cdot \mathrm{dlog}_h(z/(Cg^{\mathrm{rnd}_i}))$, $C_3 = \overline{h(\zeta_{1,j^*})} + \sum_{i=1}^{\kappa} \overline{b_{2,i}(\zeta_{1,j^*})}\,\mathrm{dlog}_h\, b_{2,i} - \sum_{i=1,i\neq i^*}^{\kappa} d_i \cdot \overline{b_{2,i}(\zeta_{1,j^*})} \cdot \mathrm{dlog}_h(z/(Cg^{\mathrm{rnd}_i}))$, $C_4 = -\overline{b_{2,i}(\beta_{1,j^*})} \cdot \mathrm{dlog}_h(z/(Cg^{\mathrm{rnd}_{i^*}}))$, and $C_5 = -\overline{b_{2,i}(\zeta_{1,j^*})}\,\mathrm{dlog}_h(z/(Cg^{\mathrm{rnd}_{i^*}}))$. It must hold that $C_4/C_3 \neq C_1$ as otherwise a reduction $\mathcal{B}_8$ embedding in h could solve for $\mathrm{dlog}_h z$ and then for $\mathrm{dlog}_g h$. Consequently, the probability of $\mathsf{E}_{10} \wedge (\delta''(\vec{d}) \neq \bot)$ is at most $\frac{7}{q} + \mathbf{Adv}^{\mathsf{DLP}}_{\mathcal{B}_8}$.

In total, we obtain that $\Pr[\mathsf{E}_{10}] \leq \frac{13}{q} + \mathbf{Adv}^{\mathsf{DLP}}_{\mathcal{B}_8} + \mathbf{Adv}^{\mathsf{DLP}}_{\mathcal{B}_7}$. □

We now want to bound the probability of E_9 in $\mathbf{G}_9$.

Claim E.6. $\Pr[\mathsf{E}_9 \wedge \neg\mathsf{E}_{10}] \leq 4 \cdot \mathbf{Adv}^{\mathsf{DLP}}_{\mathcal{B}_9}$.

Proof. We provide a reduction $\mathcal{B}_9$ that will solve the discrete logarithm problem if $\mathsf{E}_9 \wedge \neg\mathsf{E}_{10}$.

- **Setup.** The reduction receives a discrete logarithm challenge Y. Its samples $w_0, \ldots, w_n \leftarrow_\$ \mathbb{Z}_q$. It sets $H_{\mathsf{pp}}(i) = g^{w_i} =: h_i$ for $i \in \{0, \ldots, n\}$ It flips a bit b. If $b = 0$ it sets $y = Y$, and samples $w, w' \leftarrow_\$ \mathbb{Z}_q$. It sets $h = g^w$ and $z = g^{w'}$. If $b = 1$, it samples another bit b'. It samples $x \leftarrow_\$ \mathbb{Z}_q$ and sets $y = g^x$. If $b = 0$, it sets $z = Y$ and samples $w \leftarrow_\$ \mathbb{Z}_q$, it then sets $h = g^w$. If $b = 1$, it sets $h = Y$ and samples w' and sets $z = g^{w'}$.
  It programs the random oracle $H_1(g, h, y) = z$. It outputs the public key (g, h, y, z) along with the parameters $h_0, \ldots h_n$.
- **Online Phase.** The reduction responds to oracle queries as follows:
  - **Hash queries to** $H_1, H_3, H_{\mathsf{pp}}, H_P$**:** Lazy sampling.
  - **Registration Queries.** On input of a commitment C and proof π, verify the proof. Abort if invalid or if one of the abort conditions of the previous games occurs. Otherwise, extract $\mathrm{dlog}\, C$ and save it in a list along with C.
  - **Queries to** $\mathcal{S}_1$**:** On input of commitment C_i, check if C_i is registered and abort if not. Increase session id counter i by 1. If $b = 0$, sample values $c_i, r_i, u_{1,i}, u_{2,i}$, $\mathrm{rnd}_i \leftarrow_\$ \mathbb{Z}_q$. Set $a_i = y^{c_i} g^{r_i}$, set $b_{1,i} = g^{u_{1,i}}$ and $b_{2,i} = h^{u_{2,i}}$. If $b = 1$, sample values $u_i, d_i, s_{1,i}, s_{2,i}, \mathrm{rnd}_i$. Set $a_i = g^{u_i}$. Set $b_{1,i} = (Cg^{\mathrm{rnd}_i})^{d_i} g^{s_{1,i}}$ and $b_{2,i} = (z/(Cg^{\mathrm{rnd}_i}))^{d_i} g^{s_{2,i}}$. In both cases return $(\mathrm{rnd}_i, a_i, b_{1,i}, b_{2,i})$.
  - **Queries to** $\mathcal{S}_2$**:** On input of a session id i and challenge e_i, if $b = 0$ compute $d_i = e_i - c_i$ and $s_{1,i} = u_{1,i} - d_i(\mathrm{rnd}_i + \mathrm{dlog}\, C_i)$, $s_{2,i} = u_{2,i} - d_i \cdot \left(\frac{w' - (\mathrm{dlog}_g(C_j) + \mathrm{rnd}_i)}{w}\right)$. If $b = 1$, compute $c_i = e_i - d_i$ and $r_i = u_i - c_i x$. In either case, return $(c_i, d_i, r_i, s_{1,i}, s_{2,i})$ to the adversary.
- **Output determination.** Close all signing sessions as described above. Identify the special session j^* as by the abort condition of the previous game, a γ_j can be computed for all ζ_j occurring in signatures. If the adversary wins and the event $\mathsf{E}_9 \wedge \neg\mathsf{E}_{10}$ occurs, compute the discrete logarithm of Y as described below.

Solving for the Discrete Logarithm. The reduction can solve for the discrete logarithm as follows:

- If $b = 0$ and $\omega'(\vec{c}) \neq \omega_{j^*}$: Compute $\rho' = \overline{g(\alpha_{j^*})} + w \cdot \overline{h(\alpha_{j^*})} + w' \cdot \overline{z(\alpha_{j^*})} + \sum_{i=1}^{n} w_i \cdot \overline{h_i(\alpha_{j^*})} + \sum_{i=1}^{\kappa} (r_i \cdot \overline{a_i(\alpha_{j^*})} + u_{1,i} \cdot \overline{b_{1,i}(\alpha_{j^*})} + u_{2,i} \cdot w \cdot \overline{b_{2,i}(\alpha_{j^*})})$. Then, compute $\mathrm{dlog}\, Y = \frac{\rho' - \rho_{j^*}}{\omega_{j^*} - \omega'}$.
- If $b = 0$ and $\delta'(\vec{c}) \neq \delta_{j^*}$. For a group element o, define $\not{y}_{[o]} = \overline{g(o)} + w \cdot \overline{h(o)} + w' \cdot \overline{z(o)} + \sum_{i=1}^{n} w_i \cdot \overline{h_i(o)} + \sum_{i=1}^{\kappa} (r_i \cdot \overline{a_i(o)} + u_{1,i} \cdot \overline{b_{1,i}(o)} + s + u_{2,i} \cdot w \cdot \overline{b_{2,i}(o)})$ First, compute $\sigma_1' = \not{y}_{[\beta_{1,j^*}]} - \delta'(\vec{c}) \cdot \not{y}_{[\zeta_{1,j^*}]}$. Finally, compute $\mathrm{dlog}_g\, \zeta_{1,j^*} = \frac{\sigma_1' - \sigma_{j^*}}{\delta_{j^*} - \delta'}$ and $\mathrm{dlog}\, Y = \frac{\mathrm{dlog}_g\, \zeta_{1,j^*} - \not{y}_{[\zeta_{1,j^*}]}}{\overline{y(\zeta_{1,j^*})} + \sum_{i=1}^{\kappa} c_i \cdot \overline{a_i(\zeta_{1,j^*})}}$.

- If $b = 1, b' = 1$, and $\delta''(\vec{d}) \neq \perp$. For a group element o, denote by $\not{h}_{[o]} = \overline{g(o)} + w' \cdot \overline{z(o)} + x \cdot \overline{y(o)} + \sum_{i=1}^{n} w_i \cdot \overline{h_i(o)} + \sum_{i=1}^{\kappa} (u_i \cdot \overline{a_i(o)} + (d_i \cdot (\mathrm{rnd}_i + \mathrm{dlog}_g C_i) + s_{1,i}) \cdot b_{1,i[o]} + (d_i \cdot (w' - (\mathrm{rnd}_i + \mathrm{dlog}_g C_i))) \cdot \overline{b_{2,i}(o)})$. Compute $\sigma_1'' = \not{h}_{[\beta_{1,j^*}]} - \delta''(\vec{d}) \cdot \not{h}_{[\zeta_{1,j^*}]}$. Then compute $\mathrm{dlog}_g \zeta_{1,j^*} = \frac{\sigma_1'' - \sigma_{j^*}}{\delta_{j^*} - \delta''}$ and finally $\mathrm{dlog}_g Y = \frac{\mathrm{dlog}\, \zeta_{1,j^*} - \not{h}_{[\zeta_{1,j^*}]}}{\overline{h(\zeta_{1,j^*})} + \sum_{i=1}^{\kappa} s_{2,i} \cdot \overline{b_{2,i}(\zeta_{1,j^*})}}$.
- If $b = 1, b' = 0$, and $\delta'''(\vec{d}) \neq \perp$. Similar to before, we will compute the discrete logarithm step by step. For a group element o, denote $\not{z}_{[o]} = \overline{g(o)} + w \cdot \overline{h(o)} + x \cdot \overline{y(o)} + \sum_{i=1}^{n} w_i \cdot \overline{h_i(o)} + \sum_{i=1}^{\kappa} (u_i \cdot \overline{a_i(o)} + (d_i \cdot (\mathrm{rnd}_i + \mathrm{dlog}\, C_i) + s_{1,i}) \cdot \overline{b_{1,i}(o)} + (w \cdot s_{2,i} - d_i \cdot (\mathrm{rnd}_i + \mathrm{dlog}\, C_i)) \cdot \overline{b_{2,i}(o)})$ First, we define $\sigma_1''' = \not{z}_{[\beta_{1,j^*}]} - \delta'''(\vec{d}) \cdot \not{z}_{[\zeta_{1,j^*}]}$. Finally, compute $\mathrm{dlog}_g Y = \frac{\mathrm{dlog}_g\, \zeta_{1,j^*} - \not{z}_{[\zeta_{1,j^*}]}}{\overline{z(\zeta_{1,j^*})} + \sum_{i=1}^{\kappa} d_i \cdot \overline{b_{2,i}(\zeta_{1,j^*})}}$

In total, this upper bounds the probability of $\mathsf{E}_9 \wedge \neg \mathsf{E}_{10}$ by $4 \cdot \mathbf{Adv}^{\mathsf{DLP}}_{\mathcal{B}_9}$. □

Putting this together with claim E.5 yields that.

$$\Pr[\mathbf{G_{10}} = 1] \geq \Pr[\mathbf{G_9} = 1] - 4 \cdot \mathbf{Adv}^{\mathsf{DLP}}_{\mathcal{B}_9} - \frac{7}{q}.$$

Simulating $\mathbf{G_{10}}$. We provide a reduction $\mathcal{B}_{10}$ that uses an adversary $\mathcal{A}$ that wins $\mathbf{G_{10}}$ to break the OMMIM-security of $\mathsf{ID}_{\mathsf{ACL}}$. The reduction proceeds as follows:

- **Setup.** On input of a public key $\mathsf{pk} = (g, h, y, z)$, the reduction samples $w_0, \ldots, w_n \leftarrow_\$ \mathbb{Z}_q$. It sets $H_{\mathsf{pp}}(i) = g^{w_i} =: h_i$ for $i \in \{0, \ldots, n\}$. It further sets $H_1(g, h, y) = z$. It provides pk along with $\mathsf{pp} = (h_0, \ldots, h_n)$ to the adversary.
- **Registration queries.** On input C, π, verify that π is a valid proof for C. Abort according to the conditions of $\mathbf{G_{10}}$. Compute $\mathrm{dlog}_g C$ from the representation (recall that by the above abort conditions, the representation of C consists only of $g, h_0, \ldots h_n$ for which the reduction knows the discrete logarithms.) Add $(C, \mathrm{dlog}\, C)$ to an internal list of registered commitments.
- **Signing Queries.**
  - **Queries to $\mathcal{S}_1$.** On input C of a previously registered C, open a prover session using a call to $\mathcal{P}_1$. Receive commitments $a_i, b_{1,i}, b_{2,i}$ as well as rnd_i. Compute $\mathrm{rnd}_i' = \mathrm{rnd}_i - \mathrm{dlog}_g C$ using $\mathrm{dlog}_g C$ from the list of registered commitments. Output $a_i, b_{1,i}, b_{2,i}, \mathrm{rnd}_i'$ to the adversary.
  - **Queries to $\mathcal{S}_2$.** On input e_i for a not yet closed session i, send e_i to the $\mathcal{P}_2$ oracle to close the corresponding prover session. Receive $c_i, d_i, r_i, s_{1,i}, s_{2,i}$ from the prover and forward this response to the adversary.
- **Hash Queries.** We explain how the reduction handles hash queries.
  - **To $H_P, H_{\mathsf{pp}}, H_1$.** Answered by lazy sampling.
  - **To H_3.** On input of $(\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta, m)$, it computes $\gamma = \mathrm{dlog}_z \zeta$ from the representation of ζ provided by the adversary. Furthermore, compute $z_1' = \zeta_1^{1/\gamma}$. If $z_1' = z_{1,i}$ for some $i \in [\kappa]$, it sets $\mathrm{rnd}'' = \mathrm{rnd}_i$. Otherwise, it computes $\mathrm{rnd}'' = \mathrm{dlog}_g \zeta_1 / \gamma$ if possible, otherwise abort (note that this reflects the abort condition introduced in $\mathbf{G_{10}}$). It then opens a verifier session by sending $\mathrm{rnd}'', \alpha, \beta_1^{\frac{1}{\gamma}}, \beta_2^{\frac{1}{\gamma}}$. It forwards the verifier response ε to the adversary.
- **Output determination.** Once the adversary outputs its signatures $(m_1, \sigma_1), \ldots, (m_{\ell+1}, \sigma_{\ell+1})$, parse each signature as $(\zeta_i, \zeta_{1,i}, \omega_i, \delta_i, \rho_i, \sigma_{1,i}, \sigma_{2,i}\mu_i)$. Identify the verifier session and the previously computed γ_i from the hash query $H_3(\zeta_i, \zeta_{1,i}, g^{\rho_i} y^{\omega_i}, \zeta_{1,i}^{\delta_i} g^{\sigma_{1,i}}, \zeta_{2,i}^{\delta_i} h^{\sigma_{2,i}}, \zeta_i^{\delta_i} z^{\mu_i}, m_i)$. Close the corresponding verifier session by sending over $(\omega_i, \delta_i, \rho_i/\gamma_i, \sigma_{1,i}/\gamma_i, \sigma_{2,i}/\gamma_i)$.

It thus holds that $\Pr[\mathbf{G_{10}} = 1] \leq \mathbf{Adv}^{\ell\text{-OMMIM}}_{\mathsf{ID}_{\mathsf{ACL}}}(\mathcal{B}_{10})$.

And we obtain

$$\begin{aligned}
\mathbf{Adv}^{\ell\text{-OMUF}}_{\mathsf{BS}_{\mathsf{ACL}}}(\mathcal{A}) \leq & \mathbf{Adv}^{\ell\text{-OMMIM}}_{\mathsf{ID}_{\mathsf{ACL}}}(\mathcal{B}_{10}) + 4 \cdot \mathbf{Adv}^{\mathsf{DLP}}_{\mathbb{G}}(\mathcal{B}_9) \\
& + \frac{12}{q} + \mathbf{Adv}^{\mathsf{DLP}}_{\mathbb{G}}\mathcal{B}_8 + \mathbf{Adv}^{\mathsf{DLP}}_{\mathbb{G}}\mathcal{B}_7 \\
& + \frac{5}{q} + \mathbf{Adv}^{\mathsf{DLP}}_{\mathbb{G}}\mathcal{B}_6 + \mathbf{Adv}^{\mathsf{DLP}}_{\mathbb{G}}\mathcal{B}_5 \\
& + 2 \cdot \mathbf{Adv}^{\mathsf{DLP}}_{\mathbb{G}}(\mathcal{B}_4) + \frac{2}{q} \\
& + \frac{\ell + 1}{q} + \frac{3 \cdot \xi}{q} + \frac{3}{q} \\
& + \mathbf{Adv}^{\mathsf{DLP}}_{\mathbb{G}}(\mathcal{B}_3) + \mathbf{Adv}^{\mathsf{DLP}}_{\mathbb{G}}(\mathcal{B}_2) + \mathbf{Adv}^{\mathsf{DLP}}_{\mathbb{G}}(\mathcal{B}_1) \\
\leq & \varepsilon' + 13 \cdot \varepsilon + \frac{26 + \ell + 3 \cdot \xi}{q}.
\end{aligned}$$

□

F PROOF OF THEOREM 4.3

Theorem 4.3. *If* ACL *is* (t, ε)-OMUF-*secure in the AGM + ROM, then* $\mathsf{BS}_{\mathsf{Abe}}$ *is* $(t, \varepsilon + \frac{q_S(q_S + q_{\mathsf{H}_2})}{q})$-OMUF-*secure in the AGM+ROM where* q_S *is the number of opened signing sessions and* q_{H_2} *is the number of queries made to hash oracle* H_2 *by an adversary against* OMUF-*security of* $\mathsf{BS}_{\mathsf{Abe}}$.

Proof. Assume, for the sake of contradiction, that $\mathsf{BS}_{\mathsf{Abe}}$ is not OMUF-secure, and thus, there is an adversary $\mathcal{A}$ that wins the game ℓ-OMUF against $\mathsf{BS}_{\mathsf{Abe}}$. We construct a reduction $\mathcal{B}$' that wins the game ℓ-OMUF against ACL.

The reduction behaves as follows:

- **Setup.** On input of a public key $\mathsf{pk} = (g, h, y, z)$, it registers a commitment C to the all 0 vector with the challenger. It forwards the public key to the adversary.
- **Online Phase.** The reduction handles oracle queries by the adversary as follows:
  - **Hash queries to H_1, H_3:** Forward to the corresponding oracles provided by the challenger.
  - **Hash queries to H_2:** Lazy sampling for values not programmed otherwise.
  - **Queries to $\mathcal{S}_1$:** Sample a value $\mathrm{rnd}_i' \leftarrow_\$ \mathbb{Z}_q$. Open $\mathcal{S}_1$ session using C with the challenger. Receive $\mathrm{rnd}_i, a_i, b_{1,i}, b_{2,i}$. Compute $z_{1,i} := C g^{\mathrm{rnd}_i}$ and set $H_2(\mathrm{rnd}_i') = z_{1,i}$. Abort if H_2 is already programmed at this point. Output $\mathrm{rnd}_i', a_i, b_{1,i}, b_{2,i}$ to the adversary.

  - **Queries to $\mathcal{S}_2$:** Query the corresponding oracle provided by the challenger for the corresponding session id i.
- **Output Determination.** When the adversary outputs message signature pairs $(m_1, \sigma_1), \ldots, (m_{\ell+1}, \sigma_{\ell+1})$, the reduction forwards those pairs to its own challenger.

We first compute the probability that the reduction aborts due to a pre-programmed hash value of H_2. Let q_{H_2} be the number of hash queries made to H_2 by the adversary, and let q_S be the number of $\mathcal{S}_1$ queries made by the adversary. Then the probability that the reduction aborts is at most $\frac{q_S \cdot (q_S + q_{H_2})}{q}$.

We obtain that $\mathbf{Adv}^{\mathsf{OMUF}}_{\mathcal{B}',\mathsf{ACL}} \geq \mathbf{Adv}^{\mathsf{OMUF}}_{\mathcal{A},\mathsf{BS}_{\mathsf{Abe}}} - \frac{q_S \cdot (q_S + q_{H_2})}{q}$. □

G SECURE SHOWING OF ACL

Theorem G.1. *The* ACL *scheme over a group* $\mathbb{G}$ *where the* DLP *is* (t, ε)*-hard is* ℓ-SH *secure with*

$$\mathbf{Adv}^{\ell\text{-SH}}_{\mathsf{BS}_{\mathsf{ACL}}}(\mathcal{A}) = 9\varepsilon + \frac{12 + 3\xi + \ell + 1}{q} + \mathbf{Adv}^{\ell\text{-RT}}_{\mathsf{ID}_{\mathsf{ACL}}}(\mathcal{B}_{\ell\text{-RT}}) + \mathbf{Adv}^{\mathsf{Bind}}_{\mathsf{PedCom}}(\mathcal{B}_{\mathsf{Bind}})$$

Proof Overview. Like in the proof of theorem 4.1, we first rule out some dishonest behaviour of the adversary regarding the algebraic representations it submits throughout its run.

We then consider two ways in which the adversary could generate signatures that do not match the commitments it used during the signing.

The first way is to generate a signature with values ζ, ζ_1 such that $\mathsf{dlog}_\zeta \zeta_1 \neq \mathsf{dlog}_z g^{\mathrm{rnd}_i} C_i$ for all i. We will show that we can reduce this case to contradict the Restrictive Tagging Lemma.

The other case is that all signatures can be mapped to signing sessions, but the adversary provides a vector $\vec{L}'$ tied to a signature that does not match the vector $\vec{L}$ that is used when opening the commitment C used during the signing. In this case, we will contradict the binding property of the Pedersen commitment.

Proof Sketch. The notation and simplifying assumptions are as in the proof of theorem 4.1.

We start out from the secure show game and then modify the game using the same hops ($\mathbf{G_1}$ to $\mathbf{G_{10}}$) as in the proof of theorem 4.1.

This yields a modified variant of the game as follows:

- The representations of the commitments C registered by the adversary only contain non-zero exponents for $g, h_0, \ldots, h_n$.
- The representations of all ζ_j (submitted during hash queries) that occur in signatures are such that $z^{z_{[\zeta_j]}} = \zeta_j$.
- For signatures with $\mathsf{dlog}_\zeta \zeta_1 \neq \mathsf{dlog}_z g^{\mathrm{rnd}_i} C_i$ for all i, the representation submitted for ζ_1 during the hash query $g, h_0, \ldots, h_n$.

We distinguish two cases.

The first case is that there exists a signature $\sigma = (\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu)$ with $\mathsf{dlog}_\zeta \zeta_1 \neq \mathsf{dlog}_z g^{\mathrm{rnd}_i} C_i$ for all i. We denote this by E_1. In this case, we provide a reduction $\mathcal{B}_{\ell\text{-RT}}$ that breaks the game ℓ-**RT**. $\mathcal{B}_{\ell\text{-RT}}$ is identical to the reduction $\mathcal{B}_6$ from the proof of theorem 4.1.

We have

$$\Pr[\mathbf{G_{10}} = 1 \wedge \mathsf{E}_1] \leq \mathbf{Adv}^{\ell\text{-RT}}_{\mathsf{ID}_{\mathsf{ACL}}}(\mathcal{B}_{\ell\text{-RT}})$$

In case all signatures "match", we consider the outputs of the adversary. The adversary needs to provide commitments $C = (\zeta, \zeta_1)$ and an opening $r = (\gamma, \mathrm{rnd}', R', \vec{L}')$ for each signature.

We note that the value γ is uniquely defined by ζ. The game verifies that $z^\gamma = \zeta$.

By the previous case, we know that $g^{\mathrm{rnd}'} h_0^R \prod_{i=1}^n h_i^{L'_i} = g^{\mathrm{rnd}_i} C_i$ for some i. Let $R, \vec{L}_i$ be the opening of C_i. I.e. it holds that $g^{\mathrm{rnd}'} h_0^{R'} \prod_{i=1}^n h_i^{L'_i} = g^{\mathrm{rnd}_i} h_0^R \prod_{i=1}^n h_i^{L_i}$

We note that these are essentially two openings of Generalized Pedersen commitments with public parameters $g, h_0, \ldots, h_n$.

Thus, we use this adversary to break the binding property of the generalized Pedersen Commitment by a straightforward reduction $\mathcal{B}_{\mathsf{Bind}}$.

We obtain

$$\Pr[\mathbf{G_{10}} = 1 \wedge \neg \mathsf{E}_1] \leq \mathbf{Adv}^{\mathsf{Bind}}_{\mathsf{PedCom}}(\mathcal{B}_{\mathsf{Bind}})$$

Plugging in the game hops from theorem 4.1 yields the theorem statement. □

H DEFERRED FIGURES FROM THE MAIN BODY

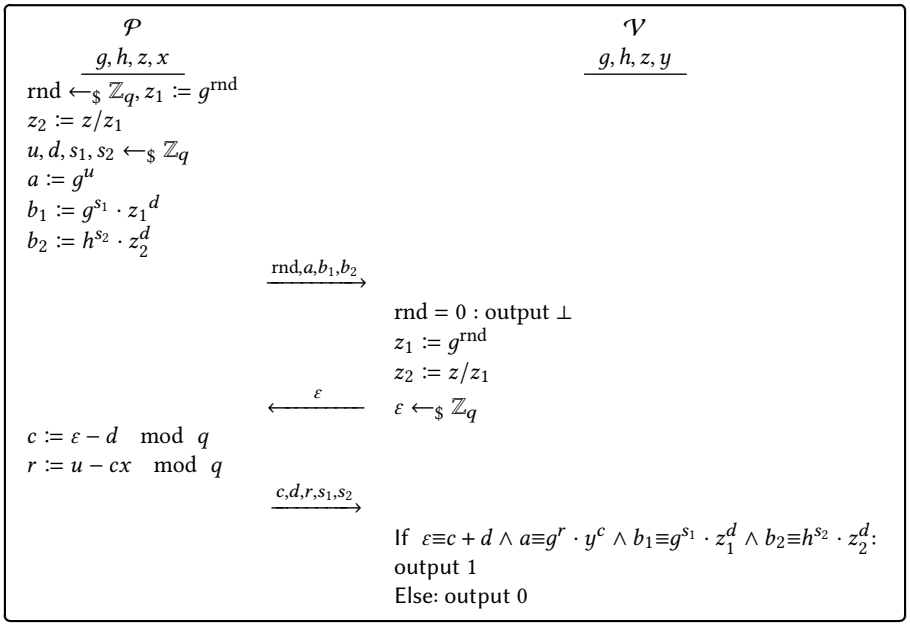

Figure 4: This figure depicts the interaction between the prover and the verifier in the identification scheme $\mathsf{ID}_{\mathsf{ACL}}$.

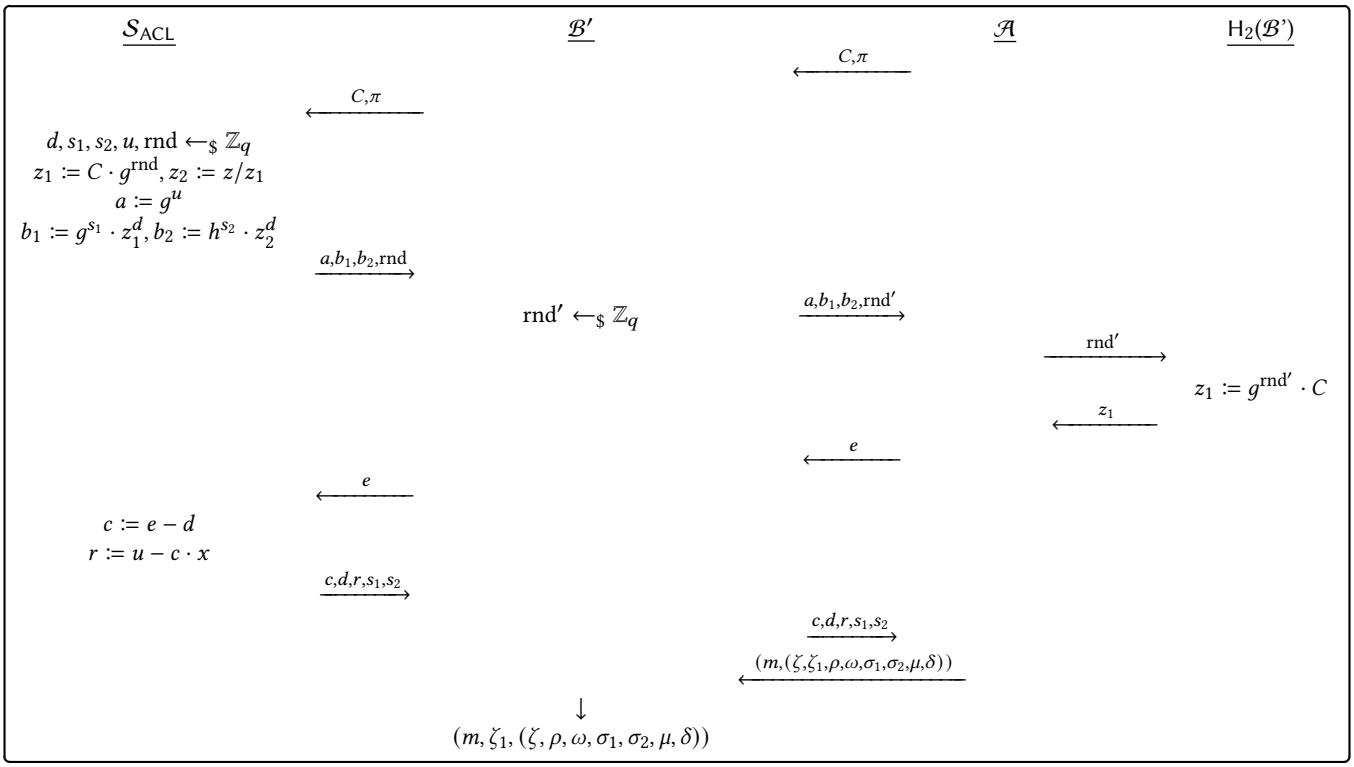

Figure 5: A high-level overview of the reduction $\mathcal{B}'$. It reduces the OMUF **security of** ACL **to the** OMUF **security of** $\mathsf{BS}_{\mathsf{Abe}}$**. In this figure,** $\mathcal{S}_{\mathsf{ACL}}$ **is the challenger's signer oracle,** $\mathcal{B}$**' is the reduction,** $\mathcal{A}$ **is a forger that wins** ℓ-OMUF **against** $\mathsf{BS}_{\mathsf{ACL}}$**, and** H_2 **is the** RO **of** $\mathsf{BS}_{\mathsf{ACL}}$**, which is simulated by the reduction** $\mathcal{B}$**'.**