Post-Quantum Anonymous Signatures from the Lattice Isomorphism Group Action

Chris van Noorden[1,2] and Paola de Perthuis[1]

[1] Centrum Wiskunde & Informatica (CWI), Nederland
[2] Universiteit Utrecht, Nederland

Abstract Post-quantum assumptions may not rely on the difficulty of finding secret subgroups as many classical schemes did. Instead, several assumptions make use of more general group actions, with the hope that quantum algorithms are not helpful in this less structured setting for an appropriate group choice. Group action-based constructions were first presented in the context of isogenies in which an ideal class group acts on elliptic curves, but equivalence problems in error-correcting codes and lattices also exhibit such structures, this time with non-commutative groups.
Previous works presented anonymity-preserving constructions in a generic group action framework; however, they were not general enough to encompass the group action underlying the *Lattice Isomorphism Problem* (LIP), for which the acting group is countably infinite and non-commutative.
We bridge this gap by, from zero-knowledge proofs of *OR* statements, building generic *blind signature* and *strong designated-verifier signature with non-delegatability* constructions from standard assumptions corresponding to a generalised group action inverse problem.

1 Introduction

Generalised cryptographic schemes built from group actions were first presented by Brassard and Yung in 1991 [BY91], and then by Couveignes in 1997 (published in the note [Cou06]) who formalised the notion of *Hard Homogeneous Spaces* (HSS), which generalised the discrete logarithm problem, corresponding to the action of a $\mathbb{Z}/n\mathbb{Z}$ group, for n a natural integer, in a unified framework with graph isomorphisms, or the action of ideal class groups over elliptic curves used in isogeny-based constructions.

Since then, group actions have been used in isogeny-based cryptography, [KTAT20] until group actions started appearing in code-based cryptology through new equivalence problems first presented in [SS13], which in comparison to isogeny-based ones lead to more efficient and easier to implement schemes, though with the drawback of using non-commutative groups (though the non-commutativity offers more resistance to the Kuperberg quantum algorithm).

This lead to generic constructions for blind signature [HLM+25, DKQ+25] and strong designated-verifier signature [MTT25] primitives from generic group

actions. However, these were not applicable to the one exhibited by the *Lattice Isomorphism Problem* (LIP) [DvW22], in which the infinite and non-commutative group of invertible integer matrices acts on quadratic forms.

Contributions. This work overcomes the aforementioned difficulty by using a more general group action fit to encompass the LIP as well as code or isogeny-based ones, in order to first build a zero-knowledge OR proof system, and then use it to build blind signatures and strong designated-verifier signatures with non-delegatability. The resulting group-action-based blind signatures not only allow lattice-based instantiations for the first time, but also improve on limitations of previous schemes in which the group required efficient square-root computations [DKQ+25] and new secret-key distributions on codes [KLP25, DKQ+25], or the use of strong assumptions such as an algebraic model and a non-standard one-more assumption [HLM+25] (though this last article proves unforgeability for more sessions than we do, which motivates the use of these assumptions). Moreover, our strong designated-verifier signatures with non-delegatability improve on [MTT25] by granting code-equivalence and lattice-isomorphism-based instantiations for the first time, by allowing the group action to be non-abelian and removing the need for a quadratic twist.

Related Work. The current article extends upon previous research rooted in isogeny-based group actions, or techniques that used zero-knowledge proofs of disjunctions, called *OR proofs* to obtain anonymity properties.

OR Proofs. Proofs of knowledge for disjunctive statements have since [CDS94] found numerous applications beyond privacy-preserving multi-party computations: interesting examples include simulation-sound zero-knowledge proofs, first introduced by Sahai [Sah99] in relation to the concept of non-malleable proofs. Namely, simulation-sound zero-knowledge proofs are useful in contexts in which attackers may adaptively try to generate new proofs from old ones, as these constructions allow showing them simulated proofs generated without using the witness to be hidden, while preventing them from reusing them to generate new ones. Witness-indistinguishable OR proofs yield such constructions by proving that either one knows a witness corresponding to the original statement, or another witness for a hard problem, where this second branch will be used to generate simulated proofs.

This work presents two complementary usages of OR proofs in the context of post-quantum anonymity: in our blind signatures, the signer proves the knowledge of one out of two secret keys, which will grant some form of simulation soundness used to obtain one-more unforgeability; and in an opposite perspective in our strong designated-verifier signatures, OR proofs will allow the intended verifier to simulate signatures, thus ensuring that they can convince no one else that they were originally provided by the signer. These strategies are not new, but were previously not usable for general group actions including the lattice isomorphism one.

In 1994, [CDS94] showed a generic way to obtain witness-indistinguishable OR proofs of knowledge of one among two secret keys, given a public-coin honest-verifier zero-knowledge proof system and a secret sharing scheme adapted to its structure. This strategy was general enough to be readily applicable to post-quantum signature schemes, as for instance demonstrated by [AHJ21] in order to obtain blind signatures from lattice-based MLWE and MSIS assumptions.

In the context of group actions, this technique may prove less straightforward– or at least, less efficient–, namely because secret sharing schemes have to be adapted to a weaker structure which cannot benefit from the expressivity of $(\mathbb{Z}/n\mathbb{Z})$-module operations. For instance, [KLLQ23] had to use multiplicative shares in $\{-1, 1\}$ instead of previous instantiations with Shamir secret sharing [Sha79] making use of finite fields.

Blind Signatures from Group Actions. In fact, CSI-Otter [KLLQ23] proposed a first construction of blind signatures (*i.e.* obtained by interacting with a signer who should not be able to recognise the origin session of the output signature) based on a non-discrete-logarithm group action, but it was tailored for isogenies and relied on the quadratic twists of elliptic curves, which was not transposable to other post-quantum families.

Other works [KLP25, DKQ+25, HLM+25] then recently proposed blind signatures from more generic group actions, removing the need for a quadratic twist and allowing the group to be non-commutative, hence also error-correcting-code instantiations; however, these were not yet applicable to the lattice isomorphism group action.

In more detail, [KLP25] overcame this difficulty by introducing a new code assumption, the MIMCE (in Definition 2.20), in which secret keys must be symmetric or antisymmetric. Then, [DKQ+25], also inspired by the construction of [KLLQ23], proposed another adaptation doubling its public key size and requiring the group to allow efficient square root computations, under the condition that it be safe to reuse group elements, also resulting in new error-correcting-code assumptions, the DmILCE and ILCE (Problems 2 and 3).

Tanuki [HLM+25] then proposed constructions with sequential and concurrent securities for polynomially many sessions, and as for the two previous schemes, a group action general enough to encompass both isogeny and code-based instantiations. However, they relied on an algebraic model and a non-standard one-more assumption. In fact the authors state in [HLM+25, Remark 3] that the open problem of a LIP-based instantiation of blind signatures would provide an interesting contribution to the field–which we now provide in this work. On the other hand, unlike [HLM+25], we prove unforgeability of our blind signatures for only logarithmically many sessions.

The table in Figure 1 compares the assumptions and concurrent one-more unforgeability for different group-action-based blind signature schemes.

Strong Designated-Verifier Signatures with Non-Delegatability from Group Actions. While *designated-verifier signatures* ensure that only the intended verifier may be convinced of the signer's voluntary issuance, *strong* designated-verifier

Construction	Group Action Requirements	Base Assumptions	Model	Concurrent OMUF
[KLLQ23]	Co, Fr, Fi, QT	GAIP	ROM	$\Theta(\log(\lambda))$
[DKQ$^+$25]	Fr, Fi, Sqr	IGAP, GASDDH	ROM	$\Theta(\log(\lambda))$
[KLP25]	Fr, Fi	IGAP	ROM	$\Theta(\log(\lambda))$
[HLM$^+$25, Sec. 5]	Fi	OMGAIP	ROM, AGAM	$\mathsf{pol}(\lambda)$
Ours (GABS)	IG	GAIP	ROM	$\Theta(\log(\lambda))$

Figure 1. A comparison between blind signature schemes based on group actions. For a group action $* : \mathbb{G} \times \mathcal{Q} \to \mathcal{Q}$, it uses the following abbreviations: Co : $\mathbb{G}$ is commutative, Fr : $*$ is free, Fi : $\mathbb{G}$ is finite, IG : existence of an instance generator (see Definition 6), Sqr : efficient squareroot computations in $\mathbb{G}$, QT: a quadratic twist operation, available in isogeny-based group actions.

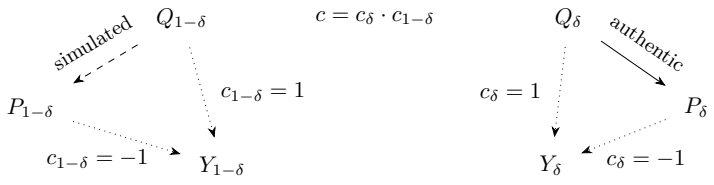
signatures (SDVS) additionally require that no external observer may learn with respect to which signer identities signatures are stated, in order to enhance the scheme's anonymity properties. Further, as SDVS were conceived for voting applications, *non-delegatability* ensures that, outside of the verifier, no one may construct a valid signature without the *full* signer secret key.

Strong designated-verifier signatures with non-delegatability (SDVSND) from group actions were recently proposed by [MTT25]. However, their construction only applies to the abelian case and uses quadratic twist of elliptic curves, specific to isogeny-based instantiations.

Free Lattice-Isomorphism Group Actions. Our blind-signature construction originally required free group actions, that we instantiated with the LIP using automorphism-free lattices. We since removed this constraint, but still provide this family of lattices as it could be of independent interest for other constructions.

Technical Overview.

OR Proofs. Our proofs of disjunctions are similar to the ones presented in [KLLQ23, DKQ$^+$25], however, we do not use pairs of inverse group elements. Even though our challenge space is $\{-1, 1\}^\lambda$, for which we use multiplicative secret sharing in $\{-1, 1\}$, challenge components in $\{-1, 1\}$ will correspond to group exponents in $\{0, 1\}$: indeed, we observe that the elements in the secret sharing of c only determine the choice of one among two branches (linking the commitment either to Q or to P), and do not need to be understood literally as -1 and 1. This allows us to remove both the use of a quadratic twist from elliptic curves and the non-standard assumptions with particular secret key distributions used in [KLP25, DKQ$^+$25]. This approach is summarised in the following diagram,



in which the prover demonstrates the knowledge of one (corresponding to the index δ) among two group elements, linking Q_0 to P_0 and Q_1 to P_1. The $1-\delta$ branch is simulated by choosing its challenge beforehand and then sampling the corresponding commitment $Y_{1-\delta}$, for which the prover will know the linking group element. Then, as c is fixed by an honest verifier (or drawn from a hash function modelled as a random oracle), and this determines c_δ, the prover must demonstrate authentic knowledge of the group element linking P_δ to Q_δ. This is repeated λ times in parallel to obtain adequate soundness, yielding vectors $\boldsymbol{Y}_0, \boldsymbol{Y}_1$ of commitments and $\boldsymbol{c}_0, \boldsymbol{c}_1$ of challenges.

Blind Signatures. Our construction follows the [KLLQ23] general approach from OR proofs. Zero-knowledge proofs are inserted to allow the verification of the well-formedness of public keys and commitments, in order to prevent malicious attacks in which elements in the wrong orbits would be inserted in order to obtain distinguishers.

Given the element Q defined by the scheme and public keys (P_0, P_1), a valid signature is a witness-indistinguishable proof knowledge of one of the two elements linking P_0 and P_1 to Q. The main challenge of blind signature constructions is then to enable the user to hide their intended message and commitment to random coins for the signature when communicating with the signer, and then to unblind them when releasing their final signature.

To this aim, the signer generates *two* OR proofs of knowledge of witness either for P_0 or P_1, that are linked in the sense that using one branch (selected by the challenge) in one of them will imply using the other one in the other OR proof. The signer's first message consists of the respective commitments $\boldsymbol{W}_0^{(1)}$, $\boldsymbol{W}_1^{(1)}$, and $\boldsymbol{W}_0^{(-1)}$, $\boldsymbol{W}_1^{(-1)}$ of these (1) and (-1) sigma-protocol OR proofs (corresponding to the Y_0, Y_1 in the above diagram). Then, the user also draws random commitments, in a way to would allow her to simulate knowledge of group element linking $\boldsymbol{W}_0^{(1)}$ to $\boldsymbol{W}_0^{(-1)}$ for a challenge $\boldsymbol{d}_0$, and $\boldsymbol{W}_1^{(1)}$ to $\boldsymbol{W}_1^{(-1)}$ for a challenge $\boldsymbol{d}_1$. She then draws a challenge depending on her personal message and commitments, and sends a secret share $\boldsymbol{c}'$ of this challenge masked with $\boldsymbol{d}_0$ and $\boldsymbol{d}_1$ to the signer. The signer then uses $\boldsymbol{c}'$ as the challenge for both of her two linked sigma OR proofs using opposite challenge branches, and returns these proofs to the user. To build the final unblinded signature, the user multiplies the elements in her simulated proof of knowledge with the ones returned by the signer, and this composition yields the unmasked signature.

The reader will find a diagram summarising this overview in Figure 1, in which c^*_δ is determined as $c^*_\delta = c \cdot c_{1-\delta} \cdot d_\delta = c' \cdot c^*_{1-\delta}$.

Strong Designated-Verifier Signatures with Non-Delegatability. The authors of [MTT25] combine an OR proof with a Diffie-Hellman type of key exchange adapted to the group action setting: however, it relies on group commutativity, and is hence not applicable to code or lattice equivalence-based instantiations.

We circumvent this inconvenience by using a Key-Encapsulation Mechanism (KEM) to establish a common secret between the verifier and signer. Moreover, lattice-isomorphism group actions do not allow uniform sampling on the group,

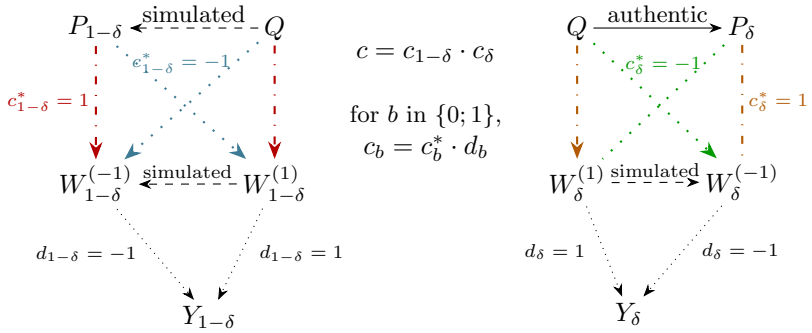


Figure 2. Description of the building-blocks of our blind signatures

which is infinite. As a consequence, our OR proofs' distributions depend on the public keys in the statement, which obliges us to hide them in order to grant privacy of the signer's identities, allowing the signatures to be anonymous in the eyes of external observers. To this aim, we use the session-key provided by the KEM to encrypt sensitive elements in the OR proof, in a KEM-DEM paradigm [Sho01]. This requires a careful adaptation to still be able to extract signer secret-keys from non-verifier-generated signatures, in order to obtain non-delegatability.

Open Directions for Future Work. Interestingly, the distribution of our OR proofs depend on the relations chosen in their language: in other words, when we prove the knowledge of one out of two secret keys corresponding to two public keys, the distribution of the proof depends on the public keys. Comparing the lattice isomorphism group action to instantiations from previous works, this appears because of the infiniteness of the group, on which we do not use a uniform distribution.

1. One consequence of this observation is that removing the use of a key-encapsulation mechanism in our SDVSND does not seem straightforward.
2. Our construction verifies k-one-more unforgeability for logarithmically many sessions in the security parameter, and [BLL+21, DHP24, KLR24] show that opening λ concurrent sessions, for λ the security parameter, would allow successful ROS attacks to forge signatures. Future work will explore the obtention of unforgeability for polynomially many sessions, in the line of recent articles like [HLM+25].
3. We leave open an extension to partial blindness, which we believe should be obtained straightforwardly by the application of previous techniques.
4. Our OR proofs straightforwardly generalise to proving knowledge of one among polynomially many witnesses, and would hence provide ring signatures from the LIP group action.

2 Preliminaries

Notations. Let $\mathbb{Z}$, $\mathbb{N}$, $\mathbb{R}$ and $\mathbb{R}_+$ denote integers, natural integers, reals, and positive (or null) reals, respectively. For a finite set $\mathcal{X}$, let $|\mathcal{X}|$ denote the cardinality of $\mathcal{X}$, and $[n]$ the subset $\{1, 2, ..., n\} \subset \mathbb{N}$. Let 1^n denote the bit string of all ones of length n, for two bitstrings s and t, let $s\|t$ denote the concatenation of s and t, and $|s|$ the length of s. $\{0,1\}^*$ will be the set of bitstrings of finite length. If an algorithm Alg runs in polynomial time it will be called efficient and if it is also probabilistic, a probabilistic polynomial time algorithm (PPT). $\mathsf{out} \leftarrow \mathsf{Alg}$ and $\mathsf{out} \xleftarrow{\$} \mathsf{Alg}$ will denote the output of deterministic and probabilistic algorithms respectively; if $\mathcal{X}$ is a finite set then $x \xleftarrow{\$} \mathcal{X}$ denotes sampling from it uniformly at random. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}_+$ is called negligible if it decreases faster than any inverse polynomial function.

Unless specified otherwise, vectors will be vertical and denoted with bold letters while their components will be denoted in regular font and with a subscript index. If only the components v_i for a vector are defined, it is assumed that they implicitly correspond to a vector $\boldsymbol{v}$. Matrices will be denoted with capital letters. Given a matrix B, its transpose is denoted by $B^\top$ and B^* will denote its Gram-Schmidt orthogonalisation. For $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ its columns vectors, its norm will be defined as $\|B\| := \max_i\{\|\boldsymbol{b}_i\|_2\}$. $\mathcal{G}\ell_n(\mathbb{Q})$ will denote the general linear group of degree n over $\mathbb{Q}$, and $\mathcal{G}\ell_n(\mathbb{Z})$ its subgroup over $\mathbb{Z}$; similarly, $\mathcal{PG}\ell_n(\mathbb{Q})$ will denote the projective linear group over $\mathbb{Q}$, and $\mathcal{PG}\ell_n(\mathbb{Z})$ its subgroup over $\mathbb{Z}$. $\mathcal{S}_n^{>0}(\mathbb{R})$ will denote the set of real-valued positive definite symmetric matrices, also called quadratic forms. For a quadratic form Q in $\mathcal{S}_n^{>0}(\mathbb{R})$, B_Q will be the unique upper-triangular lattice basis for Q, *i.e.* such that $Q = B_Q^\top B_Q$, obtained by Cholesky decomposition, and the norm associated to Q on $\mathbb{R}^n$ will be defined as $\|\cdot\|_Q : \mathbf{x} \mapsto \sqrt{\mathbf{x}^\top Q \mathbf{x}}$. For $i \in [n]$, the i-th minimal distance of Q is defined as $\lambda_i(Q) = \arg\min_{r \in \mathbb{R}^+}\{\mathrm{rank}(\mathrm{span}_{\mathbb{R}}(\mathbb{Z}^n \cap \mathcal{B}_{Q,r})) = i\}$, for $\mathcal{B}_{Q,r}$ the origin-centered ball of radius r for the norm $\|\cdot\|_Q$.

2.1 Generalised Group Actions

Previous works introduced generic constructions from group actions, usually instantiated with isogenies. We however, in this work, will provide them for more general group actions than before. Namely, for cases in which the group is not commutative (we consider right group actions) and infinite, given a good sampling algorithm, with properties specified hereafter. This will allow instantiations of resulting constructions not only from isogenies, but also from the Lattice Isomorphism Problem.

Definition 1 (Group Action). *Let $\mathbb{G}$ be a group and let $\mathcal{Q}$ be a set. A right group action of $\mathbb{G}$ on $\mathcal{Q}$ is a map $* : \mathcal{Q} \times \mathbb{G} \to \mathcal{Q}$ such that:*

1. *there exists e in $\mathbb{G}$ such that for all Q in $\mathcal{Q}$, $*(Q, e) = Q$;*
2. *for all $g_1, g_2 \in \mathbb{G}$ and $Q \in \mathcal{Q}$, $*(Q, g_1 g_2) = *(*(Q, g_1), g_2)$.*

We usually denote $Q * g := *(Q, g)$ for $g \in \mathbb{G}$ and $Q \in \mathcal{Q}$.

Definition 2 (Orbit). *Let $\mathcal{Q} * \mathbb{G} \to \mathcal{Q}$ be a group action and let $Q \in \mathcal{Q}$ be a set element. The orbit of Q, denoted $[Q]$, is the set*

$$[Q] := \{Q * g \mid g \in \mathcal{G}\}.$$

Definition 3 (Stabiliser). *Let $\mathcal{Q} * GG \to \mathcal{Q}$ be a group action and let $Q \in \mathcal{Q}$ be a set element. The stabaliser of Q, denoted $\mathrm{Stab}_{\mathbb{G}}(Q)$, is the set*

$$\mathrm{Stab}_{\mathbb{G}}(Q) := \{g \in \mathbb{G} \mid Q * g = Q\}.$$

Definition 4 (Group Action Family). *A group action family with respect to an indexing set $\mathcal{I}$ is a collection of group actions parametrized by $\mathcal{I}$, denoted $(*_i : \mathcal{Q}_i \times \mathbb{G}_i \to \mathcal{Q}_i)_{i \in \mathcal{I}}$.*

Definition 5 (Free Group Action). *A group action $* : \mathcal{Q} \times \mathbb{G} \to \mathcal{Q}$ is free if there is a unique e in $\mathbb{G}$ such that, for any Q in $\mathcal{Q}$ and g in $\mathbb{G}$, $Q * g = Q$ iff g is equal to e.*

To be able to use a countably infinite group, this work requires the existence of a sampling algorithm that we call the instance generator, and specify hereafter.

Definition 6 (Instance Generator). *An instance generator IG for a group action family $(*_\lambda : \mathcal{Q}_\lambda \times \mathbb{G}_\lambda \to \mathcal{Q}_\lambda)_{\lambda \in \mathbb{N}}$ is a PPT algorithm that on input $\lambda \in \mathbb{N}$ and Q in $\mathcal{Q}_\lambda$ outputs, except with negligible probability in λ, (Q', g) in $\mathcal{Q}_\lambda \times \mathbb{G}_\lambda$ such that $Q *_\lambda g = Q'$, where the distribution of Q' depends only on $[Q]$, and g is uniform among the group elements which acting on Q yield Q'.*

Our constructions will rely on hardness of the Group Action Inverse Problem for a specified instance generator IG ($\mathrm{GAIP}_{\mathsf{IG}}$), where the GAIP stems from Couveigne's vectorization problem [Cou06, Problem 6].

Definition 7 (Group Action Inverse Problem). *Let IG be an instance generator, specifying for any λ in $\mathbb{N}$, group actions $*_\lambda : \mathcal{Q}_\lambda \times \mathbb{G}_\lambda \to \mathcal{Q}_\lambda$ and $Q_\lambda \in \mathcal{Q}_\lambda$. The $\mathrm{GAIP}_{\mathsf{IG}}$ will be called hard if for any PPT algorithm $\mathcal{A}$ and $(P, h) \xleftarrow{\$} \mathsf{IG}(Q_\lambda)$, there exists a negligible function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}_+$ such that*

$$\Pr\left[Y = P *_\lambda g' \;\middle|\; \begin{array}{l}(Y, g) \xleftarrow{\$} \mathsf{IG}(P) \\ g' \xleftarrow{\$} \mathcal{A}(Y, P)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

When the definition of the instance generator is clear from the context, the $\mathrm{GAIP}_{\mathsf{IG}}$ will sometimes be referred to as the GAIP.

Definition 8 (Useful Group Actions). *In the context of this work, a group action family $\mathcal{F} = (*_\lambda : \mathcal{Q}_\lambda \times \mathbb{G}_\lambda \to \mathcal{Q}_\lambda)_{\lambda \in \mathbb{N}}$ will be called useful iff*

1. *it admits an instance generator IG for $\mathcal{F}$, as specified in Definition 6;*
2. *for every natural λ, there is an algorithm with polynomial time in λ to evaluate $*_\lambda$, compute group operations in $\mathbb{G}_\lambda$, and to check whether an element is in $\mathbb{G}_\lambda$.*

Free useful group actions will refer to useful group action families in which all the actions are free.

2.2 The Lattice Isomorphism Group Action Family

Let n be a natural integer, and $\mathcal{S}_n^{>0}(\mathbb{R})$ the set of symmetric positive-definite matrices in $\mathbb{R}^{n\times n}$, corresponding to quadratic forms on $\mathbb{R}^n$. Given Q in $\mathcal{S}_n^{>0}(\mathbb{R})$, the LIP exhibits the action of the group $\mathcal{GL}_n(\mathbb{Z})$ on $[Q]$ which to Q' in $[Q]$ and U in $\mathcal{GL}_n(\mathbb{Z})$ associates $U^TQ'U$. In this article, the group $\mathcal{GL}_n(\mathbb{Z})$ will be restricted to $\mathcal{PGL}_n(\mathbb{Z})$.

Our original blind signature construction required the group action to be free; we have since lifted this constraint, but still provide the corresponding family of automorphism-free lattices as it could be of independent interest for other constructions.

Definition 9 (Lattice Isomorphism Group Actions). *In this article, given a natural integer n and Q in $\mathcal{S}_n^{>0}(\mathbb{R})$, the Lattice Isomorphism Group Action will refer to the action of $\mathcal{PGL}_n(\mathbb{Z})$ on $\mathcal{S}_n^{>0}(\mathbb{R})$, restricted to $* : \mathcal{PGL}_n(\mathbb{Z}) \times [Q] \to [Q], (U, R) \mapsto R * U := U^\top RU$.*

An Instance Generator. The reader will find an instantiation of an instance generator IG for which the $\mathrm{GAIP}_{\mathsf{IG}}$ hardness relies on the average-case search-LIP in Appendix A.

Free Lattice Isomorphism Group Actions. For a quadratic form Q in $\mathcal{S}_n^{>0}(\mathbb{R})$, we define

$$\mathrm{Aut}(Q) := \left\{U \in \mathcal{PGL}_n(\mathbb{Z}) | U^\top QU = Q\right\}$$

the automorphism group of Q, and say that Q is automorphism-free if $\mathrm{Aut}(Q)$ is trivial, *i. e.* reduced to the identity. Our original blind signature construction required freeness of the group action. We have since removed this constraint, but still provide our instantiation of automorphism-free lattices and a free LIP-based group action in Lemma 11, as it could be of independent interest for other constructions.

Lemma 10 (Free Lattice Isomorphism Group Actions, from [BBCK24], Proposition 3). *Let Q be in $\mathcal{S}_n^{>0}(\mathbb{R})$. Then, the action of the group $\mathcal{PGL}_n(\mathbb{Z})$ on $[Q]$ is free if and only if Q is automorphism-free.*

We provide a new instantiation of automorphism-free lattice isomorphism group-actions, described in the following lemma.

Lemma 11 (Automorphism-Free Quadratic Forms). *For n a natural integer,*

$$Q_n := \begin{pmatrix} n+1 & 1 & 1 & \dots & 1 \\ 1 & n+2 & 1 & \dots & 1 \\ \vdots & \ddots & \vdots & \ddots & 1 \\ 1 & 1 & \dots & 1 & 2n \end{pmatrix}$$

represents an automorphism-free quadratic form.

Proof. Let U be in $\mathrm{Aut}(Q_n)$, and $\boldsymbol{u}_k = (u_1, \ldots, u_n)$ be the k-th column of U. Then

$$\begin{aligned} \boldsymbol{u}_k^\top Q_n \boldsymbol{u}_k &= n + k \\ \sum_{i=1}^{n} (n+i) u_i^2 + 2 \sum_{j \neq i} u_j u_i &= n + k \\ \left(\sum_{i=1}^{n} (n+i-1) u_i^2 \right) + \left(\sum_{i=1}^{n} u_i \right)^2 &= n + k. \end{aligned}$$

The strict positivity of $\left(\sum_{i=1}^{n}(n+i-1)u_i^2\right) + \left(\sum_{i=1}^{n} u_i\right)^2$ for any non-zero $\boldsymbol{u}_k$ proves the positive definiteness of the symmetric matrix Q_n. Then, for each index i in $[n]$,

$$n u_i^2 \leq (n+i-1) u_i^2 \leq \sum_{j=1}^{n} (n+j-1) u_j^2 + \left(\sum_{j=1}^{n} u_j \right)^2 = n + k$$

so $|u_i| \leq 1$. Moreover, if there were $i \neq j$ in $[n]$ such that $|u_i| = |u_j| = 1$, one would have

$$2n + 1 \leq 2n + i + j - 2 \leq \sum_{j=1}^{n} (n+j-1) u_j^2 + \left(\sum_{j=1}^{n} u_j \right)^2 = n + k,$$

which is impossible so there is a unique index j such that

$$(n+j-1) u_j^2 + u_j^2 = n + k,$$

which is $j = k$, and denoting e_i the i-th vector of the canonical basis of $\mathbb{Z}^n$, $\boldsymbol{u}_k = b_k \boldsymbol{e}_k$, for some b_k in $\{-1, 1\}$. Thus, for any column index ℓ different from k,

$$\boldsymbol{u}_\ell^\top Q_n \boldsymbol{u}_k = 1 \iff b_j b_k = 1$$

and finally, U is equal to the identity in $\mathcal{PGL}_n(\mathbb{Z})$. □

2.3 Nondeterministic Polynomial Time (NP) OR Relations

Definition 12 (NP Relation). *Let X and W be sets. We call the elements in $\mathcal{X}$ statements. We assume that elements of $\mathcal{X}$ and $\mathcal{W}$ are represented by bit strings. A subset $\mathcal{R} \subset \mathcal{X} \times \mathcal{W}$ is an NP-relation if:*

- *$\mathcal{R}$ is polynomially bounded,* i. e. *there exists a polynomial $p \in \mathbb{N}[\lambda]$ such that $|w| < p(|x|)$ for all $(x, w) \in \mathcal{R}$;*
- *there exists a deterministic polynomial time algorithm that on input $(x, w) \in \mathcal{X} \times \mathcal{W}$ outputs 1 if $(x, w) \in \mathcal{R}$ and 0 otherwise.*

For an NP relation $\mathcal{R}$ and tuple (x, w) in $\mathcal{R}$, we say that w is a witness for x.

Definition 13 (NP OR Relation). *Let $\mathcal{R}_0 \subset \mathcal{X}_0 \times \mathcal{W}_0$ and $\mathcal{R}_1 \subset \mathcal{X}_1 \times \mathcal{W}_1$ be two NP relations. We define $\mathcal{X}^{\mathsf{OR}} = \mathcal{X}_0 \times \mathcal{X}_1$, $\mathcal{W}^{\mathsf{OR}} = (\{0\} \times \mathcal{W}_0) \cup (\{1\} \times \mathcal{W}_1)$ and the relation $\mathcal{R}^{\mathsf{OR}} \subset \mathcal{X}^{\mathsf{OR}} \times \mathcal{W}^{\mathsf{OR}}$ such that $((x_0, x_1), (b, w))$ is in $\mathcal{R}^{\mathsf{OR}}$ if and only if (x_b, w) is in $\mathcal{R}_b$. $\mathcal{R}^{\mathsf{OR}}$ is the NP OR-relation for $\mathcal{R}_0$ and $\mathcal{R}_1$.*

Sigma protocols are three-round protocols designed for an NP relation $\mathcal{R} \subset \mathcal{X} \times \mathcal{W}$ that allow a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that, given a public statement $x \in \mathcal{X}$, they know a witness $w \in \mathcal{W}$ such that (x, w) is in $\mathcal{R}$. We provide a formal definition in Appendix G.

2.4 Blind signatures

We now define the concept of blind signatures, for which Section 4 will provide a generalised group action instantiation.

Definition 14 (Blind Signature Scheme). *A blind signature scheme is a tuple of four algorithms* $(\mathsf{Gen}, \mathsf{BSign}, \mathsf{BUser}, \mathsf{Verify})$ *that function as described below.*

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pp}, \mathsf{pk}, \mathsf{sk})$**:** outputs public parameters pp, a secret key sk, used to sign messages, and a public key pk, to verify signatures.

$\mathsf{BSign}(\mathsf{sk})/\mathsf{BUser}(m, \mathsf{pk}) \to \sigma$**:** the algorithms BSign and BUser may interact during their execution. The algorithm BSign takes as input the secret key sk. The algorithm BUser, on input a message m an the public key pk, outputs a signature σ at the end of execution. We denote $\sigma \xleftarrow{\$} \langle \mathsf{BSign}(\mathsf{sk}), \mathsf{BUser}(\mathsf{pk}, m) \rangle$ the execution of these algorithms. As BSign and BUser may interact during their execution, we view the generation of σ as an alternation of signer and user algorithms. In this sequence we denote the i-th signer or user algorithm by BSign_i and BUser_i, respectively, and initiate it with BSign_1.

$\mathsf{Verify}(\mathsf{pk}, m, \sigma) \to b$**:** on input a signature σ, a message m and the public key pk, outputs a bit b equal to one if verification passes and else zero.

Definition 15 (Correctness). *A blind signature scheme satisfies correctness if there exists a negligible function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}_+$ such that for any message m and natural integer λ,*

$$\Pr\left[\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1 \;\middle|\; \begin{array}{c} (\mathsf{pp}, \mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda) \\ \sigma \xleftarrow{\$} \langle \mathsf{BSign}(\mathsf{sk}), \mathsf{BUser}(\mathsf{pk}, m) \rangle \end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Definition 16 (Blindness, adapted from [KLX22]). *A blind signature scheme satisfies blindness if there exists a negligible function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}_+$ such that for any PPT algorithm $\mathcal{A}$ and $\lambda \in \mathbb{N}$, for $\mathsf{Exp}^{\mathsf{Blind}}_{\mathcal{A}}$ the experiment described in Figure 3,*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{Blind}}_{\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda).$$

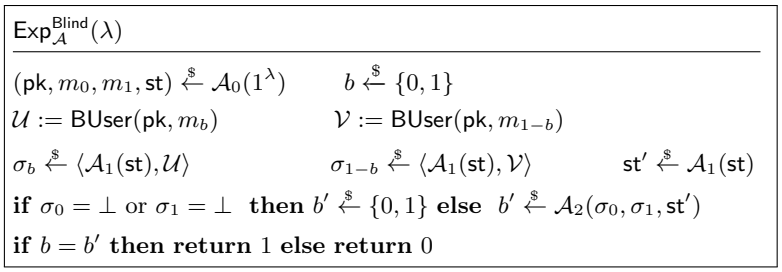


Figure 3. The blindness security game, in which the adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ has a one-time access to $\mathcal{U}$ and $\mathcal{V}$'s sequential algorithms (in the order it wishes to as long as this sequentiality is preserved), through $\mathcal{A}_1(\mathsf{st})$ which share a common internal state when interacting with them and then outputting the state st'.

Definition 17 (k-One-More Unforgeability, adapted from [KLX22]). *A blind signature scheme satisfies k-one-more unforgeability, with k in $\mathbb{N}$, if for any PPT algorithm $\mathcal{A}$, there exists a negligible function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}_+$ such that, for λ in $\mathbb{N}$ and $\mathsf{Exp}^{k-\mathsf{OMUF}}_{\mathcal{A}}(\lambda)$ the experiment described in Figure 4,*

$$\Pr\left[\mathsf{Exp}^{k-\mathsf{OMUF}}_{\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

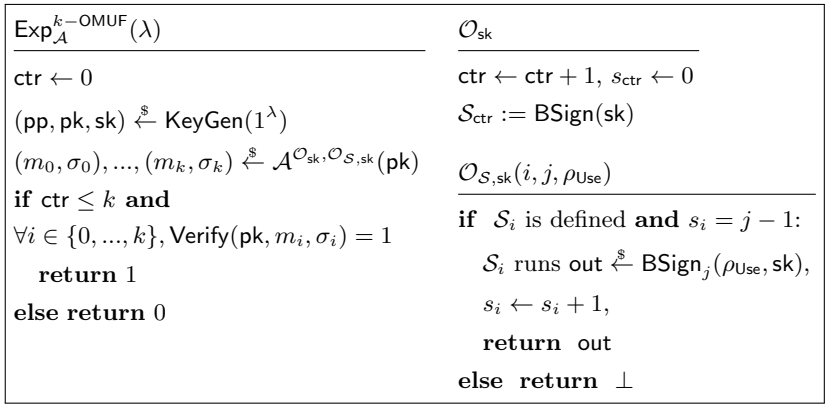


Figure 4. The k-one-more unforgeability game.

Proof Strategy for One-More Unforgeability. [KLX22] introduced convenient conditions implying one-more unforgeability for blind signatures, which are recalled, along with the corresponding formalisation, in Appendix C.

2.5 Strong Designated Verifier Signatures with Non-Delegatability

In Section 5, this work will introduce an instantiation of Strong Designated Verifier Signatures (SDVS) with non-delegatability based on the hardness of the

GAIP, inspired by the isogeny-based construction of [MTT25]. A strong designated verifier signature scheme is a tuple of algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, described below. The KeyGen algorithm is used to generate both signer and verifier keys.

$\mathsf{KeyGen}(1^\lambda, \ell) \to (\mathsf{pk}, \mathsf{sk})$**:** on input a security parameter λ in $\mathbb{N}$, and label ℓ in $\{s, v\}$ specifying whether the keys are intended for a signer of verifier, outputs a secret key sk and public key pk.

$\mathsf{Sign}(\mathsf{sk}_s, \mathsf{pk}_s, \mathsf{pk}_v, m) \to \sigma$**:** on input a message m and the keys $\mathsf{sk}_s, \mathsf{pk}_s, \mathsf{pk}_v$, outputs a signature σ.

$\mathsf{Verify}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m, \sigma) \to b$**:** on input a signature σ, message m and the keys sk_v, pk_s and pk_v, outputs a bit b, set to zero in case of rejection and one otherwise.

Definition 18 (Correctness). *A strong designated verifier signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *satisfies correctness if there exists a negligible function* $\mathsf{negl} : \mathbb{N} \to \mathbb{R}_+$ *such that for any natural integer* λ, $(\mathsf{pk}_s, \mathsf{sk}_s) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, s)$, $(\mathsf{pk}_v, \mathsf{sk}_v) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, v)$ *and message* m*,*

$$\Pr\left[\mathsf{Verify}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m, \sigma) = 1 \;\middle|\; \sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}_s, \mathsf{pk}_s, \mathsf{pk}_v, m)\right] = 1 - \mathsf{negl}(\lambda).$$

Definition 19 (Strong Unforgeability, from [MTT25]). *A strong designated verifier signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *satisfies strong unforgeability if for any PPT algorithm* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl} : \mathbb{N} \to \mathbb{R}_+$ *such that for any* $\lambda \in \mathbb{N}$*, and* $\mathsf{Exp}^{\mathsf{SUF}}_{\mathcal{A}}$ *the experiment defined in Figure 5,*

$$\Pr\left[\mathsf{Exp}^{\mathsf{SUF}}_{\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

$\mathsf{Exp}^{\mathsf{SUF}}_{\mathcal{A}}(\lambda)$	$\mathcal{O}^{\mathsf{Sign}}(m)$
$L^{\mathsf{Sign}} \xleftarrow{\$} \{\}$	$\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}_s, \mathsf{pk}_s, \mathsf{pk}_v, m)$
$(\mathsf{pk}_s, \mathsf{sk}_s) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, s)$	$L^{\mathsf{Sign}} \leftarrow L^{\mathsf{Sign}} \cup \{(m, \sigma)\}$
$(\mathsf{pk}_v, \mathsf{sk}_v) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, v)$	**return** σ
$(m^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}^{\mathsf{Sign}}, \mathcal{O}^{\mathsf{Verify}}}(\mathsf{pk}_s, \mathsf{pk}_v)$	$\mathcal{O}^{\mathsf{Verify}}(m, \sigma)$
if $(m^*, \sigma^*) \notin L^{\mathsf{Sign}}$ **and**	$b \leftarrow \mathsf{Verify}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m, \sigma)$
$\mathsf{Verify}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m^*, \sigma^*) = 1$	**return** b
return 1	
else return 0	

Figure 5. The strong unforgeability game.

Definition 20 (Perfect Non-Transferability, adapted from [SS08]). *A strong designated verifier signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *satisfies perfect non-transferability if there exists a PPT algorithm* Sim *such that for any natural*

integer λ, $(\mathsf{pk}_s, \mathsf{sk}_s) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, s)$, $(\mathsf{pk}_v, \mathsf{sk}_v) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, v)$, *and message* m, *the distributions on the outputs of* $\mathsf{Sign}(\mathsf{sk}_s, \mathsf{pk}_s, \mathsf{pk}_v, m)$ *and* $\mathsf{Sim}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m)$ *are equal.*

A Designated-Verifier Signature scheme is called *Strong* if it verifies privacy of signer identities, as defined hereafter.

Definition 21 (Privacy of Signer Identities, adapted from [LV05]). *A strong designated verifier signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *satisfies privacy of signer identities if for any PPT algorithm* $\mathcal{A}$, *there exists a negligible function* $\mathsf{negl} : \mathbb{N} \to \mathbb{R}_+$ *such that for any* λ *in* $\mathbb{N}$, *for* $\mathsf{Exp}^{\mathsf{PSI}}_{\mathcal{A}}$ *the experiment defined in Figure 6,*

$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{PSI}}_{\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

$\mathsf{Exp}^{\mathsf{PSI}}_{\mathcal{A}}(\lambda)$

$(\mathsf{pk}_0, \mathsf{sk}_0) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, s)$
$(\mathsf{pk}_1, \mathsf{sk}_1) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, s)$
$(\mathsf{pk}_v, \mathsf{sk}_v) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, v)$
$m^* \xleftarrow{\$} \mathcal{A}^{\mathcal{O}^{\mathsf{Sign}}, \mathcal{O}^{\mathsf{Verify}}}(\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{pk}_v)$
$b \xleftarrow{\$} \{0,1\}$, $\sigma^* \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{pk}_v, m^*)$
$b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}^{\mathsf{Sign}}, \mathcal{O}^{\mathsf{Verify}^*}}(\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{pk}_v, \sigma^*)$
if $b' = b$ **return** 1, **else return** 0

$\mathcal{O}^{\mathsf{Sign}}(\beta, m)$

return $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}_\beta, \mathsf{pk}_\beta, \mathsf{pk}_v, m)$

$\mathcal{O}^{\mathsf{Verify}}(\beta, m, \sigma)$

return $\beta \leftarrow \mathsf{Verify}(\mathsf{sk}_v, \mathsf{pk}_\beta, \mathsf{pk}_v, m, \sigma)$

$\mathcal{O}^{\mathsf{Verify}^*}(\beta, m, \sigma)$

if $(m, \sigma) = (m^*, \sigma^*)$ **return** $\perp$, **else**
return $\beta \leftarrow \mathsf{Verify}(\mathsf{sk}_v, \mathsf{pk}_\beta, \mathsf{pk}_v, m, \sigma)$

Figure 6. Experiment for the privacy of signer identities.

In this article's construction, the verifier's secret key will consist of one witness for an OR relation on two $\mathrm{GAIP}_{\mathsf{IG}}$ instances, alongside a KEM decapsulation key. Non-delegatability captures a notion showing that no signatures may be produced using neither signer nor verifier secret keys. In this work, this will mean without using a solution to a $\mathrm{GAIP}_{\mathsf{IG}}$ instance, disregarding the decapsulation key; we hence define the following adapted non-delegatability notion.

Definition 22 (Non-Delegatability). *An SDVS scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *satisfies non-delegatability with knowledge error* κ *if there exists a probabilistic oracle machine* Ext *such that for every PPT algorithm* $\mathcal{F}$, *there exists a polynomial* poly *in* $\mathbb{N}[X]$ *such that for any* m *in* $\{0,1\}^*$, λ *in* $\mathbb{N}$, $(\mathsf{pk}_s, \mathsf{sk}_s) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, s)$, $(\mathsf{pk}_v, \mathsf{sk}_v) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, v)$, *denoting*

$$\epsilon := \Pr\left[\mathsf{Verify}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m, \sigma) = 1 \mid \sigma \xleftarrow{\$} \mathcal{F}(m, \mathsf{pk}_s, \mathsf{pk}_v)\right],$$

and $\mathcal{F}_m$ *the restriction of* $\mathcal{F}$ *to the input* m, $\mathsf{Ext}^{\mathcal{F}_m}(m, \mathsf{pk}_s, \mathsf{pk}_v)$ *runs in expected polynomial time, and considering equality of the distributions of* Sign *on two*

inputs,

$$\Pr\left[\forall m, \mathsf{Sign}(\mathsf{sk}', \mathsf{pk}_s, \mathsf{pk}_v, m) = \mathsf{Sign}(\mathsf{sk}_s, \mathsf{pk}_s, \mathsf{pk}_v, m) \mid \mathsf{sk}' \xleftarrow{\$} \mathsf{Ext}^{\mathcal{F}_m}(m)\right] \geq (\epsilon - \kappa)\left(\frac{\epsilon}{\mathsf{poly}(\lambda)} - \kappa\right).$$

3 Zero-Knowledge Proofs Systems for the $\mathrm{GAIP}_{\mathsf{IG}}$

Standard zero-knowledge proofs may straightforwardly be obtained with sigma (*i.e.* three-way) commit and prove protocols analogous to discrete-logarithm based constructions, that may be made non-interactive using the Fiat-Shamir transform [FS87].

3.1 Zero-Knowledge Proofs Systems for the $\mathrm{GAIP}_{\mathsf{IG}}$

This section recalls the straightforward way to obtain a zero-knowledge proof of knowledge for a GAIP relation using a sigma protocol.

Definition 23 (GAIP NP Relation). *Let* IG *be an instance generator such that the* $\mathrm{GAIP}_{\mathsf{IG}}$ *is hard, and given* λ *in* $\mathbb{N}$, $* : \mathcal{Q} \times \mathbb{G} \to \mathcal{Q}$ *be the corresponding group action, and* Q *in* $\mathcal{Q}$. $\mathcal{R}_{Q,\lambda} := \{(Q', U) \in [Q] \times \mathbb{G} \mid Q' = Q * U\}$ *defines an NP relation that we will call the* $\mathrm{GAIP}_{\mathsf{IG}}$ *relation for* Q.

A sigma protocol for GAIP relations may be found in Appendix G.1 (Definition 60), as well as its non-interactive version obtained by a Fiat-Shamir transformation (Definition 62), and proofs for correctness, special soundness with error $2^{-\lambda}$, and perfect honest-verifier zero-knowledge (Theorem 61).

3.2 Zero-Knowledge OR Proofs Systems for the $\mathrm{GAIP}_{\mathsf{IG}}$

Definition 24 (GAIP NP OR Relation). *Let* $\mathcal{R}_{Q_0,\lambda}$ *and* $\mathcal{R}_{Q_1,\lambda}$ *be two relations satisfying Definition 23 for the* $\mathrm{GAIP}_{\mathsf{IG}}$*, and* $* : \mathcal{Q} \times \mathbb{G} \to \mathcal{Q}$ *be the corresponding group action for both* $\mathcal{R}_{Q_0,\lambda}$ *and* $\mathcal{R}_{Q_1,\lambda}$.

$$\mathcal{R}^{\mathsf{OR}}_{Q_0,Q_1,\lambda} := \{((x_0, x_1), (b, w)) \in ([Q_0] \times [Q_1]) \times (\{0,1\} \times \mathbb{G}) | (x_b, w) \in \mathcal{R}_{Q_b,\lambda}\}$$

defines an NP OR relation that we will call the $\mathrm{GAIP}_{\mathsf{IG}}$ *OR relation for* (Q_0, Q_1).

We now provide a sigma protocol for the NP OR relation $\mathcal{R}^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$ of Definition 24, following the general framework of [CDS94] generating these from a secret sharing in the challenge space and sigma protocols for each relation. This construction generalises directly to zero-knowledge proofs of knowledge of one among many statements.

Definition 25 (Sigma Protocol for $\mathrm{GAIP}_{\mathsf{IG}}$ OR Relations). *We define a three-way sigma protocol to prove, given a statement* x*, the knowledge of a witness* w *such that* (x, w) *is in* $\mathcal{R}^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$ *satisfying Definition 24, as the tuple of algorithms* $\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda} = (\mathsf{Commit}, \mathsf{Challenge}, \mathsf{Prove}, \mathsf{Verify})$ *described hereafter.*

$\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda}.\mathsf{Commit}(x,w) \to (\mathsf{st}_{\mathsf{Prove}}, \rho_{\mathsf{Prove}_1})$**:** on input (x,w):

1. parses $(\delta, A_\delta) \leftarrow w$.
2. samples $\boldsymbol{c}_{1-\delta} \xleftarrow{\$} \{-1,1\}^\lambda$ and, for $i \in [\lambda]$, $(Y_{\delta,i}, H_{\delta,i}) \xleftarrow{\$} \mathsf{IG}(Q_\delta)$ and $(Y_{1-\delta,i}, U_{1-\delta,i}) \xleftarrow{\$} \begin{cases} \mathsf{IG}(P_{1-\delta}) & \text{if } c_{1-\delta,i} = -1 \\ \mathsf{IG}(Q_{1-\delta}) & \text{if } c_{1-\delta,i} = 1 \end{cases}$.
3. outputs $\mathsf{st}_{\mathsf{Prove}} := (\boldsymbol{H}_\delta, \boldsymbol{c}_{1-\delta}, \boldsymbol{U}_{1-\delta})$ and $\rho_{\mathsf{Prove}_1} := (\boldsymbol{Y}_0, \boldsymbol{Y}_1)$.

$\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda}.\mathsf{Challenge}(x, \rho_{\mathsf{Prove}_1}) \to \rho_{\mathsf{Verify}}$**:** on input $(x, \rho_{\mathsf{Prove}_1})$ samples $\boldsymbol{c} \xleftarrow{\$} \{-1,1\}^\lambda$ and outputs $\rho_{\mathsf{Verify}} := \boldsymbol{c}$.

$\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda}.\mathsf{Prove}(x, w, \mathsf{st}_{\mathsf{Prove}}, \rho_{\mathsf{Verify}}) \to \rho_{\mathsf{Prove}_2}$**:** on input $(x, w, \mathsf{st}_{\mathsf{Prove}}, \rho_{\mathsf{Verify}})$:

1. parses $(\delta, A_\delta) \leftarrow w$, $(\boldsymbol{H}_\delta, \boldsymbol{c}_{1-\delta}, \boldsymbol{U}_{1-\delta}) \leftarrow \mathsf{st}_{\mathsf{Prove}}$ and $\boldsymbol{c} \leftarrow \rho_{\mathsf{Verify}}$.
2. for $i \in [\lambda]$, sets $\boldsymbol{c}_{\delta,i} = c_i \cdot c_{1-\delta,i}$ and $U_{\delta,i} = \begin{cases} A_\delta^{-1} \cdot H_{\delta,i} & \text{if } c_{\delta,i} = -1 \\ H_{\delta,i} & \text{if } c_{\delta,i} = 1 \end{cases}$,
3. outputs $\rho_{\mathsf{Prove}_2} := (\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{U}_0, \boldsymbol{U}_1)$.

$\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda}.\mathsf{Verify}(x, \rho_{\mathsf{Prove}_1}, \rho_{\mathsf{Verify}}, \rho_{\mathsf{Prove}_2}) \to b$**:** on input $(x, \mathsf{st}_{\mathsf{Verify}}, \rho_{\mathsf{Prove}_2})$ parses $(P_0, P_1) \leftarrow x$, $(\boldsymbol{Y}_0, \boldsymbol{Y}_1) \leftarrow \mathsf{st}_{\mathsf{Prove}_1}$, $\boldsymbol{c} \leftarrow \rho_{\mathsf{Verify}}$ and $(\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{U}_0, \boldsymbol{U}_1) \leftarrow \rho_{\mathsf{Prove}_2}$. If for each i in $[\lambda]$:

$c_{0,i} \cdot c_{1,i} = c_i$, and $\forall b \in \{0,1\}$, $\begin{cases} P_b * U_{b,i} = Y_{b,i} & \text{if } c_{b,i} = -1 \\ Q_b * U_{b,i} = Y_{b,i} & \text{if } c_{b,i} = 1 \end{cases}$,

then the algorithm outputs $b \leftarrow 1$, else it returns $b \leftarrow 0$.

We prove the correctness of $\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$ in Appendix G.2. For $b \in \{0,1\}$, we provide algorithms Ext_b in Figure 7 that act as special soundness witness extractors for a singular branch of $\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$. In Lemma 67 (Appendix G.3) we provide conditions under which thy are successful so they may be used generally.

$\mathsf{Ext}_b(\sigma = (\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{U}_0, \boldsymbol{U}_1),\ \sigma' = (\boldsymbol{c}'_0, \boldsymbol{c}'_1, \boldsymbol{U}'_0, \boldsymbol{U}'_1))$

if $\boldsymbol{c}_0 = \boldsymbol{c}'_0$ and $\boldsymbol{c}_1 = \boldsymbol{c}'_1$, **return** $\perp$
else $\exists t \in [\lambda], b \in \{0,1\} \mid c_{b,t} \neq c'_{b,t}$:
  if $c_{b,t} = -1$, **return** $U'_{b,t} \cdot U_{b,t}^{-1}$
  else $c_{b,t} = 1$: **return** $U_{b,t} \cdot U'^{-1}_{b,t}$

Figure 7. The witness extractors Ext_b for $b \in \{0,1\}$.

Theorem 26 (Special Soundness for $\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$). *The $\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$ protocol satisfies special soundness with soundness error $2^{-\lambda}$.*

Proof. See Appendix G.3.

Theorem 27 (Perfect Honest-Verifier Zero Knowledge of $\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$). *The $\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$ protocol satisfies perfect honest-verifier zero knowledge,* i.e. *there exists an algorithm* Sim *that on input $\lambda \in \mathbb{N}$ and a statement $x = (P_0, P_1) \in [Q_0] \times [Q_1]$, produces accepting transcripts for x matching the distribution of honestly generated transcripts.*

Proof. See Appendix G.4.

Definition 28 (Non-Interactive OR Proof System). *The protocol in Definition 25 can be made non-interactive using the Fiat-Shamir transform [FS87]. Let* $\Pi^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$*, consisting of algorithms* $(\mathsf{ComPr}, \mathsf{Verify})$ *denote the commit and prove scheme for which on input a statement* x *and witness* w *such that* (x, w) *is in* $\mathcal{R}^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$*,satisfying Definition 24,* ComPr *generates a non-interactive proof of knowledge of* w*, which may be checked with* Verify*.*

4 Blind Signatures from the $\mathrm{GAIP}_{\mathsf{IG}}$

This section presents a blind signature scheme whose security relies on this work's generalised $\mathrm{GAIP}_{\mathsf{IG}}$, using the corresponding zero-knowledge OR proof system described in Definition 25.

For λ a security parameter in $\mathbb{N}$, let $\mathcal{H} : \{0,1\}^* \to \{0,1\}^\lambda$ denote a hash function, and for a group action family $\mathcal{F}$ satisfying definition 8 with an instance generator IG. Π refers to the non-interactive proof system of Definition 62, Σ^{OR} to the OR proof system mentioned in Definition 28, and Π^{OR} to its non-interactive version. The $\mathsf{GABS} = (\mathsf{KeyGen}, \mathsf{Sign}_1, \mathsf{Use}_1, \mathsf{Sign}_2, \mathsf{Use}_2)$ blind signature scheme is defined hereafter.

$\mathsf{GABS.KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$**:** on input a natural integer λ, corresponding to the group action $* : [Q] \times \mathbb{G} \to [Q]$ in $\mathcal{F}$, this algorithm samples $\delta \xleftarrow{\$} \{0,1\}$, $(P_0, A_0) \xleftarrow{\$} \mathsf{IG}(Q)$ and $(P_1, A_1) \xleftarrow{\$} \mathsf{IG}(Q)$, computes proofs $\pi_0 \xleftarrow{\$} \Pi_{Q,\lambda}.\mathsf{ComPr}(P_0, A_0)$, $\pi_1 \xleftarrow{\$} \Pi_{Q,\lambda}.\mathsf{ComPr}(P_1, A_1)$. It then sets and returns $\mathsf{sk} = (\delta, A_\delta)$ and $\mathsf{pk} = (\mathsf{pk}_0 := (P_0, \pi_0, \pi_1), \mathsf{pk}_1 := (P_1, \pi_0, \pi_1))$.

$\mathsf{GABS.Sign}_1(\mathsf{sk}, \mathsf{pk}) \to (\mathsf{st}_{\mathsf{Sign}}, \rho_{\mathsf{Sign},1})$**:** on input sk, pk:

1. parses $(\delta, A_\delta) \leftarrow \mathsf{sk}$ and $((P_0, \pi_0, \pi_1), (P_1, \pi'_0, \pi'_1)) \leftarrow \mathsf{pk}$;
2. aborts if $\pi_0 \neq \pi'_0$, $\pi_1 \neq \pi'_1$, π_0 does not verify, or π_1 does not verify;
3. samples $\boldsymbol{c}^*_{1-\delta} \xleftarrow{\$} \{-1,1\}^\lambda$;
4. for $i \in [\lambda]$:
   - samples $(W^{(1)}_{\delta,i}, H^{(1)}_{\delta,i}) \xleftarrow{\$} \mathsf{IG}(Q)$ and $(W^{(-1)}_{\delta,i}, H^{(-1)}_{\delta,i}) \xleftarrow{\$} \mathsf{IG}(Q)$;
   - sets, for $\beta \in \{-1,1\}$, $\pi^{(\beta)}_{\delta,i} \xleftarrow{\$} \Pi^{\mathsf{OR}}_{Q,P_\delta,\lambda}.\mathsf{ComPr}((W^{(\beta)}_{\delta,i}, W^{(\beta)}_{\delta,i}), (0, H^{(\beta)}_{\delta,i}))$,
     $(W^{(1)}_{1-\delta,i}, U^{*(1)}_{1-\delta,i}) \xleftarrow{\$} \begin{cases} \mathsf{IG}(P_{1-\delta}) & \text{if } c^*_{1-\delta,i} = -1 \\ \mathsf{IG}(Q) & \text{if } c^*_{1-\delta,i} = 1 \end{cases}$,
     $(W^{(-1)}_{1-\delta,i}, U^{*(-1)}_{1-\delta,i}) \xleftarrow{\$} \begin{cases} \mathsf{IG}(Q) & \text{if } c^*_{1-\delta,i} = -1 \\ \mathsf{IG}(P_{1-\delta}) & \text{if } c^*_{1-\delta,i} = 1 \end{cases}$, and
     $\pi^{(\beta)}_{\delta-1,i} \xleftarrow{\$} \Pi^{\mathsf{OR}}_{Q,P_{1-\delta},\lambda}.\mathsf{ComPr}\left(\left(W^{(\beta)}_{1-\delta,i}, W^{(\beta)}_{1-\delta,i}\right), \left(\frac{1-\beta\cdot c^*_{1-\delta,i}}{2}, U^{*(\beta)}_{1-\delta,i}\right)\right)$;
5. sets $\mathsf{st}_{\mathsf{Sign}} = (\boldsymbol{c}^*_{1-\delta}, (\boldsymbol{H}^{(\beta)}_\delta, \boldsymbol{U}^{*(\beta)}_{1-\delta})_{\beta\in\{-1;1\}})$, $\rho_{\mathsf{Sign},1} = (\boldsymbol{W}^{(\beta)}_b, \boldsymbol{\pi}^{(\beta)}_b)_{b,\beta}$, and outputs $(\mathsf{st}_{\mathsf{Sign}}, \rho_{\mathsf{Sign},1})$.

$\mathsf{GABS.Use}_1(m, \rho_{\mathsf{Sign},1}) \to (\mathsf{st}_{\mathsf{Use}}, \rho_{\mathsf{Use}})$**:** on input $m \in \{0,1\}^*$, and first-signer message $\rho_{\mathsf{Sign},1}$:

1. parses $(\boldsymbol{W}^{(\beta)}_b, \boldsymbol{\pi}^{(\beta)}_b)_{b\in\{0,1\},\beta\in\{-1;1\}} \leftarrow \rho_{\mathsf{Sign},1}$;

2. if $\exists b \in \{0,1\}$, $\beta \in \{-1,1\}$, $i \in [\lambda]$ such that $\pi_{b,i}^{(\beta)}$ does not verify, aborts;
3. for $b \in \{0,1\}, i \in [\lambda]$ samples $\boldsymbol{d}_b \xleftarrow{\$} \{-1,1\}^\lambda$, $(Y_{b,i}, V_{b,i}) \xleftarrow{\$} \mathsf{IG}\left(W_{b,i}^{(d_{b,i})}\right)$;
4. sets $\boldsymbol{c} \leftarrow \mathcal{H}(\boldsymbol{Y}_0 \| \boldsymbol{Y}_1 \| m)$, $\boldsymbol{c}' := (c_k \cdot d_{0,k} \cdot d_{1,k})_{k\in[\lambda]}$, $\mathsf{st}_{\mathsf{Use}} := (\boldsymbol{d}_b, \boldsymbol{V}_b, \boldsymbol{Y}_b)_b$ and $\rho_{\mathsf{Use}} = \boldsymbol{c}'$;
5. outputs $(\mathsf{st}_{\mathsf{Use}}, \rho_{\mathsf{Use}})$.

$\mathsf{GABS.Sign}_2(\mathsf{sk}, \mathsf{st}_{\mathsf{Sign}}, \rho_{\mathsf{Use}}) \to \rho_{\mathsf{Sign},2}$**:** on input sk, $\mathsf{st}_{\mathsf{Sign}}$ and ρ_{Use}:

1. parses $(\delta, A_\delta) \leftarrow \mathsf{sk}$, $(\boldsymbol{c}^*_{1-\delta}, (\boldsymbol{H}^{(\beta)}_\delta, \boldsymbol{U}^{*(\beta)}_{1-\delta})_{\beta\in\{-1;1\}}) \leftarrow \mathsf{st}_{\mathsf{Sign}}$, $\boldsymbol{c}' \leftarrow \rho_{\mathsf{Use}}$;
2. for $i \in [\lambda]$ sets: $\boldsymbol{c}^*_{\delta,i} \leftarrow c'_i \cdot c^*_{1-\delta,i}$, $U^{*(1)}_{\delta,i} \leftarrow A_\delta^{(c^*_{\delta,i}-1)/2} \cdot H^{(1)}_{\delta,i}$ and $U^{*(-1)}_{\delta,i} \leftarrow A_\delta^{-(c_{\delta,i}+1)/2} \cdot H^{(-1)}_{\delta,i}$;
3. outputs $\rho_{\mathsf{Sign},2} := (\boldsymbol{c}^*_b, \boldsymbol{U}^{*(1)}_b, \boldsymbol{U}^{*(-1)}_b)_{b\in\{0,1\}}$.

$\mathsf{GABS.Use}_2(\mathsf{st}_{\mathsf{Use}}, \rho_{\mathsf{Sign},2}, m, \mathsf{pk}) \to \sigma$**:** on input $\mathsf{st}_{\mathsf{Use}}$, $\rho_{\mathsf{Sign},2}$, m and pk:

1. parses $((P_0,\pi_0,\pi_1),(P_1,\pi'_0,\pi'_1)) \leftarrow \mathsf{pk}$, $(\boldsymbol{d}_b, \boldsymbol{V}_b, \boldsymbol{Y}_b)_{b\in\{0,1\}} \leftarrow \mathsf{st}_{\mathsf{Use}}$ and $(\boldsymbol{c}^*_b, \boldsymbol{U}^{*(1)}_b, \boldsymbol{U}^{*(-1)}_b)_{b\in\{0,1\}} \leftarrow \rho_{\mathsf{Sign},2}$;
2. aborts if $\pi_0 \neq \pi'_0$, $\pi_1 \neq \pi'_1$, or π_0 or π_1 does not verify;
3. aborts if $\exists b \in \{0,1\}, i \in [\lambda] \mid c^*_{b,i} \notin \{-1,1\}$ or $U^{*(1)}_{b,i} \notin \mathbb{G}$ or $U^{*(-1)}_{b,i} \notin \mathbb{G}$;
4. for $b \in \{0,1\}$ and $i \in [\lambda]$, sets $c_{b,i} \leftarrow c^*_{b,i} \cdot d_{b,i}$ and $U_{b,i} \leftarrow U^{*(d_{b,i})}_{b,i} \cdot V_{b,i}$,
5. aborts if $\exists b \in \{0,1\}$, $i \in [\lambda] \mid Y_{b,i} \neq \begin{cases} P_b * U_{b,i} & \text{if } c_{b,i} = -1 \\ Q * U_{b,i} & \text{if } c_{b,i} = 1 \end{cases}$;
6. outputs $\sigma := (\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{U}_0, \boldsymbol{U}_1)$.

$\mathsf{GABS.Verify}(\mathsf{pk}, m, \sigma) \to b$**:** on input pk, m and σ:

1. parses $((P_0,\pi_0,\pi_1),(P_1,\pi'_0,\pi'_1)) \leftarrow \mathsf{pk}$ and $(\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{U}_0, \boldsymbol{U}_1) \leftarrow \sigma$;
2. if $\pi_0 \neq \pi'_0$, $\pi_1 \neq \pi'_1$, π_0 or π_1 does not verify, or $\exists b \in \{0,1\}$, $i \in [\lambda]$ such that $U_{b,i} \notin \mathbb{G}$, outputs 0;
3. for $b \in \{0,1\}$ and $i \in [\lambda]$, sets $Y_{b,i} \leftarrow \begin{cases} P_b * U_{b,i} & \text{if } c_{b,i} = -1 \\ Q * U_{b,i} & \text{if } c_{b,i} = 1 \end{cases}$,
4. computes $\boldsymbol{c} \leftarrow \mathcal{H}(\boldsymbol{Y}_0 \| \boldsymbol{Y}_1 \| m)$;
5. if for each $i \in [\lambda]$, $c_i = c_{0,i} \cdot c_{1,i}$, outputs 1, else 0.

Proofs of the correctness, blindness under chosen keys, and one-more unforgeability of GABS may be found in Appendices H.1, H.2 and H.3 respectively.

5 Strong Designated Verifier Signatures with Non-Delegatability from the GAIP$_{\mathsf{IG}}$

Using our generic zero-knowledge OR-proof in 25 we extend the construction of [MTT25] to rely on a GAIP$_{\mathsf{IG}}$ for more general groups, non necessarily commutative nor allowing uniform distributions. For λ a security parameter in $\mathbb{N}$ and a (not necessarily free) group action family satisfying Definition 8 for the GAIP$_{\mathsf{IG}}$, let $* : \mathcal{Q} \times \mathbb{G} \to \mathcal{Q}$ denote the one corresponding to λ, with $Q \in \mathcal{Q}$ specified by IG. Let $\mathcal{R}_{Q,\lambda}$ denote the GAIP$_{\mathsf{IG}}$ relation for $* : [Q] \times \mathbb{G} \to [Q]$ and let $\mathcal{R}^{\mathsf{OR}}_{Q,Q,\lambda}$

denote the NP OR-relation constructed using two $\mathcal{R}_{Q,\lambda}$ relations. Let $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ be a key encapsulation mechanism satisfying correctness and SK-CCA-security (Appendix D). Let $\mathsf{SKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a symmetric key encryption scheme satisfying correctness and IND-EAV-security (Appendix E). We define a strong designated verifier signature scheme $\mathsf{GASDVS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ hereafter, based on the OR-proof system $\Sigma^{\mathsf{OR}}_{Q,Q,\lambda}$ (Definition 25) and encryption under SKE so that verification is only possible with knowledge of the encryption key.

$\mathsf{GASDVS.KeyGen}(1^\lambda, \ell)$**:** on input a natural integer λ and a label $\ell \in \{s, v\}$. If $\ell = s$ the algorithm samples $(P_s, A_s) \xleftarrow{\$} \mathsf{IG}(Q)$ and returns $(\mathsf{pk}_s := P_s, \mathsf{sk}_s := A_s)$. If $\ell = v$, the algorithm samples $(P_v, A_v) \xleftarrow{\$} \mathsf{IG}(Q)$ samples $(\mathsf{pk}_\mathsf{K}, \mathsf{sk}_\mathsf{K}) \xleftarrow{\$} \mathsf{KEM.Gen}(1^\lambda)$ and returns $(\mathsf{pk}_v := (P_v, \mathsf{pk}_\mathsf{K}), \mathsf{sk}_v = (A_v, \mathsf{sk}_\mathsf{K}))$.

$\mathsf{GASDVS.Sign}(\mathsf{sk}_s, \mathsf{pk}_s, \mathsf{pk}_v, m) \to (\sigma, E)$**:** provided the keys $(\mathsf{sk}_s, \mathsf{pk}_s, \mathsf{pk}_v)$ and a message m, the algorithm:

1. parses $A_s \leftarrow \mathsf{sk}_s$, $P_s \leftarrow \mathsf{pk}_s$, $(P_v, \mathsf{pk}_\mathsf{K}) \leftarrow \mathsf{pk}_v$;
2. sets $x = (P_s, P_v)$ and $w = (0, A_s)$
3. computes $(\mathsf{st}, \rho = (\boldsymbol{Y}_s, \boldsymbol{Y}_v)) \xleftarrow{\$} \Sigma^{\mathsf{OR}}_{Q,Q,\lambda}.\mathsf{Commit}(x, w)$.
4. sets $(K, E) \xleftarrow{\$} \mathsf{KEM.Encaps}(\mathsf{pk}_\mathsf{K})$ and $\boldsymbol{e} = \mathcal{H}(m \| K \| E \| \boldsymbol{Y}_s \| \boldsymbol{Y}_v)$.
5. computes $(\boldsymbol{c}_s, \boldsymbol{c}_v, \boldsymbol{U}_s, \boldsymbol{U}_v) \xleftarrow{\$} \Sigma^{\mathsf{OR}}_{Q,Q,\lambda}.\mathsf{Prove}(x, w, \mathsf{st}, \boldsymbol{e})$.
6. sets $V \xleftarrow{\$} \mathsf{SKE.Enc}(K, (\boldsymbol{U}_s, \boldsymbol{U}_v))$, sets $\sigma \leftarrow (\boldsymbol{c}_s, \boldsymbol{c}_v, V, \boldsymbol{Y}_s, \boldsymbol{Y}_v)$ and returns (σ, E).

$\mathsf{GASDVS.Verify}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m, \sigma, E) \to b$**:** on input $(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v)$, a message m, a signature σ and an encapsulation E, the algorithm:

1. parses $(A_v, \mathsf{sk}_\mathsf{K}) \leftarrow \mathsf{sk}_v$, $P_s \leftarrow \mathsf{pk}_s$, $(P_v, \mathsf{pk}_\mathsf{K}) \leftarrow \mathsf{pk}_v$ and $(\boldsymbol{c}_s, \boldsymbol{c}_v, V, \boldsymbol{Y}_s, \boldsymbol{Y}_v) \leftarrow \sigma$;
2. sets $K \leftarrow \mathsf{KEM.Decaps}(\mathsf{sk}_\mathsf{K}, E)$ and $(\boldsymbol{U}_s, \boldsymbol{U}_v) \leftarrow \mathsf{SKE.Dec}(K, V)$;
3. for $i \in [\lambda]$, sets $Y'_{s,i} \leftarrow \begin{cases} P_s * U_{s,i} & \text{if } c_{s,i} = -1 \\ Q * U_{s,i} & \text{if } c_{s,i} = 1 \end{cases}$, sets $Y'_{v,i} \leftarrow \begin{cases} P_v * U_{v,i} & \text{if } c_{v,i} = -1 \\ Q * U_{v,i} & \text{if } c_{v,i} = 1 \end{cases}$, and $\boldsymbol{e} \leftarrow \mathcal{H}(m \| K \| E \| \boldsymbol{Y}_s \| \boldsymbol{Y}_v)$;
4. if $\boldsymbol{Y}'_s = \boldsymbol{Y}_s$, $\boldsymbol{Y}'_v = \boldsymbol{Y}_v$ and for all $i \in [\lambda]$, $e_i = c_{s,i} \cdot c_{v,i}$, returns 1, else outputs 0.

Proofs of the correctness, non-transferability, strong unforgeability, privacy of signer identities and non-delegatability of GASDVS may be found in respective Appendices I.1, I.2, I.3, I.4 and I.5.

Acknowledgements. We thank Julia Kastner for helpful discussions. Paola de Perthuis was supported by the NWO Gravitation Project QSC.

References

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

BBCK24. Benjamin Benčina, Alessandro Budroni, Jesús-Javier Chi-Domínguez, and Mukul Kulkarni. Properties of lattice isomorphism as a cryptographic group action. In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *PQCrypto 2024, Part I*, pages 170–201. Springer, Cham, June 2024.

BdPP23. Théophile Brézot, Paola de Perthuis, and David Pointcheval. Covercrypt: an efficient early-abort kem for hidden access policies with traceability from the ddh and lwe. In Gene Tsudik, Mauro Conti, Kaitai Liang, and Georgios Smaragdakis, editors, *ESORICS 2023, Part I*, volume 14344 of *LNCS*, pages 372–392, September 2023.

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

BLP+13. Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013.

BN06. Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006.

BY91. Gilles Brassard and Moti Yung. One-way group actions. In Alfred J. Menezes and Scott A. Vanstone, editors, *CRYPTO'90*, volume 537 of *LNCS*, pages 94–107. Springer, Berlin, Heidelberg, August 1991.

CDS94. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Berlin, Heidelberg, August 1994.

Cou06. Jean-Marc Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291, 2006.

DHP24. Khue Do, Lucjan Hanzlik, and Eugenio Paracucchi. M&M'S: Mix and match attacks on Schnorr-type blind signatures with repetition. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 363–387. Springer, Cham, May 2024.

DKQ+25. Dung Hoang Duong, Xuan Thanh Khuc, Youming Qiao, Willy Susilo, and Chuanqi Zhang. Blind signatures from cryptographic group actions. Cryptology ePrint Archive, Report 2025/397, 2025.

DvW22. Léo Ducas and Wessel P. J. van Woerden. On the lattice isomorphism problem, quadratic forms, remarkable lattices, and cryptography. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 643–673. Springer, Cham, May / June 2022.

FO13. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, January 2013.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987.

HLM+25. Lucjan Hanzlik, Yi-Fu Lai, Marzio Mula, Eugenio Paracucchi, Daniel Slamanig, and Gang Tang. Tanuki: New frameworks for (concurrently secure) blind signatures from post-quantum group actions. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part IV*, volume 16248 of *LNCS*, pages 35–69. Springer, Singapore, December 2025.

KLLQ23. Shuichi Katsumata, Yi-Fu Lai, Jason T. LeGrow, and Ling Qin. CSI-Otter: Isogeny-based (partially) blind signatures from the class group action with a twist. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part III*, volume 14083 of *LNCS*, pages 729–761. Springer, Cham, August 2023.

KLP25. Veronika Kuchta, Jason T. LeGrow, and Edoardo Persichetti. Post-quantum blind signatures from matrix code equivalence. Cryptology ePrint Archive, Report 2025/274, 2025.

KLR24. Shuichi Katsumata, Yi-Fu Lai, and Michael Reichle. Breaking parallel ROS: Implication for isogeny and lattice-based blind signatures. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part I*, volume 14601 of *LNCS*, pages 319–351. Springer, Cham, April 2024.

KLX22. Julia Kastner, Julian Loss, and Jiayu Xu. The Abe-Okamoto partially blind signature scheme revisited. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 279–309. Springer, Cham, December 2022.

KTAT20. Tomoki Kawashima, Katsuyuki Takashima, Yusuke Aikawa, and Tsuyoshi Takagi. An efficient authenticated key exchange from random self-reducibility on CSIDH. In Deukjo Hong, editor, *ICISC 20*, volume 12593 of *LNCS*, pages 58–84. Springer, Cham, December 2020.

LV05. Fabien Laguillaumie and Damien Vergnaud. Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In Carlo Blundo and Stelvio Cimato, editors, *SCN 04*, volume 3352 of *LNCS*, pages 105–119. Springer, Berlin, Heidelberg, September 2005.

MTT25. Hiroki Minamide, Keisuke Tanaka, and Masayuki Tezuka. Strong designated verifier signatures with non-delegatability from CSIDH. In *International Conference on Information Security and Cryptology, ICISC 2025*, 2025.

PS00. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.

Sah99. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999.

Sha79. Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

Sho01. Victor Shoup. A proposal for an iso standard for public key encryption. https://shoup.net/papers/iso-2_1.pdf, December 2001.

SS08. Siamak Fayyaz Shahandashti and Reihaneh Safavi-Naini. Construction of universal designated-verifier signatures and identity-based signatures from standard signatures. In Ronald Cramer, editor, *PKC 2008*, volume 4939 of *LNCS*, pages 121–140. Springer, Berlin, Heidelberg, March 2008.

SS13. Nicolas Sendrier and Dimitris E. Simos. The hardness of code equivalence over and its application to code-based cryptography. In Philippe Gaborit, editor, *PQCrypto 2013*, pages 203–216. Springer, Berlin, Heidelberg, June 2013.

Appendices

A An Instance Generator for the Lattice Isomorphism Problem

For Q in $\mathcal{S}_n^{>0}(\mathbb{R})$, s a strictly positive real number, we define on the orbit of Q the following distribution: $\rho_{Q,s} : \mathbb{R}^n \to \mathbb{R}_+,\ \mathbf{x} \mapsto e^{-\pi \cdot ||\mathbf{x}||_Q^2 s^{-2}}$, and, as the sum of its evaluations on $\mathbb{Z}^n$ converges, denote $\rho_{Q,s}(\mathbb{Z}^n) := \sum_{\mathbf{x} \in \mathbb{Z}^n} \rho_{Q,s}(\mathbf{x})$.

Definition 29 (Discrete Gaussian distribution). *The origin-centred discrete Gaussian distribution $\mathcal{D}_{Q,s}$ satisfies*

$$\Pr_{X \xleftarrow{\$} \mathcal{D}_{Q,s}}[X = \mathbf{x}] = \begin{cases} \frac{\rho_{Q,s}(\mathbf{x})}{\rho_{Q,s}(\mathbb{Z}^n)} & \text{if } \mathbf{x} \in \mathbb{Z}^n \\ 0 & \text{otherwise} \end{cases}.$$

Lemma 30 (Efficient Discrete Sampling, from [BLP+13], Lemma 2.3). *Let Q be in $\mathcal{S}_n^{>0}(\mathbb{R})$. [BLP+13] provides a polynomial-time algorithm to sample from $\mathcal{D}_{Q,s}$ if $s \geq ||B_Q^*|| \cdot \sqrt{\log(2n+4)/\pi}$.*

Let IG be the instance generating algorithm, originally presented in [DvW22] and defined as in Algorithm 1. The algorithm takes as input a quadratic form Q and a real number $s > 0$.

$\mathsf{IG}(Q, s)$

If $Q \notin \mathcal{S}_n^{>0}(\mathbb{R})$ **or** $s < \max\left\{\lambda_n(Q), \left\|B_Q^*\right\| \sqrt{\log(2n+4)/\pi}\right\}$, **abort**
else
  $C \leftarrow 1 - (1 + e^{-\pi})^{-1}, m \leftarrow \lceil \frac{2n}{C} \rceil$
  while $\mathrm{rank}(Y) < n$ **do**
    Sample $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m \xleftarrow{\$} \mathcal{D}_{Q,s}$ (*c. f.* Lemma 30)
    and set $Y := (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m)$.
  end
  Compute U in $\mathcal{PGL}_n(\mathbb{Z})$ such that $U^{-1}V$ is the canonical upper-triangular Hermite Normal Form of Y.
  return $(Q * U, U)$.
end

Algorithm 1: Lattice Isomorphism Group Action Inverse Problem Instance Generator IG, from [DvW22]. If the algorithm does not terminate within polynomial time, it returns $\perp$.

Ducas and Woerden [DvW22] then showed that the first component output by IG only depended on the class $[Q]$ and not its particular representative.

Lemma 31 (Independence and Efficiency of IG, from [DvW22], Lemma 3.4). *Let Q be in $\mathcal{S}_n^{>0}(\mathbb{R})$ and s be a real number greater than or equal to*

$\max\left\{\lambda_n(Q), \left\|B_Q^*\right\|\sqrt{\log(2n+4)/\pi}\right\}$. *Then* $\mathsf{IG}(Q,s)$ *runs in polynomial time and returns* $(R,U) \in [Q] \times \mathcal{PGL}_n(\mathbb{Z})$*, where* R *follows a distribution that we call* $\mathcal{D}_s([Q])$ *and only depends on* $[Q]$ *and* s*, not* Q *itself, and* U *satisfies* $R = U^\top QU$*.*

This sampler allows the definition of an average-case Group-Action Inverse Problem (GAIP), which is called the search Lattice Isomorphism Problem.

Definition 32 (ac-sLIP$_s^Q$, from [DvW22], Definition 3.7). *Let* Q *be in* $\mathcal{S}_n^{>0}(\mathbb{R})$*, and* s *be a real number greater than or equal to* $\lambda_n(Q)$ *and* $\left\|B_Q^*\right\|\sqrt{\log(2n+4)/\pi}$*. The corresponding average-case search lattice isomorphism problem, ac-sLIP*$_s^Q$ *is, given* $R \xleftarrow{\$} \mathcal{D}_s([Q])$*, to find* $U \in \mathcal{PGL}_n(\mathbb{Z})$ *such that* $R = U^\top QU$*.*

Lemma 33 (Hardness of GAIP$_{\mathsf{IG}}$). *Let, for any natural number* n*,* $Q_n \in \mathcal{S}_n^{>0}(\mathbb{R})$ *and* s_n *be arbitrarily set to be greater than or equal to* $\max\{\lambda_n(Q), \left\|B_Q^*\right\|\sqrt{\log(2n+4)/\pi}\}$*. The algorithm* IG*, taking for any* λ *in* $\mathbb{N}$*,* (Q_λ, s_λ) *as input fulfils Definition 6 of an instance generator, for which GAIP*$_{\mathsf{IG}}$ *is hard under the* $(sLIP_{s_n}^{Q_n})_{n\in\mathbb{N}}$ *assumptions.*

Proof. Indeed, for any natural integer λ and PPT adversary $\mathcal{A}$, one can straightforwardly define a PPT adversary $\mathcal{B}$ behaving as $\mathcal{A}$ such that $\Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{IG}}(\lambda) = 1\right]$ is exactly the advantage of $\mathcal{B}$ in the $\mathrm{sLIP}_{s_\lambda}^{Q_\lambda}$ game with security parameter λ.

B Background on Commit and Prove Zero-Knowledge Proofs

Sigma protocols are three-round protocols designed for an NP relation $\mathcal{R} \subset \mathcal{X} \times \mathcal{W}$. They allow a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that, given a public statement $x \in \mathcal{X}$, they know a witness $w \in \mathcal{W}$ such that (x,w) is in $\mathcal{R}$.

Definition 34 (Sigma Protocol). *Let* $\mathcal{C}$ *denote a challenge space. A sigma protocol for the relation* $\mathcal{R} \subset \mathcal{X} \times \mathcal{W}$ *is a collection of PPT algorithms* $\Sigma_{\mathcal{R}} = (\mathsf{Commit}, \mathsf{Challenge}, \mathsf{Prove}, \mathsf{Verify})$ *such that*

$\mathsf{Commit}(x,w) \to (\mathsf{st}_{\mathcal{P}}, a)$**:** *on input a statement witness pair* $(x,w) \in \mathcal{R}$*, computes a commitment* a *and state* $\mathsf{st}_{\mathcal{P}}$*;*

$\mathsf{Challenge}(x,a) \to c$**:** *on input a statement* x *and commitment* a*, samples a challenge* $c \xleftarrow{\$} \mathcal{C}$ *and returns it;*

$\mathsf{Prove}(x,w,c,\mathsf{st}_{\mathcal{P}}) \to z$**:** *on input a challenge* c*,* $(x,w) \in \mathcal{R}$ *and prover state* $\mathsf{st}_{\mathcal{P}}$*, outputs a response* z*;*

$\mathsf{Verify}(x,a,c,z) \to b$**:** *on input a statement* x *and transcript* (a,c,z)*, consisting of a statement, commitment and response, computes and outputs* $b \in \{0,1\}$*, corresponding to* 1 *for acceptation and* 0 *otherwise.*

Definition 35 (Correctness). *A sigma protocol satisfies correctness if for all* $(x, w) \in \mathcal{R}$.

$$\Pr\left[\mathsf{Verify}(x,a,c,z)=1 \;\middle|\; \begin{array}{l}(\mathsf{st}_{\mathcal{P}},a) \xleftarrow{\$} \mathsf{Commit}(x,w)\\ c \xleftarrow{\$} \mathsf{Challenge}(x,a)\\ z \xleftarrow{\$} \mathsf{Prove}(x,w,c,\mathsf{st}_{\mathcal{P}})\end{array}\right] = 1$$

Definition 36 (Special soundness). *A sigma protocol as in Definition 34 satisfies special soundness if there exists a PPT extractor algorithm* $\mathcal{E}$ *that given two accepting transcripts* (a, c, z) *and* (a, c', z') *for a statement* x *such that* $c \neq c'$*, outputs a witness* w *for* x.

Definition 37 (Perfect Honest-Verifier Zero-Knowledge). *A sigma protocol satisfies perfect honest-verifier zero-knowledge if there exists a PPT simulator algorithm* Sim *that given a statement* x *outputs an accepting transcript* (a, c, z) *for that statement, such that the output of* Sim *is statistically indistinguishable from honestly generated transcripts.*

C Background on Unforgeability Proofs for Blind Signatures from [KLX22] and [KLLQ23]

In this section we provide the definitions and lemmas necessary to state the theorem from which we obtain the one-more unforgeability of GABS. The theory that we discuss was originally introduced in [KLX22] to prove the security of the Abe-Okamoto partially blind signature scheme and some definitions have been generalised in [KLLQ23] for compatibility with more general constructions. In Section C.1 we discuss some details required to even further generalise the proof strategy for usage in the $\mathrm{GAIP}_{\mathsf{IG}}$ setting, which may use countably infinite groups and sets.

Contrary to [KLX22] and [KLLQ23], our OR proofs may have multiple valid witness for each branch. In Section C.2 we discuss how the proof from [KLX22] may be adapted to fit this multiple-witness setting.

We first state the following definition 38 that describes the setting in which the proof strategy from [KLX22] is applicable. This definition adapts the preparation from [KLLQ23, Section 3.2], but contrastingly considers more than two possible secret keys for each valid public key, due to the multiple-witness setting. We formalise this in condition 2. in Definition 38. In section H we show that Definition 38 is satisfied for the GABS construction.

Definition 38 (Required Conditions for the OMUF Proof Strategy). *Let* BS *be a blind signature scheme that is to be specified. For* λ *in* $\mathbb{N}$*, corresponding to a group action* $* : [Q] \times \mathbb{G} \to [Q]$ *admitting an instance generator* IG*, the following hold:*

1. *the signatures that are produced under* BS *correspond to proofs for* $\Sigma^{\mathsf{OR}}_{Q,Q,\lambda}$ *(Definition 25) for the NP OR relation* $\mathcal{R}^{\mathsf{OR}}_{Q,Q,\lambda} \subset \mathcal{X}^{\mathsf{OR}} \times \mathcal{W}^{\mathsf{OR}}$ *(Definition 24).*

2. *Valid public keys for* BS *must contain an element* $(P_0, P_1) \in \mathcal{X}^{\mathsf{OR}}$ *and for each honestly generated public key* pk*, its set of corresponding secret keys is given by* $\mathcal{S}_{\mathsf{pk}} := \{(\delta, W_\delta) \in \{0,1\} \times \mathbb{G} \mid ((P_0, P_1), (\delta, W_\delta)) \in \mathcal{R}^{\mathsf{OR}}\}$*. Furthermore, denoting for* $b \in \{0,1\}$ *the sets* $\mathcal{S}_{\mathsf{pk},b} = \{(\delta, W_\delta) \in \mathcal{S}_{\mathsf{pk}} \mid \delta = b\}$*, these form a partition of* $\mathcal{S}_{\mathsf{pk}}$*. For each valid public key* pk *there exists a bijection* $\phi_{\mathsf{pk}} : S_{\mathsf{pk},0} \to \mathcal{S}_{\mathsf{pk},1}$*.*
3. *One may assume without lowering her advantage that an* OMUF *game adversary outputting* $k+1$ *message signatures pairs for* $k \in \mathbb{N}$ *has performed* $k+1$ *distinct hash queries in order to do so, and that each hash query corresponds to exactly one message signature pair (in the sense that the query to the hash oracle corresponds to the one performed when verifying it);*
4. *One may assume without lowering her advantage that an adversary's message* ρ_{Use} *sent to* BSign_2 *is in the challenge space of* $\Sigma^{\mathsf{OR}}_{Q,Q,\lambda}$*.*
5. *Consider instances as defined in Definition 39 hereafter. The set* $\mathcal{I}$ *of all possible instances must admit a distribution according to which an instance may be efficiently sampled, and under which* 0*-side instances and* 1*-side instances are equally likely to be sampled.*

The one-more unforgeability proof reduces the one-more unforgeability of a blind signature scheme satisfying Definition 38 to the $\mathrm{GAIP}_{\mathsf{IG}}$ problem. To this aim, a $\mathrm{GAIP}_{\mathsf{IG}}$ challenge is embedded in the public key pk, such that al secret keys in either $\mathcal{S}_{\mathsf{pk},0}$ or $\mathcal{S}_{\mathsf{pk},1}$ contain a solution to it. The remaining components of the public key and the other possible secret key can be initialised such that singing can still be simulated for the one-more unforgeability adversary. Using a forking strategy, one may hope to obtain a pair of signatures from which the secret key containing a solution to the $\mathrm{GAIP}_{\mathsf{IG}}$ challenge can be extracted. Theorem 49 states that if the one-more unforgeability adversary is sufficiently likely to succeed, then one is able to extract the secret key containing the challenge solution with non-negligible probability.

Definition 39 (Instances, from [KLLQ23], Definition 3.1). *Consider a blind signature scheme* BS *which satisfies the conditions* 1. *to* 4. *in Definition 38. Let* pk *be an honestly generated public key and recall that* $\mathcal{S}_{\mathsf{pk},0}$ *and* $\mathcal{S}_{\mathsf{pk},1}$ *together contain all secret keys that correspond to* pk*. For* b *in* $\{0,1\}$*, a* b*-side instance* I *consists of a secret key* $\mathsf{sk}_b \in \mathcal{S}_{\mathsf{pk},b}$ *and any randomness used by the signer algorithm* $\mathsf{Sign}(\mathsf{sk}_b)$*, excluding the randomness used for key generation.*

In the rest of this section we consider a fixed public key pk.

We introduce a second type of vector notation where vectors are denoted by an overhead arrow. This type of vector notation indicates that each of the vector components corresponds to one of the message signature pairs that the adversary outputs in the OMUF game. A component of such a vector is denoted with a subscript index or a superscript index in parenthesis if the component itself can already be indexed using subscript. Vector components are denoted without the overhead arrow. Let $\overrightarrow{h}$ denote the $(k+1)$-long vector of answers from a random oracle $\mathcal{O}^{\mathcal{H}}$.

$\mathcal{W}$ will be a wrapper algorithm interacting with an adversary $\mathcal{A}$ which uses fixed randomness rand, taking as other inputs an instance I and the vector $\overrightarrow{h}$. $\mathcal{W}(I, \mathsf{rand}, \overrightarrow{h})$ will output $\perp$ if the adversary aborts, and else will return $\mathcal{A}$'s output. Let ϵ denote $\mathcal{A}$'s success probability.

Definition 40 (Succesfull Tuples, from [KLX22]). *The set of successful tuples is defined as*

$$\mathsf{Succ} := \{(I, \mathsf{rand}, \overrightarrow{h}) \mid \mathcal{W}(I, \mathsf{rand}, \overrightarrow{h}) \neq \perp\}$$

Definition 41 (Succesful Forking, from [KLX22], Definition 7). *Two successful input tuples $(I, \mathsf{rand}, \overrightarrow{h}), (I, \mathsf{rand}, \overrightarrow{h'}) \in \mathsf{Succ}$, fork from each other at index $i \in [k+1]$ if $\forall j < i$, $h_j = h'_j$, and $h_i \neq h'_i$. i-th successful forking sets are defined as*

$$F_i(I, \mathsf{rand}) = \{(h_i, h'_i) \mid (I, \mathsf{rand}, \overrightarrow{h}) \text{ and } (I, \mathsf{rand}, \overrightarrow{h'}) \text{ fork at index } i.\}$$

Definition 42 (Query Transcript, from [KLX22], Definition 5). *Let $\mathcal{W}$ be the wrapper algorithm, running on input $(I, \mathsf{rand}, \overrightarrow{h})$. Let the query transcript, denoted by $\overrightarrow{e}(I, \mathsf{rand}, \overrightarrow{h})$, be the vector of messages from the adversary playing the role of a user to BSign during the simulation by $\mathcal{W}$.*

Definition 43 (Full Transcript, from [KLX22], Definition 6). *Let $\mathcal{W}$ be the wrapper algorithm, which runs on input $(I, \mathsf{rand}, \overrightarrow{h})$. Let $\mathsf{trans}(I, \mathsf{rand}, \overrightarrow{h})$ denote the full transcript, given by the transcript produced between the signer and the adversary.*

Definition 44 (Partners, from [KLX22], Definition 8). *Two successful tuples $(I, \mathsf{rand}, \overrightarrow{h}), (I, \mathsf{rand}, \overrightarrow{h'}) \in \mathsf{Succ}$ are partners at index $i \in [k+1]$ if:*

1. *$(I, \mathsf{rand}, \overrightarrow{h})$ and $(I, \mathsf{rand}, \overrightarrow{h'})$ fork at index i.*
2. *$\overrightarrow{e}(I, \mathsf{rand}, \overrightarrow{h}) = \overrightarrow{e}(I, \mathsf{rand}, \overrightarrow{h'})$*

We define the set

$$\mathsf{prt}_i(I, \mathsf{rand}) := \left\{ \left(\overrightarrow{h}, \overrightarrow{h'}\right) \;\middle|\; \begin{array}{l} (I, \mathsf{rand}, \overrightarrow{h}) \text{ and } (I, \mathsf{rand}, \overrightarrow{h'}) \\ \text{are partners at index } i \end{array} \right\}.$$

Definition 45 (Triangles, from [KLX22], Definition 9). *We denote a triangle at index $i \in [k+1]$ with respect to (I, rand) by*

$$\triangle_i(I, \mathsf{rand}) := \left\{ \begin{array}{l|l} ((I, \mathsf{rand}, \overrightarrow{h}), & (\overrightarrow{h}, \overrightarrow{h'}) \in \mathsf{prt}_i(I, \mathsf{rand}) \\ (I, \mathsf{rand}, \overrightarrow{h'}), & (\overrightarrow{h}, \overrightarrow{h''}) \in F_i(I, \mathsf{rand}) \\ (I, \mathsf{rand}, \overrightarrow{h''})) & (\overrightarrow{h'}, \overrightarrow{h''}) \in F_i(I, \mathsf{rand}) \end{array} \right\}.$$

For a triangle $((I, \mathsf{rand}, \overrightarrow{h}), (I, \mathsf{rand}, \overrightarrow{h'}), (I, \mathsf{rand}, \overrightarrow{h''})) \in \triangle_i(I, \mathsf{rand})$ we say that $((I, \mathsf{rand}, \overrightarrow{h}), (I, \mathsf{rand}, \overrightarrow{h'}))$ denotes the base and we call $((I, \mathsf{rand}, \overrightarrow{h}), (I, \mathsf{rand}, \overrightarrow{h''}))$ and $((I, \mathsf{rand}, \overrightarrow{h'}), (I, \mathsf{rand}, \overrightarrow{h''}))$ sides.

The following definition 46 adapts [KLLQ23, Definition 3.9], which considered specifically the case $|\mathcal{S}_{\mathsf{pk},0}| = |\mathcal{S}_{\mathsf{pk},1}| = 1$. Later in Section C.2 we justify our generalisation.

Definition 46 (Witness Extractors, adapted from [KLLQ23], Definition 3.9). *Let I, rand be fixed and consider a successful fork $(\overrightarrow{h}, \overrightarrow{h'}) \in F_i(I, \mathsf{rand})$ for some $i \in [k+1]$. Note that $(I, \mathsf{rand}, \overrightarrow{h})$ and $(I, \mathsf{rand}, \overrightarrow{h'})$ are successful tuples. Let σ_i and σ'_i denote the signatures that correspond to h_i and h'_i, respectively. Two deterministic algorithms Ext_0 and Ext_1 are witness extractors if for any such (σ_i, σ'_i)*

$$(\mathsf{Ext}_0(\sigma_i, \sigma'_i), \mathsf{Ext}_1(\sigma_i, \sigma'_i)) \in (\mathcal{S}_{\mathsf{pk},0} \times \{\bot\}) \cup (\{\bot\} \times \mathcal{S}_{\mathsf{pk},1}) \cup (\mathcal{S}_{\mathsf{pk},0} \times \mathcal{S}_{\mathsf{pk},1}).$$

For $b \in \{0,1\}$, we say that a b-side witness can be extracted from $(I, \mathsf{rand}, \overrightarrow{h})$ and $(I, \mathsf{rand}, \overrightarrow{h'})$ at index i if $\mathsf{Ext}_b(\sigma_i, \sigma'_i) \in \mathcal{S}_{\mathsf{pk},b}$.

Lemma 47 (From [KLX22], Lemma 2).

Let rand, $\overrightarrow{h}$ be fixed. For successful tuples $(I, \mathsf{rand}, \overrightarrow{h}) \in \mathsf{Succ}$ there exists a map $\Phi_{\mathsf{rand}, \overrightarrow{h}}(I)$ sending a b-side instance I to a $(1-b)$-side instance. Furthermore, $\Phi_{\mathsf{rand}, \overrightarrow{h}}$ is a self-inverse bijection and it respects transcripts, that is, for all $(I, \mathsf{rand}, \overrightarrow{h}) \in \mathsf{Succ}$,

$$\mathsf{trans}\left(I, \mathsf{rand}, \overrightarrow{h}\right) = \mathsf{trans}\left(\Phi_{\mathsf{rand}, \overrightarrow{h}}(I), \mathsf{rand}, \overrightarrow{h}\right).$$

Additionally, we define $\Phi(I, \mathsf{rand}, \overrightarrow{h}) := \left(\Phi_{\mathsf{rand}, \overrightarrow{h}}(I), \mathsf{rand}, \overrightarrow{h}\right)$ for successful tuples $(I, \mathsf{rand}, \overrightarrow{h}) \in \mathsf{Succ}$.

The following definition contains a minor adaptation from [KLX22], stating that 'a' b-side witness can be extracted rather than 'the' b-side witness. We require this adaptation to fit the multiple-witness setting. In Section C.2 we explain how the proof strategy from [KLX22] remains applicable.

Lemma 48 (Adapted from from [KLX22], Corollary 3). *Consider fixed values for rand and $\overrightarrow{h}$ and consider $(\overrightarrow{h}, \overrightarrow{h'}, \overrightarrow{h''}) \in \triangle_i(I, \mathsf{rand})$ for some $i \in [k+1]$. If a b-side witness can be extracted from the base $((I, \mathsf{rand}, \overrightarrow{h}), (I, \mathsf{rand}, \overrightarrow{h'}))$ at index i, then then one can also extract a b-side instance at index i from at least one of the sides $((I, \mathsf{rand}, \overrightarrow{h}), (I, \mathsf{rand}, \overrightarrow{h''}))$ or $((I, \mathsf{rand}, \overrightarrow{h'}), (I, \mathsf{rand}, \overrightarrow{h''}))$.*

Theorem 49 (Obtaining k-One-More Unforgeability, adapted from Theorem 3.12 from [KLLQ23]). *Let BS be a blind signature scheme satisfying Definition 38. In particular, let $\mathrm{GAIP}_{\mathsf{IG}}$ be the problem from condition 1. of Definition 38. Let $\mathcal{C}$ be the challenge space of the sigma protocol for the NP OR relation underlying BS. If Lemma 47 and Lemma 48 hold, then for all k, λ in*

$\mathbb{N}$, *if there exists a PPT adversary* $\mathcal{A}$ *that makes* $Q_{\mathcal{H}}$ *hash queries to a random oracle, if* $\mathsf{Adv}^{k-\mathsf{OMUF}}_{\mathcal{A}}(\lambda) \geq \frac{C_1}{|\mathcal{C}|} \cdot \binom{Q_{\mathcal{H}}}{k+1}$ *then*

$$\mathsf{Adv}^{k-\mathsf{OMUF}}_{\mathcal{A}}(\lambda) \leq \sqrt{\binom{Q_{\mathcal{H}}}{k+1}^2 \cdot \frac{(k+1)^3}{C_2} \cdot \mathsf{Adv}^{\mathrm{GAIP}_{\mathsf{IG}}}_{\mathcal{A}}(\lambda)}$$

for some universal constants $C_1, C_2 > 0$.

C.1 The OMUF Proof Strategy with Infinite Sets

The proof in [KLX22] and its adaptation in [KLLQ23] are both written with finite sets and uniform sampling in mind. The setting that we describe in Section 2.1, especially Definition 6, considers infinite sets from which we can sample according to non-uniform distributions. It is still possible to employ the one-more unforgeability proof strategy from [KLX22] for of GABS, but with adequate precautions. In this section we discuss a change in notation and some minor additions required to reuse the proof.

Let $\mathcal{I}$ denote the set of all instances as in Definition 39, and $\mathcal{R}$ the set of all possible random tapes for a fixed adversary $\mathcal{A}$ that makes $k+1$ many hash queries to the random oracle. The set of all possible combined random oracle responses equals $(\{0,1\}^\lambda)^{k+1}$. In the setting of [KLX22] and [KLLQ23] the tuples $(I, \mathsf{rand}, \overrightarrow{h})$ are sampled uniformly at random from $\mathcal{I} \times \mathcal{R} \times (\{0,1\}^\lambda)^{k+1}$. Therefore, given a subset $\mathcal{G} \subset \mathcal{I} \times \mathcal{R} \times (\{0,1\}^\lambda)^{k+1}$, the probability that a randomly sampled tuple $(I, \mathsf{rand}, \overrightarrow{h})$ lies in $\mathcal{G}$ is equal to the ratio of both cardinalities. This is used as a choice of notation in [KLX22].

Instances I for the GABS setting contain elements in a group $\mathbb{G}$ sampled using an instance generator (Definition 6). In particular $\mathbb{G}$ can be an infinite group and the sampling output of the instance generator does not follow the uniform distribution, thus the above notation using cardinalities cannot be used in the context of GABS. Still, note that for a fixed value Q instances I in the context of GABS can be sampled from $\mathcal{I}$ according to some fixed but non-uniform distribution that corresponds to an honest execution of the signer protocol. The probability for an instance to lie in a given subset of $\mathcal{I}$ is then determined by this distribution. For the unforgeability proof we require that the distribution fulfills condition 5. from Definition 38.

Let $\mathcal{D}_{\mathcal{I}}$ denote the distribution on $\mathcal{I}$, according to which instances are sampled. In the rest of this section, let $X = (I, \mathsf{rand}, \overrightarrow{h})$ be random variable with $I \xleftarrow{\$} \mathcal{D}_{\mathcal{I}}$, and $\mathsf{rand} \xleftarrow{\$} \mathcal{R}$ as well as $\overrightarrow{h} \xleftarrow{\$} \{0,1\}^{\lambda(k+1)}$ chosen uniformly at random.

Let B^b_T for $b \in \{0,1\}$ be defined analogous to [KLX22, Definition 14]. We adapt the following definition from [KLX22]. In [KLX22], $B^\times_T$ is defined as the larger set of two finite B^y_T and B^z_T, considering $b \in \{y, z\}$. In our setting, the corresponding sets B^0_T and B^1_T might be infinite. Therefore, we define $\times := \arg\max_{b \in \{0,1\}} \{\Pr_X[X \in B^b_T]\}$.

Remaining definitions and statements that would rely on cardinalities of infinite sets for probabilities, can be reformulated using standard probability notation.

The proofs in [KLX22] sometimes considered a situation where only the vector of random oracle responses was variable, for a fixed instance and random tape on which to run the adversary. In this case the notation using cardinalities of sets need not be adapted as the set of possible hash query responses remains finite and admits uniform sampling in the context of GABS.

In particular we note that the sets $T^{\times}_{T,i}(I, \mathsf{rand}, \overrightarrow{h})$ and $A^{\times}_{T,i}(I, \mathsf{rand}, \overrightarrow{h})$ from [KLX22, Definitions 15, 17] remain finite an can be used with cardinality notation for probabilities in our setting. This is required for the statements and proofs of [KLX22, Lemma 8, Corollary 4, Lemma 9, Lemma 10].

The proof of [KLX22] uses the classical splitting lemma [PS00, Lemma 7] and the bucket lemma [KLX22, Lemma 13]. The splitting lemma as formulated in [PS00] can be directly used in the context of GABS and the formulation in [KLX22, Lemma 12] can also be used by replacing uniform sampling by more general distributions. The same goes for bucket lemma formulated in [KLX22], as explained hereafter.

Lemma 50 (Bucket Lemma, generalised). *Let $\mathcal{X}$ be a (possibly infinite) set, $\mathcal{D}$ a distribution on $\mathcal{X}$, b be a natural integer, and $\mathcal{B}_1, ..., \mathcal{B}_b \subset \mathcal{X}$ such that $\bigcup_{i=1}^{b} \mathcal{B}_i = \mathcal{X}$. Then for all $\alpha \in (0,1)$ there exists a set $\mathcal{G}_\alpha \subset \mathcal{X}$ such that*

1. $\Pr_{x \xleftarrow{\$} \mathcal{D}}[x \in \mathcal{G}_\alpha] > (1-\alpha)$.
2. *For all $x \in \mathcal{G}_\alpha$, there exists $i \in [b]$ such that $x \in B_i$ and $\Pr_{y \xleftarrow{\$} \mathcal{D}}[y \in B_i] \geq \frac{\alpha}{b}$.*

Proof. Let α be in $(0,1)$, $\mathcal{B} := \{\mathcal{B}_1, ..., \mathcal{B}_b\}$, $\widetilde{\mathcal{B}}_\alpha := \left\{ B \in \mathcal{B} \mid \Pr_{y \xleftarrow{\$} \mathcal{D}}[y \in B] < \frac{\alpha}{b} \right\}$, and $\mathcal{F}_\alpha := (\mathcal{X} \setminus \cup_i \mathcal{B}_i) \cup (\bigcup \widetilde{\mathcal{B}}_\alpha)$. One has

$$\begin{aligned} \Pr_{x \xleftarrow{\$} \mathcal{D}}[x \in \mathcal{F}_\alpha] \leq \Pr_{x \xleftarrow{\$} \mathcal{D}}\left[x \in \bigcup \widetilde{\mathcal{B}}_\alpha\right] &= \sum_{B \in \widetilde{\mathcal{B}}_\alpha} \Pr_{x \xleftarrow{\$} \mathcal{D}}[x \in B] \\ &< \sum_{B \in \widetilde{\mathcal{B}}_\alpha} \frac{\alpha}{b} \leq b \cdot \frac{\alpha}{b} = \alpha. \end{aligned}$$

Hence for $\mathcal{G}_\alpha := \mathcal{X} \setminus \mathcal{F}_\alpha$, $\Pr_{x \xleftarrow{\$} \mathcal{D}}[x \in \mathcal{G}_\alpha] \geq \alpha$, and statement 1 is satisfied. Moreover, statement 2 directly follows from the definition of $\mathcal{F}_\alpha$. □

The proof in [KLX22] contains a single lemma [KLX22, Lemma 6] of which the proof relies on the uniform distribution on the finite set $\mathcal{X} := \mathcal{I} \times \mathcal{R} \times (\{0,1\}^\lambda)^{k+1}$. The argument they make is that if a subset of $\mathcal{B} \subset \mathcal{X}$, containing only successful tuples, can be partitioned into 2 disjoint subsets $\mathcal{B}_0 \cup \mathcal{B}_1 = \mathcal{B}$ such that there exists a bijection between $\mathcal{B}_0$ and $\mathcal{B}_1$, then

$$\Pr_{x \xleftarrow{\$} \mathcal{X}}[x \in \mathcal{B}_0] = \Pr_{x \xleftarrow{\$} \mathcal{X}}[x \in \mathcal{B}_1] = \frac{1}{2} \cdot \Pr_{x \xleftarrow{\$} \mathcal{X}}[x \in \mathcal{B}]. \tag{1}$$

If $\mathcal{X}$ is finite and sampling is done uniformly at random then this is indeed true as $|\mathcal{B}_0|=|\mathcal{B}_1|$ and $|\mathcal{B}_0|+|\mathcal{B}_1|=|\mathcal{B}|$. In our case $\mathcal{X}$ does not admit a uniform distribution, but Equation 1 is still required for more general distributions $\mathcal{D}$. For the proof in [KLX22, Lemma 6] the bijection between $\mathcal{B}_0$ and $\mathcal{B}_1$ is provided by Φ fulfilling Lemma 47. We provide a general Lemma 51 that states we still obtain an analog of Equation 1, replacing the uniform distribution by $\mathcal{D}$, under specific conditions on $\mathcal{D}$ and Φ.

Lemma 51. *Let $\mathcal{X}$ be a (possibly infinite) set admitting a subset $\mathcal{B}$, $\mathcal{B}_0 \cup \mathcal{B}_1 = \mathcal{B}$ be a partitioning of $\mathcal{B}$, $\mathcal{D}$ a distribution on $\mathcal{X}$ and Φ be a function from $\mathcal{X}$ to $\mathcal{X}$ that is a bijection between $\mathcal{B}_0$ and $\mathcal{B}_1$, such that for all $y \in \mathcal{X}$, $\Pr_{x \xleftarrow{\$} \mathcal{D}}[x = y] = \Pr_{x \xleftarrow{\$} \mathcal{D}}[x = \Phi(y)]$. Then,*

$$\Pr_{x \xleftarrow{\$} \mathcal{D}}[x \in \mathcal{B}_0] = \Pr_{x \xleftarrow{\$} \mathcal{D}}[x \in \mathcal{B}_1] = \frac{1}{2} \cdot \Pr_{x \xleftarrow{\$} \mathcal{D}}[x \in \mathcal{B}].$$

Proof. As Φ is a bijection between $\mathcal{B}_0$ and $\mathcal{B}_1$, $\mathcal{B}_0 = \Phi(\mathcal{B}_1)$. Moreover, as for any x in $\mathcal{X}$, $\Pr_{x \xleftarrow{\$} \mathcal{D}}[x = y] = \Pr_{x \xleftarrow{\$} \mathcal{D}}[x = \Phi(y)]$,

$$\Pr_{x \xleftarrow{\$} \mathcal{D}}[x \in \mathcal{B}_0] = \Pr_{x \xleftarrow{\$} \mathcal{D}}[x \in \Phi(\mathcal{B}_1)] = \Pr_{x \xleftarrow{\$} \mathcal{D}}[x \in \mathcal{B}_1] \tag{2}$$

Furthermore, $\Pr_{x \xleftarrow{\$} \mathcal{D}}[x \in \mathcal{B}_0] + \Pr_{x \xleftarrow{\$} \mathcal{D}}[x \in \mathcal{B}_1] = \Pr_{x \xleftarrow{\$} \mathcal{D}}[x \in \mathcal{B}]$ as $\mathcal{B}_0$ and $\mathcal{B}_1$ form a partitioning of $\mathcal{B}$, which combined with Equation (2) yields the desired result.

□

In [KLX22, Lemma 6] the sets $\mathcal{B}_0$ and $\mathcal{B}_1$ are set to equal $B^{\times}_{T,0}$ and $B^{\times}_{T,0}$, respectively. In Section H.3 we make $\mathcal{D}$ and Φ explicit and provide Lemma 74, which states that also $\Pr_{y \xleftarrow{\$} \mathcal{D}}[y = x] = \Pr_{y \xleftarrow{\$} \mathcal{D}}[y = \Phi(x)]$. The following lemma 52 is a special case of Lemma 51 and may be used to directly replace [KLX22, Lemma 6].

Recall that $B^{\times}_{T,0}$ and $B^{\times}_{T,1}$ are defined according to [KLX22, Defintion 14], and that $X = (I, \mathsf{rand}, \overrightarrow{h})$ is a random variable.

Lemma 52 (Adapted from Lemma 6 [KLX22]).

$$\Pr_X[X \in B^{\times}_{T,0}] = \Pr_X[X \in B^{\times}_{T,1}] = \frac{1}{2} \Pr_X[X \in B^{\times}_{T}]$$

Proof. This is a special case of Lemma 51 with $\mathcal{B}_0 = B^{\times}_{T,0}$, $\mathcal{B}_1 = B^{\times}_{T,1}$ and $\mathcal{B} = B^{\times}_{T}$. The distribution $\mathcal{D}$ on the set of triples $(I, \mathsf{rand}, \overrightarrow{h})$, and the bijection Φ are made explicit in H.3.

□

In the statements of [KLX22, Lemmas 10, 11], a random triple $(I, \mathsf{rand}, \overrightarrow{h})$ is sampled by first sampling a uniform random bit $b \xleftarrow{\$} \{0, 1\}$ and then sampling a

uniform random b-side instance I. The values rand and $\overrightarrow{h}$ are sampled uniformly at random. On our setting, it is not possible to sample instances uniformly at random.

On closer inspection, none of the events described in the statements of [KLX22, Lemmas 10, 11] depend on the bit b. Hence, in order to use [KLX22, Lemmas 10, 11] in the proof of Theorem 49 it is important that 0-side and 1-side instances are sampled with equal likelihood. For the instance I in our random variable $X = (I, \mathsf{rand}, \overrightarrow{h})$, this is ensured by condition 5. of Definition 38. Therefore, we can simply replace their random variable $(I, \mathsf{rand}, \overrightarrow{h})$ by our random variable X and omit seperately sampling the bit b.

The proofs of [KLX22, Lemmas 10, 11] are not affected by this change.

C.2 The OMUF Proof Strategy in The Multiple Witness Setting

The hardness assumptions used by [KLX22] and [KLLQ23] both considered problems for which a problem statement always has one unique solution. As a result, their OR statements have exactly two valid witness, one for each branch. Using the formulation of this article, they consider the case $|\mathcal{S}_{\mathsf{pk},0}| = |\mathcal{S}_{\mathsf{pk},1}| = 1$ (Definition 38). Because of this, they could refer to the element in $\mathcal{S}_{\mathsf{pk},b}$ as 'the' b-side witness, while in this article it would be 'a' b-side witness, as $\mathcal{S}_{\mathsf{pk},b}$ may contain multiple elements. In this section we discuss how the proof from [KLX22] may be adapted to fit this change.

The original proof of [KLX22] embeds to problems instances in the OR-statement forming the public key. One problem has a known solution used for simulation of the signer, while the other has an unknown solution that the challenger aims to obtain. The proof of [KLX22] shows that with sufficient probability, the unknown solution may be extracted from a successful one-more unforgeability adversary. In the setting of this article, multiple such solutions may exist.

One key ingredient of the proof is, provided fixed user randomness rand and hash responses $\overrightarrow{h}$, the existence of a self-inverse bijection $\Phi_{\mathsf{rand},\overrightarrow{h}}$ on instances I, that maps b-side instances to $(1-b)$-side instances and preserves transcripts. Furthermore, the proof relies witness extractability from the underlying OR proofs. In the setting of this article, there may be multiple valid witnesses, but for extractability it suffices if at least one of them may be obtained.

We first discuss how we define the bijection Φ in the setting of this article, and then how the witness extractors may be obtained for GABS. Afterwards, we discuss some slight formulation changes in definitions and statements from [KLX22] due to the multiple-witness setting. Lastly, we discuss how the proof of [KLX22, Theorem 1] may be used to prove Theorem 49.

A b-side instance according to Definition 39 does not uniquely define a secret key in $\mathcal{S}_{\mathsf{pk},1-b}$. In order to define the bijection Φ in the multiple-witness setting, we require that that for each public key pk, there exists a bijection $\phi_{\mathsf{pk}} : \mathcal{S}_{\mathsf{pk},0} \to \mathcal{S}_{\mathsf{pk},1}$ (Definition 38 condition 2.). We use this in Section H.3 for the explicit definition for $\Phi_{\mathsf{rand},\overrightarrow{h}}$. Note that all bijections ϕ_{pk} are fixed beforehand. The

bijections ϕ_{pk} are used in the proof of Lemma 79 which is analogous to [KLX22, Lemma 2]. Subsequent results [KLX22, Corollary 1, Lemma 3, Corollary 2] can be obtained via the original proofs from [KLX22].

Next, we consider [KLX22, Definition 13, Lemma 4] that describes witness extractors and conditions under which they aught to be successful. In [KLX22], these extractors are required to extract *the* witness from one of the branches, while in our setting this can be one of multiple witnesses for each branch. We provide a replacement definition 46 for witness extractors that rely on the special soundness of the underlying OR proofs. Lemma 82 states sufficient conditions under which the extractors are successful and serves to replace [KLX22, Lemma 4].

We restate [KLX22, Corollary 3] as Lemma 48 and in Lemma 83 we show that it holds for the witness extractors in Figure 7.

Some definitions and statements in [KLX22] directly mention 'the' b-side witness. In order to adapt them to the multiple-witness setting it suffices to replace all such mentions to 'a' b-side witness. This affects the following definitions [KLX22, Definitions 14, 15, 16, 17, 19] for the sets D_i^b, $T_{T,i}^{\times}$, $O_T^{\times}$, $A_{T,i}^{\times}$ and $\widehat{O_T^{\times}}$, and [KLX22, Corollary 3, Lemma 9].

Lastly, a few proofs for intermediate results in [KLX22] mention extraction of 'the' b-side witness for $b \in \{y, z\}$, but these arguments only require extraction of 'a' witness. All such proofs can be adapted to the multiple-witness setting by replacing mentions of 'the' b-side witness by 'a' b-side witness for $b \in \{0, 1\}$, once again relying on the existence of an extractor satisfying Definition 46.

The proof of Theorem 49 Theorem 49 is the analogue to [KLX22, Theorem 1] with $C_1 = 423\left(1 - \frac{1}{(k+1)^2}\right)$, $|\mathcal{C}| = 2^{\lambda}$ and $C_2 = \frac{3}{75423744}$. The statement reduces the the unforgeability of the relevant signature scheme BS to the $\mathrm{GAIP}_{\mathsf{IG}}$.

The proof of [KLX22] first samples a uniform random bit $b \xleftarrow{\$} \{0, 1\}$, after which a b-side instance I is sampled. In our setting, one would sample a triple $X = (I, \mathsf{rand}, \overrightarrow{h'})$ directly and set $b \in \{0, 1\}$ to correspond to I being a b-side instance. This is justified by requiring condition 5. from Definition 38.

As a result, we consider our probabilities over the random variables X, $i \xleftarrow{\$} [k+1]$ and $\overrightarrow{h'} \xleftarrow{\$} \{-1, 1\}^{\lambda(k+1)}_{|\overrightarrow{h}_{[i-1]}}$. The set $\{-1, 1\}^{\lambda(k+1)}_{|\overrightarrow{h}_{[i-1]}}$ denotes the subset of $\{-1, 1\}^{\lambda(k+1)}$ in which the fist $\lambda(i-1)$ components of each element equal those of $\overrightarrow{h}$.

Furthermore, our setting considers renamed variables 0 instead of $\mathbf{y}$ and 1 instead of $\mathbf{z}$. In the case of two successful tuples $(I, \mathsf{rand}, \overrightarrow{h})$ and $(I, \mathsf{rand}, \overrightarrow{h'})$ with $(\overrightarrow{h}, \overrightarrow{h'}) \in F_i(I, \mathsf{rand})$ the statement '$\delta_i \neq \delta_i'$' from [KLX22] means in our setting that the 1-side witness extractor is successful at index i, and $\omega_i \neq \omega_i'$ means that the 0-side witness extractor is successful at index i.

The value q from [KLX22] corresponds to $2^{\lambda} = |\{-1, 1\}|^{\lambda}$, being the size of the underlying challenge space of our OR proofs.

In the setting of [KLX22], the $1-b$ component of the public key contains the $\mathrm{GAIP}_{\mathsf{IG}}$ challenge. In the proof of [KLX22], by [KLX22, Lemma 4], if the corresponding witness extractor is successful then it extracts the unique solution to this challenge. In our case, the challenge is embedded in the $1-b$ component of the public key, but the witness extractor may extract one of multiple possible solutions. Our adapted Definition 46 and Lemma 82, serving to replace [KLX22, Lemma 4], take this into account by allowing for multiple valid outputs per branch. Intermediate results from [KLX22] that are applied in the proof of [KLX22, Theorem 1], can be directly adapted, taking into account the multiple-witness setting.

In conclusion, the proof of [KLX22, Theorem 1] may be used to prove Theorem 49.

D Background on Key Encapsulation Mechanisms

Our construction of the GASDVS scheme for strong designated verifier signatures with non-delegatability requires a Key Encapsulation Mechanism (KEM), which we define and explain in this section.

Definition 53 (Key-Encapsulation Mechanism). *A Key-Encapsulation Mechanism (KEM) is a tuple of PPT algorithms* $(\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$*. On input a security parameter* λ *in* $\mathbb{N}$*, the generation algorithm* Gen *outputs a keypair* $(\mathsf{pk}, \mathsf{sk})$*. On input* pk*, the encapsulation algorithm* Encaps *outputs a shared key* K *in* $\{0,1\}^\lambda$ *together with an encapsulation* E*. On input* (sk, E)*, the decapsulation algorithm* Decaps *outputs* K *in* $\{0,1\}^\lambda$*.*

Definition 54 (Correctness of Key-Encapsulation Mechanisms). *A key-encapsulation mechanism* $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ *satisfies correctness if there exists a negligible function* $\mathsf{negl} : \mathbb{N} \to \mathbb{R}_+$ *such that for any* λ *in* $\mathbb{N}$

$$\Pr\left[K \leftarrow \mathsf{Decaps}(\mathsf{sk}, E) \;\middle|\; \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda) \\ (K, E) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk})\end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Definition 55 (Session-Key CCA Privacy of Key-Encapsulation Mechanisms [BdPP23]). *A KEM* $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ *satisfies session key indistinguishability against chosen ciphertext attacks* $(\mathsf{SK\text{-}CCA})$ *if for any PPT algorithm* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl} : \mathbb{N} \to \mathbb{R}_+$ *such that for any* λ *in* $\mathbb{N}$

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{SK\text{-}CCA}}_{\mathcal{A}}(\lambda) - \frac{1}{2}\right]\right| \leq \mathsf{negl}(\lambda).$$

for $\mathsf{Exp}^{\mathsf{SK\text{-}CCA}}_{\mathcal{A}}$ *the experiment described in Figure 8.*

An $\mathsf{SK\text{-}CCA}$-secure KEM relying on the ac-ΔLIP may be obtained by applying the FO transform [FO13] to the KEM provided in [DvW22, Section 5].

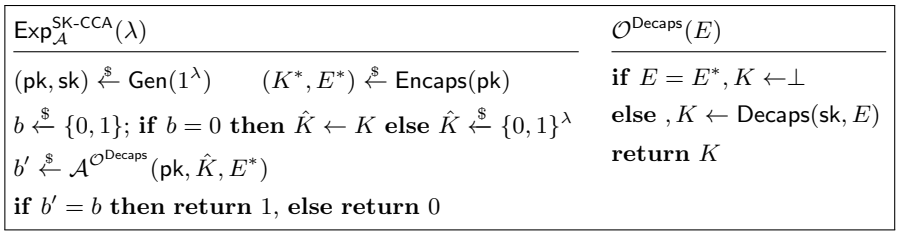


Figure 8. The session-key CCA privacy (SK-CCA) game for key-encapsulation mechanisms.

E Background on Secret-Key Encryption

Our construction of the GASDVS scheme for strong designated verifier signatures with non-delegatability requires Secret-Key Encryption (SKE), which we define and explain in this section.

Definition 56 (Secret-Key Encryption Scheme). *A secret-key encryption scheme is a tuple of PPT algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*. Let* $\lambda \in \mathbb{N}$ *be the security parameter. On input* 1^λ*, the generation algorithm* KeyGen *outputs a uniform random key* $\mathsf{sk} \xleftarrow{\$} \{0,1\}^\lambda$*. On input* $\mathsf{sk} \in \{0,1\}^\lambda$ *and a message* $m \in \{0,1\}^*$*,* $\mathsf{Enc}(\mathsf{sk}, m)$ *outputs a ciphertext* $c \in \{0,1\}^*$*. On input* $\mathsf{sk} \in \{0,1\}^\lambda$ *and a ciphertext* $c \in \{0,1\}^*$*,* $\mathsf{Dec}(\mathsf{sk}, c)$ *outputs a message* $m \in \{0,1\}^*$*.*

Definition 57 (Correctness of SKE). *A secret-key encryption scheme* $\mathsf{SKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *satisfies perfect correctness if for any* $\lambda \in \mathbb{N}$*, any* $\mathsf{sk} \in \{0,1\}^\lambda$ *and any message* $m \in \{0,1\}^*$

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{sk}, m)) = m.$$

Definition 58 (Eavesdropper Indistinguishability). *A secret-key encryption scheme* SKE *satisfies Eavesdropper Indistinguishability (*IND-EAV*) if for any PPT algorithm* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl} : \mathbb{N} \to \mathbb{R}_+$ *such that for any* $\lambda \in \mathbb{N}$

$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}EAV}}_{\mathcal{A}}(\lambda) - \frac{1}{2} \right] \right| \leq \mathsf{negl}(\lambda).$$

for $\mathsf{Exp}^{\mathsf{IND\text{-}EAV}}_{\mathcal{A}}$ *the experiment described in Figure 9.*

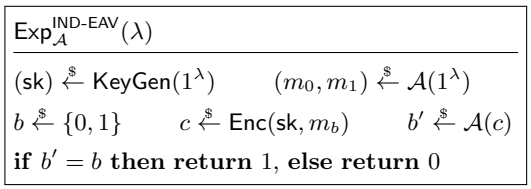


Figure 9. The IND-EAV security game for secret-key encryption.

F The Forking Lemma

The proofs for strong unforgeability and non-delegatability of the GASDVS scheme require the following forking lemma.

Definition 59 (Forking Lemma, adapted from [BN06], Lemma 1). *Let q be in $\mathbb{N}^*$, H be a set of at least two elements, and $\mathcal{W}$ be a PPT algorithm that takes as input $(x, h_1, ..., h_q)$, and runs on random coins ρ, outputting j in $[q]$ and some auxiliary value* out. *Let* IG *be a generator that on input a security parameter λ generates an input x for $\mathcal{W}$, $\mathsf{Fork}^{\mathcal{W}}$ be the forking algorithms for $\mathcal{W}$ as defined in Figure 10,*

$$\mathsf{acc}(\lambda) := \Pr\left[j \geq 1 \;\middle|\; x \xleftarrow{\$} \mathsf{IG}(1^\lambda), h_1, ..., h_q \xleftarrow{\$} H, (j, \mathsf{out}) \xleftarrow{\$} \mathcal{W}(x, h_1, ..., h_q)\right]$$

$$\textit{and} \qquad \mathsf{frk}(\lambda) := \Pr\left[b = 1 \;\middle|\; x \xleftarrow{\$} \mathsf{IG}(1^\lambda), (b, \mathsf{out}, \tilde{\mathsf{out}}) \xleftarrow{\$} \mathsf{Fork}^{\mathcal{W}}(x)\right].$$

Then, $\mathsf{frk}(\lambda) \geq \mathsf{acc}(\lambda) \cdot \left(\frac{\mathsf{acc}(\lambda)}{q} - \frac{1}{|H|}\right)$.

$\mathsf{Fork}^{\mathcal{W}}(x)$

Picks random coins ρ for $\mathcal{W}$.

Samples $h_1, ..., h_q \xleftarrow{\$} H$ and sets $(j, \mathsf{out}) \leftarrow \mathcal{W}(x, h_1, ..., h_q; \rho)$.

If $j = 0$, returns $(0, \perp, \perp)$.

Samples $h'_1, ..., h'_q \xleftarrow{\$} H$ and sets $(j', \mathsf{out}') \leftarrow \mathcal{W}(x, h_1, ..., h_{j-1}, h'_j, ..., h'_q; \rho)$.

If $j = j'$ and $h_j \neq h'_j$, returns $(1, \mathsf{out}, \mathsf{out}')$, else, $(0, \perp, \perp)$.

Figure 10. The forking algorithm for $\mathcal{W}$.

G Zero-Knowledge Proofs for GAIP Relations

G.1 Zero-Kowledge Proof for GAIP Relations

Definition 60 (Sigma Protocol for $\mathrm{GAIP}_{\mathsf{IG}}$ Relations). *We define a three-way sigma protocol to prove, given a statement x, knowledge of a witness w such that (x, w) is in $\mathcal{R}_{Q,\lambda}$ satisfying Definition 23, as the tuple of algorithms $\Sigma_{Q,\lambda} = (\mathsf{Commit}, \mathsf{Challenge}, \mathsf{Prove}, \mathsf{Verify})$ described hereafter.*

$\Sigma_{Q,\lambda}.\mathsf{Commit}(x, w) \to (\mathsf{st}_{\mathsf{Prove}}, \rho_{\mathsf{Prove}_1})$**:** samples $(\boldsymbol{R}, \boldsymbol{U}) \xleftarrow{\$} \mathsf{IG}(Q)^\lambda$ with λ draws of $\mathsf{IG}(Q)$, and outputs $(\mathsf{st}_{\mathsf{Prove}_1} := \boldsymbol{U}, \rho_{\mathsf{Prove}_1} := \boldsymbol{R})$.

$\Sigma_{Q,\lambda}.\mathsf{Challenge}(x, \rho_{\mathsf{Prove}_1}) \to \rho_{\mathsf{Verify}}$**:** on input $(x, \rho_{\mathsf{Prove}_1})$, samples $\boldsymbol{c} \xleftarrow{\$} \{0,1\}^\lambda$ and outputs $\rho_{\mathsf{Verify}} := \boldsymbol{c}$.

$\Sigma_{Q,\lambda}.\mathsf{Prove}(x, w, \mathsf{st}_{\mathsf{Prove}}, \rho_{\mathsf{Verify}}) \to \rho_{\mathsf{Prove}_2}$**:** on input $(x = Q', w = A, \mathsf{st}_{\mathsf{Prove}_1} = \boldsymbol{U}, \rho_{\mathsf{Verify}} = \boldsymbol{c})$, the algorithm sets $V_i = A^{-c_i} \cdot U_i$ for each $i \in [\lambda]$, sets $\boldsymbol{V} := (V_i)_{i\in[\lambda]}$, and outputs $\rho_{\mathsf{Prove}_2} := \boldsymbol{V}$.

$\Sigma_{Q,\lambda}.\mathsf{Verify}(x, \rho_{\mathsf{Verify}}, \rho_{\mathsf{Prove}_2}) \to b$**:** on input $(x = Q', \mathsf{st}_{\mathsf{Verify}} = \boldsymbol{c}, \rho_{\mathsf{Prove}_2} = \boldsymbol{V})$, returns $b \leftarrow 1$ if for each $i \in [\lambda]$, V_i is in $\mathbb{G}$, $R = Q * V_i$ if $c_i = 0$, and $R = Q' * V_i$ if $c_i = 1$, and outputs $b \leftarrow 0$ otherwise.

Theorem 61. *The $\Sigma_{Q,\lambda}$ protocol satisfies correctness, special soundness with error $2^{-\lambda}$, and perfect honest verifier zero-knowledge.*

Proof. The proofs for these properties are analogous to those in [DvW22, Section 4.1], but generalised to the $\mathrm{GAIP}_{\mathsf{IG}}$ setting, and detailed in the rest of this subsection.

Definition 62. *The sigma protocol in Definition 60 can be made non-interactive using the Fiat-Shamir transform [FS87]. Let $\Pi_{Q,\lambda} = (\mathsf{ComPr}, \mathsf{Verify})$ denote the commit and prove scheme for which on input a statement Q' and witness A for the $\mathrm{GAIP}^{Q}_{\mathsf{IG}}$ relation, ComPr generates a non-interactive proof of knowledge, which may be checked with Verify.*

In this section we discuss proofs for the properties of $\Sigma^{\mathrm{GAIP}}_{Q,\lambda}$ (Theorem 61).

Theorem 63. *The $\Sigma^{\mathrm{GAIP}}_{Q,\lambda}$ protocol satisfies correctness.*

Proof. Let $x := Q' \in [Q]$ be a statement and let $w := A \in \mathbb{G}$ be a witness for x, *i.e.* $Q' = Q * A$. We show that there exists a negligible function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}_+$ such that for $\lambda \in \mathbb{N}$,

$$\Pr\left[\mathsf{Verify}(x, \boldsymbol{c}, \boldsymbol{V}) = 1 \;\middle|\; \begin{array}{l} (\boldsymbol{R}, \boldsymbol{U}) \xleftarrow{\$} \mathsf{Commit}(x, w) \\ \boldsymbol{c} \xleftarrow{\$} \mathsf{Challenge}(x, \boldsymbol{R}) \\ \boldsymbol{V} \leftarrow \mathsf{Prove}(x, w, \boldsymbol{U}, \boldsymbol{c}) \end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

We consider the values $(\boldsymbol{U}, \boldsymbol{R}) \xleftarrow{\$} \mathsf{Commit}(x, w)$, $\boldsymbol{c} \xleftarrow{\$} \mathsf{Challenge}(x, \boldsymbol{R})$ and $\boldsymbol{V} \leftarrow \mathsf{Prove}(x, w, \boldsymbol{U}, \boldsymbol{c})$. Let $\epsilon^{\mathsf{Fail}}_{\mathsf{IG}}$ denote the probability that IG outputs $\perp$. Note that $\epsilon^{\mathsf{Fail}}_{\mathsf{IG}}$ is negligible in λ.

For all $i \in [\lambda]$, $R_i = Q * U_i$ as $(R_i, U_i) \xleftarrow{\$} \mathsf{IG}(Q)$. Furthermore, $\boldsymbol{c}$ is in $\{0,1\}^\lambda$ and $\boldsymbol{V}$ is computed as $V_i = A^{-c_i} \cdot U_i$. $U_i \in \mathbb{G}$ and $A \in \mathbb{G}$, so it follows that $V_i \in \mathbb{G}$. Recall that $Q' = Q * A$. We also see that

$$\begin{aligned} R = Q * U_i &= \begin{cases} Q * U_i & \text{if } c_i = 0 \\ Q * (A \cdot A^{-c_i} \cdot U_i) & \text{if } c_i = 1 \end{cases} \\ &= \begin{cases} Q * V_i & \text{if } c_i = 0 \\ Q' * V_i & \text{if } c_i = 1 \end{cases}. \end{aligned}$$

It follows that $\mathsf{Verify}(x, \boldsymbol{c}, \boldsymbol{V}) = 1$ if all calls to IG are successful.

Execution of $\Sigma_{Q,\lambda}^{\mathrm{GAIP}}$ requires λ calls to IG. It follows that

$$\Pr\left[\mathsf{Verify}(x,\boldsymbol{c},\boldsymbol{V})=1 \;\middle|\; \begin{array}{l}(\boldsymbol{R},\boldsymbol{U}) \xleftarrow{\$} \mathsf{Commit}(x,w)\\ \boldsymbol{c} \xleftarrow{\$} \mathsf{Challenge}(x,\boldsymbol{R})\\ \boldsymbol{V} \leftarrow \mathsf{Prove}(x,w,\boldsymbol{U},\boldsymbol{c})\end{array}\right] = 1-\lambda\cdot\epsilon_{\mathsf{IG}}^{\mathsf{Fail}}.$$

□

Theorem 64. *The $\Sigma_{Q,\lambda}^{\mathrm{GAIP}}$ protocol satisfies special soundness with soundness error $2^{-\lambda}$.*

Proof. We consider proofs for statements $Q' \in [Q]$. Let $(\boldsymbol{R},\boldsymbol{c},\boldsymbol{V})$ and $(\boldsymbol{R},\boldsymbol{c}',\boldsymbol{V}')$ be two accepting transcripts such that $\boldsymbol{c} \neq \boldsymbol{c}'$. We provide an extractor algorithm Ext that extracts a witness $U \in \mathbb{G}$ such that $Q' = Q * U$.

$\mathsf{Ext}((\boldsymbol{R},\boldsymbol{c},\boldsymbol{V}),(\boldsymbol{R},\boldsymbol{c}',\boldsymbol{V}')) \to U$**:** On input two accepting transcripts $(\boldsymbol{R},\boldsymbol{c},\boldsymbol{V})$ and $(\boldsymbol{R},\boldsymbol{c}',\boldsymbol{V}')$, the algorithm proceeds as follows:

1. Try to find an index i such that $c_i \neq c_i'$, else abort.
2. If $c_i = 0$ and $c_i' = 1$, return $U := V_i \cdot {V_i'}^{-1}$.
3. If $c_i = 1$ and $c_i' = 0$, return $U := V_i' \cdot V_i^{-1}$

We show that Ext indeed extracts a valid witness U. Consider two accepting transcripts $(\boldsymbol{R},\boldsymbol{c},\boldsymbol{V}),(\boldsymbol{R}',\boldsymbol{c}',\boldsymbol{V}')$ such that $\boldsymbol{c} \neq \boldsymbol{c}'$. Let $i \in [\lambda]$ such that $c_i \neq c_i'$.

If $c_i = 0$, it must hold that $c_i' = 1$ and as $(\boldsymbol{R},\boldsymbol{c},\boldsymbol{V})$ and $(\boldsymbol{R},\boldsymbol{c}',\boldsymbol{V}')$ are both accepting we find that

$$Q * V_i = R_i = Q' * V_i' = Q * A * V_i'.$$

It follows that $Q * (V_i \cdot {V_i'}^{-1}) = Q * A = Q'$, so that $V_i \cdot {V_i'}^{-1}$ is an witness for Q'.

If $c_i = 1$, it must hold that $c_i' = 0$ and as $(\boldsymbol{R},\boldsymbol{c},\boldsymbol{V})$ and $(\boldsymbol{R},\boldsymbol{c}',\boldsymbol{V}')$ are both accepting we find that

$$Q * V_i' = R_i = Q' * V_i = Q * A * V_i.$$

It follows that $Q * (V_i' \cdot V_i^{-1}) = Q * A = Q'$, so that $V_i' \cdot V_i^{-1}$ is an witness for Q'.

We conclude that Ext indeed extracts a valid witness for Q'. The success of Ext, does rely on the fact that $\boldsymbol{c} \neq \boldsymbol{c}'$. For honestly generated transcripts, $\boldsymbol{c}$ and $\boldsymbol{c}'$ are drawn uniformly at random from $\{0,1\}^\lambda$. Therefore, the probability that $\boldsymbol{c} = \boldsymbol{c}'$ under honest generation is $\frac{1}{2^\lambda}$. It follows that the knowledge soundness of $\Sigma_{Q,\lambda}^{\mathrm{GAIP}}$ is $2^{-\lambda}$.

□

Theorem 65. *The $\Sigma_{Q,\lambda}^{\mathrm{GAIP}}$ protocol satisfies perfect honest-verifier zero-knowledge.*

Proof. Denoting $Q_0 = Q$, we construct a simulator Sim that on input a statement $x := Q_1 \in [Q_0]$ produces an accepting transcript for $\Sigma_{Q,\lambda}^{\mathrm{GAIP}}$ according to the same distribution as an honest prover.

$\mathsf{Sim}(Q_1, \lambda) \to (\boldsymbol{R}, \boldsymbol{c}, \boldsymbol{V})$**:** On input Q_1, λ,

1. samples $\boldsymbol{c} \xleftarrow{\$} \{0,1\}^\lambda$;
2. for each $i \in [\lambda]$, samples $(R_i, V_i) \xleftarrow{\$} \mathsf{IG}(Q_{c_i})$;
3. returns $(\boldsymbol{R}, \boldsymbol{c}, \boldsymbol{V})$.

By inspection, Sim outputs accepting transcripts. Let $(\boldsymbol{R}, \boldsymbol{c}, \boldsymbol{V})$ be an honestly generated transcript for the statement Q' and let $(\boldsymbol{R}', \boldsymbol{c}', \boldsymbol{V}') \xleftarrow{\$} \mathsf{Sim}(Q', \lambda)$. Both $\boldsymbol{c}$ and $\boldsymbol{c}'$ are drawn independently and uniformly at random from $\{0,1\}^\lambda$, so they are equally distributed. Furthermore, for $i \in [\lambda]$, R_i is computed as $(R_i, U_i) \xleftarrow{\$} \mathsf{IG}(Q)$, and R'_i as $(R'_i, U'_i) \xleftarrow{\$} \mathsf{IG}(Q)$ or $(R'_i, U'_i) \xleftarrow{\$} \mathsf{IG}(Q')$. As IG is an instance generator (Definition 6) and $Q' \in [Q]$, so $[Q] = [Q']$, one finds that $\boldsymbol{R}$ and $\boldsymbol{R}'$ are also equally distributed, and independent of the sampling of $\boldsymbol{c}$ and $\boldsymbol{c}'$. Lastly, the components of of $\boldsymbol{V}$ and $\boldsymbol{V}'$ are sampled independently and uniformly in the set of group elements X such that $R_i = Q_{c_i} * X$ and X' such that $R'_i = Q_{c_i} * X'$ respectively. $(\boldsymbol{R}, \boldsymbol{c})$ and $(\boldsymbol{R}', \boldsymbol{c}')$ are independently and equally distributed, so $\boldsymbol{V}$ and $\boldsymbol{V}'$ are also independently and equally distributed. Finally, the distribution of $\mathsf{Sim}(Q', \lambda)$ is exactly the same as the one of an honest-verifier accepting transcript. □

G.2 Correctness of $\boldsymbol{\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda}}$

Theorem 66 (Correctness for $\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$). *The $\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$ protocol satisfies correctness,* i.e. *there exists a negligible function* $\mathsf{negl} : \mathbb{N} \to \mathbb{R}_+$ *such that for* $\lambda \in \mathbb{N}$ *and for any valid statement witness pair* (x, w)*,*

$$\Pr\left[\begin{array}{l|l} \mathsf{Verify}(x, \rho_{\mathsf{Prove}_1}, \rho_{\mathsf{Verify}}, \rho_{\mathsf{Prove}_2}) & (\mathsf{st}_{\mathsf{Prove}}, \rho_{\mathsf{Prove}_1}) \xleftarrow{\$} \mathsf{Commit}(x, w) \\ = 1 & \rho_{\mathsf{Verify}} \xleftarrow{\$} \mathsf{Challenge}(x, \rho_{\mathsf{Prove}}) \\ & \rho_{\mathsf{Prove}_2} \leftarrow \mathsf{Prove}(x, w, \mathsf{st}_{\mathsf{Prove}}, \rho_{\mathsf{Verify}}) \end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Proof. Let (x, w) be a statement witness pair, so $x = (P_0, P_1) \in [Q_0] \times [Q_1]$ and $w = (\delta, A_\delta) \in \{0,1\} \times \mathbb{G}$ with $P_\delta = Q_\delta * A_\delta$. Let $\epsilon^{\mathsf{Fail}}_{\mathsf{IG}}$ denote the probability that IG outputs $\perp$. Note that $\epsilon^{\mathsf{Fail}}_{\mathsf{IG}}$ is negligible in λ. $\mathsf{Verify}(x, \rho_{\mathsf{Prove}_1}, \rho_{\mathsf{Verify}}, \rho_{\mathsf{Prove}_2}) = 1$ precisely when

$$\forall i \in [\lambda], \quad c_{0,i} \cdot c_{1,i} = c_i \quad \text{and}$$
$$\forall b \in \{0,1\} \text{ and } \forall i \in [\lambda], \quad \begin{cases} P_b * U_{b,i} = Y_{b,i} & \text{if } c_{b,i} = -1 \\ Q_b * U_{b,i} = Y_{b,i} & \text{if } c_{b,i} = 1 \end{cases},$$

where $\rho_{\mathsf{Prove}_1} = (\boldsymbol{Y}_0, \boldsymbol{Y}_1)$, $\rho_{\mathsf{Verify}} = \boldsymbol{c}$ and $\rho_{\mathsf{Prove}_2} = (\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{U}_0, \boldsymbol{U}_1)$.

During the execution of the protocol $\boldsymbol{c}_{1-\delta}$ is sampled from $\{-1,1\}^\lambda$ and $\boldsymbol{c}_{1-\delta}$ is computed as $\boldsymbol{c}_\delta = (c_k \cdot c_{1-\delta,k})_{k \in [\lambda]}$. It follows directly that $\forall i \in [\lambda]$, $c_{0,i} \cdot c_{1,i} = c_i$.

Furthermore, for $i \in [\lambda]$, $Y_{1-\delta,i}$ and $U_{1-\delta,i}$ are computed as

$$\begin{cases} (Y_{1-\delta,i}, U_{1-\delta,i}) \xleftarrow{\$} \mathsf{IG}(P_{1-\delta}) & \text{if } c_{1-\delta,i} = -1 \\ (Y_{1-\delta,i}, U_{1-\delta,i}) \xleftarrow{\$} \mathsf{IG}(Q_{1-\delta}) & \text{if } c_{1-\delta,i} = 1 \end{cases}$$

Hence $P_{1-\delta} * U_{1-\delta,i} = Y_{1-\delta,i}$ if $c_{1-\delta,i} = -1$ and $Q_{1-\delta} * U_{1-\delta,i} = Y_{1-\delta,i}$ if $c_{1-\delta,i} = 1$.

We know that $Y_{\delta,i}$ and $U_{\delta,i}$ are computed as

$$(Y_{\delta,i}, H_{\delta,i}) \xleftarrow{\$} \mathsf{IG}(Q_\delta) \quad \text{and} \quad U_{\delta,i} = \begin{cases} A_\delta^{-1} \cdot H_{\delta,i} & \text{if } c_{\delta,i} = -1 \\ H_{\delta,i} & \text{if } c_{\delta,i} = 1 \end{cases}.$$

As $Q_\delta * H_{\delta,i} = Y_{\delta,i}$,

$$\begin{cases} Y_{\delta,i} = Q_\delta * H_{\delta,i} = (Q_\delta * A_\delta) * (A_\delta^{-1} H_{\delta,i}) = P_\delta * U_{\delta,i} & \text{if } c_{\delta,i} = -1 \\ Y_{\delta,i} = Q_\delta * H_{\delta,i} = Q_\delta * U_{\delta,i} & \text{if } c_{\delta,i} = -1 \end{cases},$$

and for any $b \in \{0,1\}$ and any $i \in [\lambda]$, if $c_{b,i} = -1$ then $P_b * U_{b,i} = Y_{b,i}$, and if $c_{b,i} = 1$ then $Q_b * U_{b,i} = Y_{b,i}$.

Finally, if the protocol is honestly executed and all calls to IG are successful, then

$$\mathsf{Verify}(x, \rho_{\mathsf{Prove}_1}, \rho_{\mathsf{Verify}}, \rho_{\mathsf{Prove}_2}) = 1.$$

Execution of $\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$ requires 2λ calls to IG. It follows that

$$\Pr\left[\begin{array}{l|l} \mathsf{Verify}(x, \rho_{\mathsf{Prove}_1}, \rho_{\mathsf{Verify}}, \rho_{\mathsf{Prove}_2}) = 1 & \begin{array}{l} (\mathsf{st}_{\mathsf{Prove}}, \rho_{\mathsf{Prove}_1}) \xleftarrow{\$} \mathsf{Commit}(x, w) \\ \rho_{\mathsf{Verify}} \xleftarrow{\$} \mathsf{Challenge}(x, \rho_{\mathsf{Prove}}) \\ \rho_{\mathsf{Prove}_2} \leftarrow \mathsf{Prove}(x, w, \mathsf{st}_{\mathsf{Prove}}, \rho_{\mathsf{Verify}}) \end{array} \end{array}\right] = 1 - 2\lambda \cdot \epsilon^{\mathsf{Fail}}_{\mathsf{IG}}.$$

□

G.3 Extractors and the Special Soundness of $\boldsymbol{\Sigma}^{\mathsf{OR}}_{\boldsymbol{Q_0},\boldsymbol{Q_1},\boldsymbol{\lambda}}$

Lemma 67. *Let $\mathcal{R}^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$ be a* $\mathrm{GAIP}_{\mathsf{IG}}$ *OR relation satisfying Definition 24 for the group action $* : \mathcal{Q} \times \mathbb{G} \to \mathcal{Q}$, $P_0 \in [Q_0]$, $P_1 \in [Q_1]$, $\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}'_0, \boldsymbol{c}'_1 \in \{0,1\}^\lambda$, $\boldsymbol{U}_0, \boldsymbol{U}_1, \boldsymbol{U}'_0, \boldsymbol{U}'_1 \in \mathbb{G}^\lambda$, $\sigma := (\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{U}_0, \boldsymbol{U}_1)$ and $\sigma' := (\boldsymbol{c}'_0, \boldsymbol{c}'_1, \boldsymbol{U}'_0, \boldsymbol{U}'_1)$. For $b \in \{0,1\}$ and $i \in [\lambda]$ let*

$$Y_{b,i} := \begin{cases} P_b * \boldsymbol{U}_{b,i} & \text{if } c_{b,i} = -1 \\ Q_b * \boldsymbol{U}_{b,i} & \text{if } c_{b,i} = 1 \end{cases} \quad \text{and} \quad Y'_{b,i} := \begin{cases} P_b * \boldsymbol{U}'_{b,i} & \text{if } c'_{b,i} = -1 \\ Q_b * \boldsymbol{U}'_{b,i} & \text{if } c'_{b,i} = 1 \end{cases}$$

be such that $\boldsymbol{Y}_0 = \boldsymbol{Y}'_0$ and $\boldsymbol{Y}_1 = \boldsymbol{Y}'_1$. For $b \in \{0,1\}$ and Ext_b the algorithm described in Figure 7, if $\boldsymbol{c}_b \neq \boldsymbol{c}'_b$, then $\mathsf{Ext}_b(\sigma, \sigma')$ outputs A such that $Q_b * A = P_b$.

Proof. Suppose that $\boldsymbol{c}_b \neq \boldsymbol{c}'_b$. Let i be an index of $[\lambda]$ such that $c_{b,i} \neq c'_{b,i}$. If $c_{b,i} = -1$, then $\mathsf{Ext}(\sigma, \sigma')$ outputs $A := U'_{b,i} \cdot U_{b,i}^{-1}$. In this case $c'_{b,i} = 1$. As $P_b * U_{b,i} = Y_{b,i} = Y'_{b,i} = Q_b * U'_{b,i}$, one has $Q_b * (U'_{b,i} \cdot U_{b,i}^{-1}) = P_b$. If $c_{b,i} = 1$, then $\mathsf{Ext}(\sigma, \sigma')$ outputs $A := U_{b,i} \cdot U'^{-1}_{b,i}$. In this case $c'_{b,i} = -1$. As $P_b * U'_{b,i} = Y_{b,i} = Y'_{b,i} = Q * U_{b,i}$, one has $Q * (U_{b,i} \cdot U'^{-1}_{b,i}) = P_b$. □

We give a proof for Theorem 26, which states that $\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$ satisfies special soundness.

Proof. Let $T := (\boldsymbol{Y}_0, \boldsymbol{Y}_1, \boldsymbol{c}, \boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{U}_0, \boldsymbol{U}_1)$ and $T' := (\boldsymbol{Y}_0, \boldsymbol{Y}_1, \boldsymbol{c}', \boldsymbol{c}'_0, \boldsymbol{c}'_1, \boldsymbol{U}'_0, \boldsymbol{U}'_1)$ be two accepting transcripts for a statement $x = (P_0, P_1) \in [Q_0] \times [Q_1]$ such that $\boldsymbol{c} \neq \boldsymbol{c}'$. We provide an algorithm $\mathcal{E}$ that extracts a witness w such that $(x, w) \in \mathcal{R}^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$. Let $\sigma := (\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{U}_0, \boldsymbol{U}_1)$ and $\sigma' := (\boldsymbol{c}'_0, \boldsymbol{c}'_1, \boldsymbol{U}'_0, \boldsymbol{U}'_1)$.

The fact that both transcripts are accepting gives us that $\forall i \in [\lambda], c_{0,i} \cdot c_{1,i} = c_i$ and $\forall i \in [\lambda], c'_{0,i} \cdot c'_{1,i} = c'_i$. As $\boldsymbol{c} \neq \boldsymbol{c}'$ follows that there must exist $b \in \{0,1\}$ and $k \in [\lambda]$ such that $c_{b,k} \neq c'_{b,k}$. First, the algorithm $\mathcal{E}$ finds values $b \in \{0,1\}$ and $k \in [\lambda]$ such that $c_{b,k} \neq c'_{b,k}$. It then runs $A \leftarrow \mathsf{Ext}_b(\sigma, \sigma')$ as defined in Figure 7 and returns (b, A). We verify that $\mathcal{E}$ actually extracts a witness w for x. Both the transcripts T and T' are accepting, so for $b \in \{0,1\}$,

$$\begin{cases} P_b * \boldsymbol{U}_{b,i} & \text{if } c_{b,i} = -1 \\ Q_b * \boldsymbol{U}_{b,i} & \text{if } c_{b,i} = 1 \end{cases} = Y_{b,i} = \begin{cases} P_b * \boldsymbol{U}'_{b,i} & \text{if } c'_{b,i} = -1 \\ Q_b * \boldsymbol{U}'_{b,i} & \text{if } c'_{b,i} = 1 \end{cases}.$$

The extractor $\mathcal{E}$ runs Ext_b for a value b such that $\boldsymbol{c}_b \neq \boldsymbol{c}'_b$. Following Lemma 67, for $A \leftarrow \mathsf{Ext}_b(\sigma, \sigma')$, $Q_b * A = P_b$. Therefore, $w := (b, A)$ is a valid witness for $x := (P_0, P_1)$.

The success of $\mathcal{E}$ relies on the fact that $\boldsymbol{c} \neq \boldsymbol{c}'$. For honestly generated transcripts, $\boldsymbol{c}$ and $\boldsymbol{c}'$ are drawn uniformly at random from $\{0,1\}^\lambda$, and the probability that $\boldsymbol{c} = \boldsymbol{c}'$ is $\frac{1}{2^\lambda}$. Hence, the knowledge soundness of $\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$ is $2^{-\lambda}$.

□

G.4 $\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$ is Honest-Verifier Zero Knowledge

We give a proof for Theorem 27, which states that $\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$ satisfies perfect honest-verifier zero knowledge.

Proof. We first define the algorithm Sim.

$\mathsf{Sim}(x, \lambda) \to (\boldsymbol{Y}_0, \boldsymbol{Y}_1, \boldsymbol{c}, \boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{U}_0, \boldsymbol{U}_1)$**:** on input $x = (P_0, P_1) \in [Q_0] \times [Q_1]$,

1. samples $\boldsymbol{c} \xleftarrow{\$} \{-1,1\}^\lambda$ and $\boldsymbol{c}_0 \xleftarrow{\$} \{-1,1\}^\lambda$ and set $\boldsymbol{c}_1 = (c_{0,k} \cdot c_k)_{k \in [\lambda]}$;
2. for $b \in \{0,1\}$ and $i \in [\lambda]$,
   samples $(Y_{b,i}, U_{b,i}) \xleftarrow{\$} \begin{cases} \mathsf{IG}(P_b) & \text{if } c_{b,i} = -1 \\ \mathsf{IG}(Q_b) & \text{if } c_{b,i} = 1 \end{cases}$;
3. outputs $(\boldsymbol{Y}_0, \boldsymbol{Y}_1, \boldsymbol{c}, \boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{U}_0, \boldsymbol{U}_1)$.

Let $T = (\boldsymbol{Y}_0, \boldsymbol{Y}_1, \boldsymbol{c}, \boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{U}_0, \boldsymbol{U}_1) \xleftarrow{\$} \mathsf{Sim}(x, (Q_0, Q_1), \lambda, s)$ be a transcript that is generated by Sim and let $T' = (\boldsymbol{Y}'_0, \boldsymbol{Y}'_1, \boldsymbol{c}', \boldsymbol{c}'_0, \boldsymbol{c}'_1, \boldsymbol{U}'_0, \boldsymbol{U}'_1)$ be an honestly generated transcript. We first show that T is an accepting transcript. It is clear that for all $i \in [\lambda]$, $c_{0,i} \cdot c_{1,i} = c_i$, due to how Sim generates $\boldsymbol{c}$, $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$. Let $b \in \{0,1\}$ and $i \in [\lambda]$. If $c_{b,i} = -1$, $Y_{b,i}$ and $U_{b,i}$ are generated as $(Y_{b,i}, U_{b,i}) \xleftarrow{\$} \mathsf{IG}(P_b)$, so $Y_{b,i} = P_b * U_{b,i}$, except with probability $\epsilon^{\mathsf{Fail}}_{\mathsf{IG}}$ equal to the one for T', and similarly for $Y_{b,i} = Q_b * U_{b,i}$ if $c_{b,i} = 1$. So, for all $b \in \{0,1\}$ and $i \in [\lambda]$, $P_b * U_{b,i} = Y_{b,i}$ if $c_{b,i} = -1$, and $Q_b * U_{b,i} = Y_{b,i}$ if $c_{b,i} = 1$ and T is an accepting transcript, except with probability $2\epsilon^{\mathsf{Fail}}_{\mathsf{IG}}$ as in the honest case.

We still need to show that the distribution of T matches the distribution of T'. The values of $\boldsymbol{c}$, $\boldsymbol{c}_0$ are sampled independently and uniformly at random from $\{-1,1\}^\lambda$. From the computation of $\boldsymbol{c}_1$ it is clear that $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$ are distributed independently and uniformly in $\{-1,1\}$, as are their counterparts in T'. Moreover, for each index pair (b,i), $(Y_{b,i}, U_{b,i})$ is drawn independently from other components from the distribution $\mathsf{IG}(P_b)$ if $c_{b,i} = -1$ and the distribution $\mathsf{IG}(Q_b)$ if $c_{b,i} = 1$, with $Y_{b,i}$ following the same distribution for both values of $c_{b,i}$, and $U_{b,i}$ uniform among elements X such that $Y_{b,i} = Z_b * X$, where Z_b is equal to P_b if $c_{b,i} = -1$ and Q_b otherwise, and hence uniformly among the set of accepting transcripts with first components $\boldsymbol{Y}_0, \boldsymbol{Y}_1, \boldsymbol{c}, \boldsymbol{c}_0, \boldsymbol{c}_1$. The same goes for the distribution of $\boldsymbol{T}'$, so the distribution on the output transcript T of Sim matches the distribution on the honestly generated transcript T'.

□

G.5 Auxiliary Lemma for the One-More Unforgeability Proof of GABS

Lemma 68. *Consider the non-interactive proof protocol $\Pi^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$ from Definition 28. Let $((P_0,P_1),(\delta,A_\delta)) \in \mathcal{R}^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$ be a statement-witness pair and let π be any accepting proof for the statement (P_0,P_1). There is a unique collection of randomness rand under which π is obtained as the output of $\Pi^{\mathsf{OR}}_{Q_0,Q_1,\lambda}.\mathsf{ComPr}((P_0, P_1),(\delta,A_\delta))$.*

Proof. Let $\mathcal{H}$ be the hash function that is used for the Fiat-Shamir transformation from which $\Pi^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$ is obtained. Let $((P_0,P_1),(\delta,A_\delta)) \in \mathcal{R}^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$ be a statement-witness pair and let $\pi = (\boldsymbol{Y}_0, \boldsymbol{Y}_1, \boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{U}_0, \boldsymbol{U}_1)$ be any accepting proof for the statement (P_0,P_1).

Define $\boldsymbol{c} = \mathcal{H}(\boldsymbol{Y}_0, \boldsymbol{Y}_1)$. Since π is an accepting proof, it holds that for all $b \in \{0,1\}$ and $i \in [\lambda]$, $c_{0,i} \cdot c_{1,i} = c_i$ and

$$Y_{b,i} = \begin{cases} P_b * U_{b,i} & \text{if } c_{b,i} = -1 \\ Q_b * U_{b,i} & \text{if } c_{b,i} = 1 \end{cases}.$$

First we define a collection of randomness for execution of $\Pi^{\mathsf{OR}}_{Q_0,Q_1,\lambda}.\mathsf{ComPr}$ and show that π is obtained from $\Pi^{\mathsf{OR}}_{Q_0,Q_1,\lambda}.\mathsf{ComPr}((P_0,P_1),(\delta,A_\delta))$ under it. We

define for $i \in [\lambda]$,

$$H_{\delta,i} := \begin{cases} A_\delta \cdot U_\delta & \text{if } c_{\delta,i} = -1 \\ U_\delta & \text{if } c_{\delta,i} = 1 \end{cases}$$

and

$$\mathsf{rand} := (\boldsymbol{H}_\delta, \boldsymbol{c}_{1-\delta}, \boldsymbol{U}_{1-\delta}).$$

Let $\pi' = (\boldsymbol{Y}'_0, \boldsymbol{Y}'_1, \boldsymbol{c}'_0, \boldsymbol{c}'_1, \boldsymbol{U}'_0, \boldsymbol{U}'_1)$ be the output of $\Pi^{\mathsf{OR}}_{Q_0,Q_1,\lambda}.\mathsf{ComPr}((P_0, P_1), (\delta, A_\delta))$ under the randomness rand. We show that $\pi' = \pi$. Inspection of the definition of $\Sigma^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$ (Definition 25) shows us that for $i \in [\lambda]$

$$\begin{aligned} Y'_{\delta,i} =& Q_\delta * H_{\delta,i} \\ =& \begin{cases} Q_\delta * (A_\delta \cdot U_{\delta,i}) & \text{if } c_{\delta,i} = -1 \\ Q_\delta * U_{\delta,i} & \text{if } c_{\delta,i} = 1 \end{cases} \\ =& \begin{cases} P_\delta * U_{\delta,i} & \text{if } c_{\delta,i} = -1 \\ Q_\delta * U_{\delta,i} & \text{if } c_{\delta,i} = 1 \end{cases} \\ =& Y_{\delta,i}, \end{aligned}$$

so that $\boldsymbol{Y}'_\delta = \boldsymbol{Y}_\delta$. Furthermore, we find that $\boldsymbol{c}'_{1-\delta} = \boldsymbol{c}_{1-\delta}$, so that for $i \in [\lambda]$

$$\begin{aligned} Y'_{1-\delta,i} =& \begin{cases} P_{1-\delta} \cdot U_{1-\delta,i} & \text{if } c'_{1-\delta,i} = -1 \\ Q_{1-\delta} * U_{1-\delta,i} & \text{if } c'_{1-\delta,i} = 1 \end{cases} \\ =& \begin{cases} P_{1-\delta} \cdot U_{1-\delta,i} & \text{if } c_{1-\delta,i} = -1 \\ Q_{1-\delta} * U_{1-\delta,i} & \text{if } c_{1-\delta,i} = 1 \end{cases} \\ =& Y_{\delta,i}, \end{aligned}$$

so that $\boldsymbol{Y}'_{1-\delta} = \boldsymbol{Y}_{1-\delta}$.

We now define $\boldsymbol{c}' := \mathcal{H}(\boldsymbol{Y}'_0, \boldsymbol{Y}'_1) = \mathcal{H}(\boldsymbol{Y}_0, \boldsymbol{Y}_1) = \boldsymbol{c}$ and find that

$$c'_{\delta,i} = c'_i \cdot c'_{1-\delta,i} = c_i \cdot c_{1-\delta,i} = c_{\delta,i},$$

so that $\boldsymbol{c}'_\delta = \boldsymbol{c}_\delta$.

Lastly, we find that $\boldsymbol{U}'_{1-\delta} = \boldsymbol{U}_{1-\delta}$ and for all $i \in [\lambda]$

$$\begin{aligned} U'_{\delta,i} =& \begin{cases} A_\delta^{-1} \cdot H_{\delta,i} & \text{if } c'_{\delta,i} = -1 \\ H_{\delta,i} & \text{if } c'_{\delta,i} = 1 \end{cases} \\ =& \begin{cases} A_\delta^{-1} \cdot A_\delta \cdot U_{\delta,i} & \text{if } c_{\delta,i} = -1 \\ U_{\delta,i} & \text{if } c_{\delta,i} = 1 \end{cases} \\ =& U_{\delta,i}, \end{aligned}$$

so that $\boldsymbol{U}'_\delta = \boldsymbol{U}_\delta$.

We conclude that $\pi' = \pi$, so that π is the output of $\Pi^{\mathsf{OR}}_{Q_0,Q_1,\lambda}.\mathsf{ComPr}((P_0, P_1), (\delta, A_\delta))$ under the randomness rand. We additionally show that rand is uniquely defined by this. Let $\widetilde{\mathsf{rand}} = (\widetilde{\boldsymbol{H}}_\delta, \widetilde{\boldsymbol{c}}_{1-\delta}, \widetilde{\boldsymbol{U}}_{1-\delta})$ be another set of randomness under which π is obtained from $\Pi^{\mathsf{OR}}_{Q_0,Q_1,\lambda}.\mathsf{ComPr}((P_0, P_1), (\delta, A_\delta))$. We show that $\widetilde{\mathsf{rand}} = \mathsf{rand}$.

Since $\Pi^{\mathsf{OR}}_{Q_0,Q_1,\lambda}.\mathsf{ComPr}((P_0, P_1), (\delta, A_\delta))$ outputs π under both rand' and rand it follows directly that $\widetilde{\boldsymbol{c}}_{1-\delta} = \boldsymbol{c}_{1-\delta}$ and $\widetilde{\boldsymbol{U}}_{1-\delta} = \boldsymbol{U}_{1-\delta}$ as these are directly output as components of π. Furthermore, the value of $\boldsymbol{c}_\delta$ is fixed by π and therefore invariant under wether rand or $\widetilde{\mathsf{rand}}$ is used. We observe that for $i \in [\lambda]$, $U_{\delta,i}$ is computed as

$$U_{\delta,i} = \begin{cases} A_\delta^{-1} \cdot H_{\delta,i} & \text{if } c_{\delta,i} = -1 \\ H_{\delta,i} & \text{if } c_{\delta,i} = 1 \end{cases}$$

under rand, and as

$$U_{\delta,i} = \begin{cases} A_\delta^{-1} \cdot \widetilde{H}_{\delta,i} & \text{if } c_{\delta,i} = -1 \\ \widetilde{H}_{\delta,i} & \text{if } c_{\delta,i} = 1 \end{cases}$$

under $\widetilde{\mathsf{rand}}$. If $c_{\delta,i} = -1$ we find that $\widetilde{H}_{\delta,i} = A_\delta \cdot U_{\delta,i} = H_{\delta,i}$, and if $c_{\delta,i} = 1$ we find that $\widetilde{H}_{\delta,i} = U_{\delta,i} = H_{\delta,i}$. Thus, $\widetilde{\boldsymbol{H}}_\delta = \boldsymbol{H}_\delta$. We conclude that $\widetilde{\mathsf{rand}} = (\widetilde{\boldsymbol{H}}_\delta, \widetilde{\boldsymbol{c}}_{1-\delta}, \widetilde{\boldsymbol{U}}_{1-\delta}) = (\boldsymbol{H}_\delta, \boldsymbol{c}_{1-\delta}, \boldsymbol{U}_{1-\delta}) = \mathsf{rand}$. □

H Proofs for the GABS scheme

H.1 Correctness of GABS.

Theorem 69 (Correctness of GABS). *The* GABS *blind signature scheme satisfies correctness;* i.e.*, there exists a negligible function* $\mathsf{negl} : \mathbb{N} \to \mathbb{R}_+$ *such that for any* λ *in* $\mathbb{N}$, $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$, *and message* $m \in \{0,1\}^*$,

$$\Pr\left[\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1 \;\middle|\; \begin{array}{l} (\mathsf{st}_{\mathsf{Sign}}, \rho_{\mathsf{Sign},1}) \xleftarrow{\$} \mathsf{Sign}_1(\mathsf{sk}, \mathsf{pk}), \\ (\mathsf{st}_{\mathsf{Use}}, \rho_{\mathsf{Use}}) \xleftarrow{\$} \mathsf{Use}_1(m, \rho_{\mathsf{Sign},1}), \\ \rho_{\mathsf{Sign}_2} \leftarrow \mathsf{Sign}_2(\mathsf{sk}, \mathsf{st}_{\mathsf{Sign}}, \rho_{\mathsf{Use}}), \\ \sigma \leftarrow \mathsf{Use}_2(\mathsf{st}_{\mathsf{Use}}, \rho_{\mathsf{Sign},2}) \end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Proof. Let λ in $\mathbb{N}$ and let $\epsilon^{\mathsf{Fail}}_{\mathsf{IG}}$ denote the probability, that the instance generator IG outputs $\perp$. Note that $\epsilon^{\mathsf{Fail}}_{\mathsf{IG}}$ is negligible in λ. Let

$$\begin{aligned} \mathsf{sk} &= (\delta, A_\delta) \in \{0,1\} \times [Q] \quad \text{and} \\ \mathsf{pk} &= (\mathsf{pk}_0 := (P_0, \pi_0, \pi_1), \mathsf{pk}_1 := (P_1, \pi_0, \pi_1)) \end{aligned}$$

be a keypair generated by $\mathsf{KeyGen}(1^\lambda)$. In particular, $P_\delta = Q * A_\delta$.

Let $m \in \{0,1\}^*$ and let $\sigma = (\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{U}_0, \boldsymbol{U}_1)$ be generated honestly according to the GABS protocol. We first show that $\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1$ if all calls to IG are successful.

In order to show that $\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1$, one needs to check that

$$\mathcal{H}(\boldsymbol{Z}_0||\boldsymbol{Z}_1||m) = (c_{0,i} \cdot c_{1,i})_{i\in[\lambda]} \tag{3}$$

where for $b \in \{0,1\}$ and $i \in [\lambda]$

$$\begin{cases} Z_{b,i} := P_b * U_{b,i} & \text{if } c_{b,i} = -1 \\ Z_{b,i} := Q * U_{b,i} & \text{if } c_{b,i} = 1 \end{cases}.$$

Let $\boldsymbol{Y}_0$ and $\boldsymbol{Y}_1$ be as specified in in the GABS protocol for the generation of σ. We first show that for $b \in \{0,1\}$,

$$\boldsymbol{Z}_b = \boldsymbol{Y}_b. \tag{4}$$

We first consider the case $b = \delta$. Let $k \in [\lambda]$. Observe that $Y_{\delta,k}$ is computed as $(Y_{\delta,k}, V_{\delta,k}) \xleftarrow{\$} \mathsf{IG}\left(W_{\delta,k}^{(d_{\delta,k})}\right)$, where $W_{\delta,k}^{(d_{\delta,k})}$ is computed as

$$\begin{cases} (W_{\delta,k}^{(d_{\delta,k})}, H_{\delta,k}^{(1)}) \xleftarrow{\$} \mathsf{IG}(Q) & \text{if } d_{\delta,k} = 1 \\ (W_{\delta,k}^{(d_{\delta,k})}, H_{\delta,k}^{(-1)}) \xleftarrow{\$} \mathsf{IG}(Q) & \text{if } d_{\delta,k} = -1 \end{cases}$$

and $d_{\delta,k} \xleftarrow{\$} \{-1,1\}$ is sampled uniformly at random. It follows that

$$\begin{cases} Y_{\delta,k} = Q * (H_{\delta,k}^{(1)} \cdot V_{\delta,k}) & \text{if } d_{\delta,k} = 1 \\ Y_{\delta,k} = Q * (H_{\delta,k}^{(-1)} \cdot V_{\delta,k}) & \text{if } d_{\delta,k} = -1 \end{cases}. \tag{5}$$

Furthermore, $U_{\delta,k}$ is computed as

$$U_{\delta,k} = \begin{cases} U_{\delta,k}^{*(1)} \cdot V_{\delta,k} & \text{if } d_{b,k} = 1 \\ U_{\delta,k}^{*(-1)} \cdot V_{\delta,k} & \text{if } d_{b,k} = -1 \end{cases}, \tag{6}$$

where $V_{\delta,k}$ is as before and $U_{\delta,k}^{*(1)}$ and $U_{\delta,k}^{*(-1)}$ are computed as

$$U_{\delta,k}^{*(1)} = \begin{cases} H_{\delta,i}^{(1)} & \text{if } c_{\delta,k}^* = 1 \\ A_\delta^{-1} \cdot H_{\delta,i}^{(1)} & \text{if } c_{\delta,k}^* = -1 \end{cases} \quad \text{and} \tag{7}$$

$$U_{\delta,k}^{*(-1)} = \begin{cases} A_\delta^{-1} \cdot H_{\delta,k}^{(-1)} & \text{if } c_{\delta,k}^* = 1 \\ H_{\delta,k}^{(-1)} & \text{if } c_{\delta,k}^* = -1 \end{cases},$$

where $H_{\delta,k}^{(1)}$ is as before and $c_{\delta,k}^*$ is a value in $\{-1,1\}$.

The value $c_{\delta,k}$ is computed as $c_{\delta,k} = c_{\delta,k}^* \cdot d_{\delta,k}$ and we consider all four possible values for $(d_{\delta,k}, c_{\delta,k}^*) \in \{-1,1\}^2$ and show that the equality in (4) holds.

If $(d_{\delta,k}, c^*_{\delta,k}) = (1,1)$, then $c_{\delta,k} = c^*_{\delta,k} \cdot d_{\delta,k} = 1$. We then find that

$$\begin{aligned}
\boldsymbol{Y}_{\delta,k} &= Q * (H^{(1)}_{\delta,k} \cdot V_{\delta,k}) && \text{by (5)} \\
&= Q * (H^{(1)}_{\delta,k} \cdot (U^{*(1)}_{\delta,k})^{-1} \cdot U_{\delta,k}) && \text{by (6)} \\
&= Q * (U^{*(1)}_{\delta,k} \cdot (U^{*(1)}_{\delta,k})^{-1} \cdot U_{\delta,k}) && \text{by (7)} \\
&= Q * U_{\delta,k},
\end{aligned}$$

so that (4) is satisfied.

If $(d_{\delta,k}, c^*_{\delta,k}) = (-1,1)$, then $c_{\delta,k} = c^*_{\delta,k} \cdot d_{\delta,k} = -1$. We then find that

$$\begin{aligned}
\boldsymbol{Y}_{\delta,k} &= Q * (H^{(-1)}_{\delta,k} \cdot V_{\delta,k}) && \text{by (5)} \\
&= Q * (H^{(-1)}_{\delta,k} \cdot (U^{*(-1)}_{\delta,k})^{-1} \cdot U_{\delta,k}) && \text{by (6)} \\
&= Q * (A_\delta \cdot U^{*(-1)}_{\delta,k} \cdot (U^{*(-1)}_{\delta,k})^{-1} \cdot U_{\delta,k}) && \text{by (7)} \\
&= (Q * A_\delta) * U_{\delta,k} \\
&= P_\delta * U_{\delta,k}
\end{aligned}$$

so that (4) is again satisfied. The validation for the cases $(d_{\delta,k}, c^*_{\delta,k}) = (1,-1)$ and $(d_{\delta,k}, c^*_{\delta,k}) = (-1,-1)$ are analogous to the previous cases. We find that for $b = \delta$ and all $k \in [\lambda]$, the equation in (4) is satisfied.

We also consider the case $b = 1-\delta$. Let $k \in [\lambda]$. We show that the equation in (4) is satisfied. The value $Y_{1-\delta,k}$ is computed as $(Y_{1-\delta,k}, V_{1-\delta,k}) \xleftarrow{\$} \mathsf{IG}\left(W^{(d_{1-\delta,k})}_{1-\delta,k}\right)$ where $d_{1-\delta,k} \xleftarrow{\$} \{-1,1\}$ is sampled uniformly at random and $c^*_{1-\delta,k}$ is a value in $\{-1,1\}$. If $c^*_{1-\delta,k} = 1$, $W^{(d_{1-\delta,k})}_{1-\delta,k}$ is computed as

$$\begin{cases}
(W^{(d_{1-\delta,k})}_{1-\delta,k}, U^{*(1)}_{1-\delta,k}) \xleftarrow{\$} \mathsf{IG}(Q) & \text{if } d_{1-\delta,k} = 1 \\
(W^{(d_{1-\delta,k})}_{1-\delta,k}, U^{*(-1)}_{1-\delta,k}) \xleftarrow{\$} \mathsf{IG}(P_{1-\delta}) & \text{if } d_{1-\delta,k} = -1
\end{cases}$$

while if $c^*_{1-\delta,k} = -1$, $W^{(d_{1-\delta,k})}_{1-\delta,k}$ is computed as

$$\begin{cases}
(W^{(d_{1-\delta,k})}_{1-\delta,k}, U^{*(1)}_{1-\delta,k}) \xleftarrow{\$} \mathsf{IG}(P_{1-\delta}) & \text{if } d_{1-\delta,k} = 1 \\
(W^{(d_{1-\delta,k})}_{1-\delta,k}, U^{*(-1)}_{1-\delta,k}) \xleftarrow{\$} \mathsf{IG}(Q) & \text{if } d_{1-\delta,k} = -1
\end{cases}.$$

So

$$\begin{cases}
Y_{1-\delta,k} = Q * (U^{*(1)}_{1-\delta,k} \cdot V_{1-\delta,k}) & \text{if } d_{1-\delta,k} = 1 \text{ and } c^*_{1-\delta,k} = 1 \\
Y_{1-\delta,k} = P_{1-\delta} * (U^{*(-1)}_{1-\delta,k} \cdot V_{1-\delta,k}) & \text{if } d_{1-\delta,k} = -1 \text{ and } c^*_{1-\delta,k} = 1 \\
Y_{1-\delta,k} = P_{1-\delta} * (U^{*(1)}_{1-\delta,k} \cdot V_{1-\delta,k}) & \text{if } d_{1-\delta,k} = 1 \text{ and } c^*_{1-\delta,k} = -1 \\
Y_{1-\delta,k} = Q * (U^{*(-1)}_{1-\delta,k} \cdot V_{1-\delta,k}) & \text{if } d_{1-\delta,k} = -1 \text{ and } c^*_{1-\delta,k} = -1
\end{cases}. \tag{8}$$

Moreover, $U_{1-\delta,k}$ is computed as

$$U_{1-\delta,k} = \begin{cases}
U^{*(1)}_{1-\delta,k} \cdot V_{1-\delta,k} & \text{if } d_{1-\delta,k} = 1 \\
U^{*(-1)}_{1-\delta,k} \cdot V_{1-\delta,k} & \text{if } d_{1-\delta,k} = -1.
\end{cases}$$

with $U^{*(1)}_{1-\delta,k}, U^{*(1)}_{1-\delta,k}$ and $V_{1-\delta,k}$, as in (8). The value $c_{1-\delta,k}$ is computed as $c_{1-\delta,k} = c^*_{1-\delta,k} \cdot d_{1-\delta,k}$. By substituting $U_{1-\delta,k}$ into (8) and using the fact that $c_{1-\delta,k} = c^*_{1-\delta,k} \cdot d_{1-\delta,k}$, we obtain

$$\begin{cases} Y_{1-b,k} = P_b * U_{1-b,k} & \text{if } c_{1-b,k} = -1 \\ Y_{1-b,k} = Q * U_{1-b,k} & \text{if } c_{1-b,k} = 1 \end{cases},$$

so equation (4) is satisfied.

We now show that (3) holds. Let $\boldsymbol{h} := \mathcal{H}(\boldsymbol{Z}_0||\boldsymbol{Z}_1||m) = \mathcal{H}(\boldsymbol{Y}_0||\boldsymbol{Y}_1||m)$ and we note that $\{\delta, 1-\delta\} = \{0,1\}$. Let $k \in [\lambda]$. The value $c_{\delta,k}$ is computed as $c_{\delta,k} = c^*_{\delta,k} \cdot d_{\delta,k}$, where $d_{\delta,k} \xleftarrow{\$} \{-1,1\}$ is sampled uniformly at random and $c^*_{\delta,k}$ is computed as $c^*_{\delta,k} = c'_k \cdot c^*_{1-\delta,k}$ for c'_k which is computed as $c'_k = h_k \cdot d_{\delta,k} \cdot d_{1-\delta,k}$. $c^*_{1-\delta,k}$ is sampled uniformly on $\{-1,1\}$. Note that for any value $v \in \{-1,1\}$, $v^2 = 1$. Then,

$$\begin{aligned} c_{\delta,k} &= h_k \cdot d_{\delta,k} \cdot d_{1-\delta,k} \cdot c^*_{1-\delta,k} \cdot d_{\delta,k} \\ &= h_k \cdot d_{1-\delta,k} \cdot c^*_{1-\delta,k}. \end{aligned}$$

Furthermore, $c_{1-\delta,k}$ is computed as $c_{1-\delta,k} = c^*_{1-\delta,k} \cdot d_{1-\delta,k}$ where $c^*_{1-\delta,k}, d_{1-\delta,k} \in \{-1,1\}$.

We find that for all $k \in [\lambda]$,

$$\begin{aligned} c_{0,k} \cdot c_{1,k} &= c_{\delta,k} \cdot c_{1-\delta,k} \\ &= h_k \cdot d_{1-\delta,k} \cdot c^*_{1-\delta,k} \cdot c^*_{1-\delta,k} \cdot d_{1-\delta,k} \\ &= h_k, \end{aligned}$$

and equality (3) holds, for any $b \in \{0,1\}$ and $k \in [\lambda]$. Thus, if all calls to IG are successful, an honestly GABS signature passes verification.

Including the generation of the non-interactive OR proofs $\boldsymbol{\pi}_0^{(\pm 1)}$ and $\boldsymbol{\pi}_1^{(\pm 1)}$, execution of the GABS scheme requires $8\lambda^2 + 6\lambda$ calls to IG. It follows that

$$\begin{aligned} &\Pr\left[\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1 \;\middle|\; \begin{array}{l} (\mathsf{st}_{\mathsf{Sign}}, \rho_{\mathsf{Sign},1}) \xleftarrow{\$} \mathsf{Sign}_1(\mathsf{sk}, \mathsf{pk}), \\ (\mathsf{st}_{\mathsf{Use}}, \rho_{\mathsf{Use}}) \xleftarrow{\$} \mathsf{Use}_1(m, \rho_{\mathsf{Sign},1}), \\ \rho_{\mathsf{Sign}_2} \leftarrow \mathsf{Sign}_2(\mathsf{sk}, \mathsf{st}_{\mathsf{Sign}}, \rho_{\mathsf{Use}}), \\ \sigma \leftarrow \mathsf{Use}_2(\mathsf{st}_{\mathsf{Use}}, \rho_{\mathsf{Sign},2}) \end{array}\right] \\ &= 1 - \epsilon^{\mathsf{Fail}}_{\mathsf{GABS}}. \end{aligned}$$

where $\epsilon^{\mathsf{Fail}}_{\mathsf{GABS}} := \left(8\lambda^2 + 6\lambda\right) \cdot \epsilon^{\mathsf{Fail}}_{\mathsf{IG}}$ is negligible in λ.

□

H.2 Blindness of GABS.

Theorem 70 (Blindness under chosen keys for GABS). *The GABS blind signature protocol satisfies blindness under chosen keys: for any PPT algorithm*

$\mathcal{A}$ and natural integer λ,

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{Blind}}_{\mathcal{A}}(\lambda)=1\right]-\frac{1}{2}\right| \leq \frac{4\lambda+1}{2^{\lambda-1}}+2\epsilon^{\mathsf{GABS}}_{\mathsf{Fail}}.$$

Proof. As the group in our generalised GAIP may be infinite, and without a uniform distribution, this proof differs from previous ones in the blind signature state-of-the-art. Let us show through a sequence of games that a PPT adversary $\mathcal{A}$ will have a negligible advantage in distinguishing between the original security game and one in which it is provided no information depending on the bit b it is trying to guess.

Game $\mathbf{G}_0$: is the original blindness game. The adversary's advantage is denoted $\mathsf{Adv}^{\mathsf{Blind}}_{\mathcal{A}}(\lambda)$.

Game $\mathbf{G}_1$: is as the previous game, except that now, the challenger calls the extractor algorithm from the proof for special soundness for $\Sigma^{\mathsf{OR}}_{Q,P_0,\lambda}$ and $\Sigma^{\mathsf{OR}}_{Q,P_1,\lambda}$, that extracts witnesses $(\boldsymbol{U}^{*(\beta)}_{\alpha,\mathcal{Y}})_{\alpha\in\{0,1\},\beta\in\{-1,1\},\mathcal{Y}\in\{\mathcal{U},\mathcal{V}\}}$ from $\mathcal{A}$ for the statements in the proofs $(\boldsymbol{\pi}^{(\beta)}_{\alpha,\mathcal{Y}})_{\alpha,\beta,\mathcal{Y}}$ returned by $\mathcal{A}_1$ in each signer first message for users $\mathcal{U}$ and $\mathcal{V}$. If any of these extractions fail, the challenger aborts and replaces $\mathcal{A}$'s guess b' by a uniformly random bit. As the $\boldsymbol{\pi}^{(\beta)}_{b,\mathcal{Y}}$ vectors have length λ, the probability for the adversary to be able to distinguish between this game and the previous one is bounded by 8λ times the soundness error of the proofs: $\left|\mathsf{Adv}^{\mathbf{G}_1}_{\mathcal{A}}(\lambda)-\mathsf{Adv}^{\mathsf{Blind}}_{\mathcal{A}}(\lambda)\right| \leq 8\lambda\cdot\mathsf{Adv}^{\mathsf{Snd}}_{\mathcal{A}}(\lambda) \leq \lambda\cdot 2^{3-\lambda}$.

Game $\mathbf{G}_2$: is as the previous game, except that now, the challenger calls the extractor algorithm from the special soundness for $\Sigma_{Q,\lambda}$ to obtain witnesses A_0 and A_1 from π_0 and π_1 respectively in the public key provided by $\mathcal{A}$. If the extraction fails, it aborts and replaces $\mathcal{A}$'s guess b' by a uniformly random bit. The adversary's advantage in distinguishing this game from the previous one is bounded by twice the soundness error of the proofs: $\left|\mathsf{Adv}^{\mathbf{G}_2}_{\mathcal{A}}(\lambda)-\mathsf{Adv}^{\mathbf{G}_1}_{\mathcal{A}}(\lambda)\right| \leq 2\cdot\mathsf{Adv}^{\mathsf{Snd}}_{\mathcal{A}}(\lambda) \leq 2^{1-\lambda}$.

Game $\mathbf{G}_3$: is as the previous game, except that now, if σ_0 and σ_1 are both valid, the challenger picks $\delta \xleftarrow{\$} \{0,1\}$, sets $\mathsf{sk} \leftarrow (\delta, A_\delta)$ from A_0 and A_1 obtained by extraction, and replaces the σ_0 and σ_1 provided to $\mathcal{A}_2$ by the respective self-generated signatures

$$\widetilde{\sigma}_0 = (\widetilde{\boldsymbol{c}}_0, \widetilde{\boldsymbol{c}}_1, \widetilde{\boldsymbol{U}}_0, \widetilde{\boldsymbol{U}}_1) \xleftarrow{\$} \langle \mathsf{BSign}(\mathsf{sk}), \mathsf{BUser}(\mathsf{pk}, m_0)\rangle$$
$$\text{and} \qquad \widetilde{\sigma}_1 = (\widetilde{\boldsymbol{b}}_0, \widetilde{\boldsymbol{b}}_1, \widetilde{\boldsymbol{T}}_0, \widetilde{\boldsymbol{T}}_1) \xleftarrow{\$} \langle \mathsf{BSign}(\mathsf{sk}), \mathsf{BUser}(\mathsf{pk}, m_0)\rangle,$$

except if either $\widetilde{\sigma}_0$ or $\widetilde{\sigma}_1$ is equal to $\perp$, which happens with probability $\epsilon^{\mathsf{GABS}}_{\mathsf{Fail}}$. In this game, for $\widetilde{\sigma}_0$, each $(\widetilde{Y}_{\alpha,i}, \widetilde{V}_{\alpha,i})$ is drawn from $\mathsf{IG}(\widetilde{W}_{\alpha,i})$ for some $\widetilde{W}_{\alpha,i} \in [Q]$, whereas in the previous one each $(Y_{\alpha,i}, V_{\alpha,i})$ was drawn from $\mathsf{IG}(W_{\alpha,i})$, where the challenger knew witnesses $B_{\alpha,i}$ such that $W_{\alpha,i} = Q{*}B_{\alpha,i}$. Hence, as IG satisfies Definition 6 for instance generators, the distributions of $(\widetilde{\boldsymbol{Y}}_{\alpha,u})_{\alpha,u\in\{0,1\}}$ and $(\boldsymbol{Y}_{\alpha,u})_{\alpha,u\in\{0,1\}}$ are identical.

Additionally, in the random oracle model, $\boldsymbol{c} \leftarrow \mathcal{H}(\boldsymbol{Y}_0 \| \boldsymbol{Y}_1 \| m_0)$ and $\widetilde{\boldsymbol{c}} \leftarrow \mathcal{H}(\widetilde{\boldsymbol{Y}}_0 \| \widetilde{\boldsymbol{Y}}_1 \| m_0)$ both follow independent uniform distributions on $\{-1,1\}^\lambda$, and act as a perfect one time pad on the $(\boldsymbol{d}_\alpha)_\alpha$ and $(\widetilde{\boldsymbol{d}}_\alpha)_\alpha$ respectively in $\rho_{\mathsf{Use}} = (c_k \cdot d_{0,k} \cdot d_{1,k})_k$ and $\widetilde{\rho}_{\mathsf{Use}} = (\widetilde{c}_k \cdot \widetilde{d}_{0,k} \cdot \widetilde{d}_{1,k})_k$ transmitted to $\mathcal{A}_1$. Then, as the $(\boldsymbol{d}_\alpha)_\alpha$ and $(\widetilde{\boldsymbol{d}}_\alpha)_\alpha$ are sampled uniformly at random from $\{-1,1\}^\lambda$, the $\boldsymbol{c}_\alpha = (c^*_{\alpha,k} \cdot d_{\alpha,k})_k$ and $\widetilde{\boldsymbol{c}}_\alpha = (\widetilde{c}^*_{\alpha,k} \cdot \widetilde{d}_{\alpha,k})_k$, for adversary-provided $c^*_{\alpha,k}$'s and $\widetilde{c}^*_{\alpha,k}$'s, are uniform in $\{-1,1\}^\lambda$.

Moreover, $Y_{\alpha,i} = Q * A_\alpha^{(1-c_{\alpha,i})/2} U_{\alpha,i}$ with $U_{\alpha,i}$ distributed uniformly at random over the variables X satisfying that equation, as $V_{\alpha,i}$ is drawn uniform at random from the variables Z such that $Y_{\alpha,i} = W_{\alpha,i} * Z$. Similarly $\widetilde{Y}_{\alpha,i} = Q * A_\alpha^{(1-\widetilde{c}_{\alpha,i})/2} \widetilde{U}_{\alpha,i}$ with $\widetilde{U}_{\alpha,i}$ distributed uniformly at random over the variables X satisfying that equation. We already found that $(Y_{\alpha,i}, c_{\alpha,i})$ and $(\widetilde{Y}_{\alpha,i}, \widetilde{c}_{\alpha,i})$ independently follow identical distributions, so, $\boldsymbol{U}_\alpha$ and $\widetilde{\boldsymbol{U}}_\alpha$ are also equally distributed, and hence, σ_0 and $\widetilde{\sigma}_0$. The same reasoning also applies to $\widetilde{\sigma}_1$ and σ_1, so finally, $\left|\mathsf{Adv}^{\mathbf{G}_3}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathbf{G}_2}_{\mathcal{A}}(\lambda)\right| \leq 2\epsilon^{\mathsf{GABS}}_{\mathsf{Fail}}$.

Game $\mathbf{G}_4$: is as the previous game, except that now, $\mathcal{U}$ is set to be $\mathsf{BUser}(\mathsf{pk}, m_{1-b})$ and $\mathcal{V}$ to be $\mathsf{BUser}(\mathsf{pk}, m_b)$. As the only $\mathcal{U}$ and $\mathcal{V}$ dependent states that $\mathcal{A}$ receives are respectively $\rho_{\mathcal{U}} = (c^{\mathcal{U}}_k \cdot d^{\mathcal{U}}_{0,k} \cdot d^{\mathcal{U}}_{1,k})_k$ and $\rho_{\mathcal{V}} = (c^{\mathcal{V}}_k \cdot d^{\mathcal{V}}_{0,k} \cdot d^{\mathcal{V}}_{1,k})_k$, where, in the random oracle model, $(c^{\mathcal{U}}_k)_k$ and $(c^{\mathcal{V}}_k)_k$ are uniformly random in $\{-1,1\}$, they both follow the same distribution, and $\mathsf{Adv}^{\mathbf{G}_4}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathbf{G}_3}_{\mathcal{A}}(\lambda)$. Hence, the adversary's view in this last game is independent of the bit b, and $\mathsf{Adv}^{\mathbf{G}_4}_{\mathcal{A}}(\lambda) = 0$.

Finally, $\mathsf{Adv}^{\mathsf{Blind}}_{\mathcal{A}}(\lambda) \leq (4\lambda+1)2^{1-\lambda} + 2\epsilon^{\mathsf{GABS}}_{\mathsf{Fail}}$.

□

H.3 One-More Unforgeability of GABS

We rely on Theorem 49 from [KLLQ23] in order to prove that GABS satisfies one-more unforgeability as described in Definition 17. This section works towards proving k-one-more unforgeability of GABS (Theorem 84). In order to prove this theorem we follow the proof strategy from [KLX22] as adapted in [KLLQ23].

In the OMUF game, the adversary is allowed to make hash queries and signing queries, and the goal for the adversary outputs more verifying message signature pairs than the amount of signing queries that it made. Following [KLX22, Section 4.1], a hash query is said to correspond to a message signature pair (m, σ) if the input for the hash query is the hash input computed in $\mathsf{GABS.Verify}(\mathsf{pk}, m, \sigma)$.

Lemma 71. *The conditions in Definition 38 are satisfied for* GABS.

Proof. By inspection of the definition of GABS, conditions 1. and 4. from Definition 38 are satisfied. Signatures produced under the GABS protocol take the form of OR-proofs for $\Sigma^{\mathsf{OR}}_{Q,Q,\lambda}$ for an NP OR-relation that uses the same NP relation for $\mathrm{GAIP}_{\mathsf{IG}}$ for both branches, so that condition 1. is satisfied. Let us denote

this NP OR-relation by $\mathcal{R}^{\mathsf{OR}}_{Q,Q,\lambda} \subset \mathcal{X}^{\mathsf{OR}} \times \mathcal{W}^{\mathsf{OR}}$, following Definition 24, with $\mathcal{X}^{\mathsf{OR}} = [Q]^2$ and $\mathcal{W}^{\mathsf{OR}} = \{0,1\} \times \mathbb{G}$. The challenge space of the sigma protocol for this NP-relation is the set $\{-1,1\}^\lambda$, in which ρ_{Use} lies, so that condition 4. is satisfied.

We now show that condition 2. is satisfied. Recall that the GABS construction considers a fixed value $Q \in \mathcal{Q}$ and public keys take the form of $\mathsf{pk} = ((P_0, \pi_0, \pi_1), (P_1, \pi_0, \pi_1))$, where (P_0, P_1) describes a statement in $\mathcal{X}^{\mathsf{OR}}$. This allows us to the define a set of corresponding secret keys

$$\mathcal{S}_{\mathsf{pk}} := \{(\delta, W_\delta) \in \{0,1\} \times \mathbb{G} \mid ((P_0, P_1), (\delta, W_\delta)) \in \mathcal{R}^{\mathsf{OR}}\}$$

and a partitioning $\mathcal{S}_{\mathsf{pk},b} := \{(\delta, W_\delta) \in \mathcal{S}_{\mathsf{pk}} \mid \delta = b\}$ for $b \in \{0,1\}$. To show that condition 2. is satisfied, we need to show that there exists a bijection $\phi_{\mathsf{pk}} : \mathcal{S}_{\mathsf{pk},0} \to \mathcal{S}_{\mathsf{pk},1}$. For this, it suffices to show that $\mathcal{S}_{\mathsf{pk},0}$ and $\mathcal{S}_{\mathsf{pk},1}$ are both in bijection with $\mathrm{Stab}_{\mathbb{G}}(Q)$. For $b \in \{0,1\}$, observe that

$$\mathcal{S}_{\mathsf{pk}} = \{(b, W) \mid W \in \mathbb{G}, P_b = Q * W\}$$

and recall that $\mathrm{Stab}_{\mathbb{G}}(Q) := \{A \in \mathbb{G} \mid Q = Q * A\}$. Also not that for an honestly generated public key pk, $\mathcal{S}_{\mathsf{pk}}$ is non-empty. Let B be in $\mathcal{S}_{\mathsf{pk}}$, and set

$$\begin{aligned} \xi : \mathrm{Stab}_{\mathbb{G}}(Q) &\to \mathcal{S}_{\mathsf{pk},b} & \qquad \zeta : \mathcal{S}_{\mathsf{pk},b} &\to \mathrm{Stab}_{\mathbb{G}}(Q) \\ A &\mapsto (b, AB), & (b, W) &\mapsto WB^{-1}. \end{aligned}$$

For A in $\mathrm{Stab}_{\mathbb{G}}(Q)$, $Q * (AB) = Q * B = P_b$, so that $(b, AB) \in \mathcal{S}_{\mathsf{pk},b}$ and ξ is well defined. For (b, W) in $\mathcal{S}_{\mathsf{pk},b}$, $Q * (WB^{-1}) = P_b * B^{-1} = Q * (BB^{-1}) = Q$, so WB^{-1} is in $\mathrm{Stab}_{\mathbb{G}}(Q)$ and ζ is well-defined.

It is straightforward to check that ξ and ζ are each other's inverses, thus providing bijections between $\mathrm{Stab}_{\mathbb{G}}(Q)$ and $\mathcal{S}_{\mathsf{pk},b}$, and condition 2. is satisfied.

In order to see that condition 3. is satisfied, we refer to the discussion in [KLX22, Section 4.1] which almost directly applies to the GABS construction. The argument from [KLX22] requires that it can be assumed that no two distinct message signature pairs correspond to the same hash query. Their argument for this relies on the discrete logarithm problem, which is not applicable to us. In our case, if two distinct message signature pairs correspond to the same hash query, then the extractors in Figure 7 can be used to extract one of the two possible secret keys corresponding to the public key, solving one of two $\mathrm{GAIP}_{\mathsf{IG}}$ instances embedded in the public key.

Inspection of GABS shows that condition 4. is also satisfied.

Later in this section, in Figure 11, we provide an explicit sampling algorithm for GABS instances. The first line of this algorithm samples a bit b uniformly at random and independently from other any variables. The output is ensured to be a b-side instance, hence condition 5. is also satisfied.

□

In the rest of the section we use the overhead arrow vector notation introduced in Section C.

Instances for GABS. We discuss what 0-side and 1-side instances look like for GABS. A b-side instance

$$I_b = (b, A_b, \mathsf{pk}_{1-b}, \overrightarrow{\boldsymbol{H}_b^{(1)}}, \overrightarrow{\boldsymbol{H}_b^{(-1)}}, \overrightarrow{\boldsymbol{c}_{1-b}^*}, \overrightarrow{\boldsymbol{U}_{1-b}^{*(1)}}, \overrightarrow{\boldsymbol{U}_{1-b}^{*(-1)}}, \overrightarrow{\Pi}),$$

for $b \in \{0,1\}$, is defined as follows:

- (b, A_b) is the secret key for if $\delta = b$.
- $\mathsf{pk}_{1-b} = (P_{1-b}, \pi_0, \pi_1)$ is the part of the public key $\mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1)$ whose secret is unknown.
- Consider $\overrightarrow{\boldsymbol{H}_b^{(1)}}$ of which we denote the j-th component by $\boldsymbol{H}_b^{(1)(j)}$ for $j \in [k+1]$. The vector $\boldsymbol{H}_b^{(1)(j)}$ corresponds to the value of $\boldsymbol{H}_b^{(1)}$ for the commitment $\boldsymbol{W}_b^{(1)}$ in the j-th first signer message such that

  $$W_{b,i}^{(1)} = Q * H_{b,i}^{(1)} \text{ for all } i \in [\lambda]$$

  when $\delta = b$.
- Consider $\overrightarrow{\boldsymbol{H}_b^{(-1)}}$ of which we denote the j-th component by $\boldsymbol{H}_b^{(-1)(j)}$ for $j \in [k+1]$. The vector $\boldsymbol{H}_b^{(-1)(j)}$ corresponds to the value of $\boldsymbol{H}_b^{(-1)}$ for the commitment $\boldsymbol{W}_b^{(-1)}$ in the j-th first signer message such that

  $$W_{b,i}^{(-1)} = Q * H_{b,i}^{(-1)} \text{ for all } i \in [\lambda]$$

  when $\delta = b$.
- Consider $\overrightarrow{\boldsymbol{c}_{1-b}^*}$ of which we denote the j-th component by ${\boldsymbol{c}_{1-b}^*}^{(j)}$ for $j \in [k+1]$. The vector ${\boldsymbol{c}_{1-b}^*}^{(j)}$ describes the simulated challenge $\boldsymbol{c}_{1-b}^*$ in the j-th first signer message for if $\delta = b$.
- Consider $\overrightarrow{\boldsymbol{U}_{1-b}^{*(1)}}$ of which we denote the j-th component by ${\boldsymbol{U}_{1-b}^{*(1)}}^{(1)(j)}$ for $j \in [k+1]$. The vector ${\boldsymbol{U}_{1-b}^{*(1)}}^{(j)}$ corresponds to the value of $\boldsymbol{U}_{1-b}^{*(1)}$ for the commitment $\boldsymbol{W}_{1-b}^{(1)}$ in the j-th first signer message such that

  $$W_{1-b,i}^{(1)} = Q * U_{1-b,i}^{*(1)} \text{ for all } i \in [\lambda]$$

  when $\delta = b$.
- Consider $\overrightarrow{\boldsymbol{U}_{1-b}^{*(-1)}}$ of which we denote the j-th component by $\boldsymbol{U}_{1-b}^{*(-1)(j)}$ for $j \in [k+1]$. The vector $\boldsymbol{U}_{1-b}^{*(-1)(j)}$ corresponds to the value of $\boldsymbol{U}_{1-b}^{*(-1)}$ for the commitment $\boldsymbol{W}_{1-b}^{(-1)}$ in the j-th first signer message such that

  $$W_{1-b,i}^{(-1)} = Q * U_{1-b,i}^{*(-1)} \text{ for all } i \in [\lambda]$$

  when $\delta = b$.
- Consider $\overrightarrow{\Pi} := (\overrightarrow{\Pi_b^{(\beta)}})_{b,\beta}$ of which we denote the j-th component by $\Pi^{(j)} = (\Pi_b^{(\beta)(j)})_{b,\beta}$ for $j \in [k+1]$. The collection Π contains all internal randomness

used to generate the proofs $(\boldsymbol{\pi}_0^{(1)}, \boldsymbol{\pi}_0^{(-1)}, \boldsymbol{\pi}_1^{(1)}, \boldsymbol{\pi}_1^{(-1)})$ showing that the entries of $\boldsymbol{W}_0^{(\pm 1)}$ and $\boldsymbol{W}_1^{(\pm 1)}$ lie in the orbit of Q, P_0 or P_1 under the group action. Specifically, $\Pi_{b,i}^{(\beta)(j)} = (\widetilde{\boldsymbol{H}}_\delta, \widetilde{\boldsymbol{c}}_{1-\delta}, \widetilde{\boldsymbol{U}}_{1-\delta})$ contains the randomness used to generate the proof $\pi_{b,i}^{(\beta)}$ in the j-th signing session.

We may assume without loss of generality that for all $j \in [k+1]$, none of the components of $\boldsymbol{H}_b^{(1)(j)}$, $\boldsymbol{H}_b^{(-1)(j)}$, $\boldsymbol{U}_{1-b}^{*(1)(j)}$ or $\boldsymbol{U}_{1-b}^{*(-1)(j)}$ are equal to $\perp$, as this corresponds to the first signer algorithm aborting before interaction with the adversary. Similarly we may assume that the proofs $(\boldsymbol{\pi}_0^{(1)}, \boldsymbol{\pi}_0^{(-1)}, \boldsymbol{\pi}_1^{(1)}, \boldsymbol{\pi}_1^{(-1)})$ are generated successfully under the randomness in $\Pi^{(j)}$.

Explicit definition for $\Phi_{\mathsf{rand},\overrightarrow{h}}$. We now construct the map $\Phi_{\mathsf{rand},\overrightarrow{h}}$ that we need to show that Lemma 47 holds for GABS. We can define $\Phi_{\mathsf{rand},\overrightarrow{h}}$ mostly explicitly and first provide that part of its definition.

Let rand, $\overrightarrow{h}$ be fixed. For every honestly generated public key pk, let $\phi_{\mathsf{pk}} : \mathcal{S}_{\mathsf{pk},0} \to \mathcal{S}_{\mathsf{pk},1}$ be the bijection from condition 2. from Definition 38, which is satisfied by GABS. We define $\phi_{\mathsf{pk},0} := \phi_{\mathsf{pk}}$ and $\phi_{\mathsf{pk},1} := \phi_{\mathsf{pk}}^{-1}$. We define a map $\Phi_{\mathsf{rand},\overrightarrow{h}}$ that maps 0-side instances to 1-side instances and vice versa. We consider values $b \in \{0,1\}$. Provided a b-side instance

$$I_b = (b, A_b, \mathsf{pk}_{1-b}, \overrightarrow{\boldsymbol{H}_b^{(1)}}, \overrightarrow{\boldsymbol{H}_b^{(-1)}}, \overrightarrow{\boldsymbol{c}_{1-b}^{*}}, \overrightarrow{\boldsymbol{U}_{1-b}^{*(1)}}, \overrightarrow{\boldsymbol{U}_{1-b}^{*(-1)}}, \overrightarrow{\Pi})$$

such that $(I, \mathsf{rand}, \overrightarrow{h}) \in \mathsf{Succ}$, we know the following:

- The b-side instance I_b, together with fixed values for rand and $\overrightarrow{h}$, fixes a query transcript $\overrightarrow{\boldsymbol{c}^*} = \overrightarrow{e}(I, \mathsf{rand}, \overrightarrow{h})$.
- The public key component pk_{1-b} is of the form (P_{1-b}, π_0, π_1).
- The public key component pk_b can be described using components of I_b as $\mathsf{pk}_b = (P_b, \pi_0, \pi_1) := (Q * A_b, \pi_0, \pi_1)$, allowing for the information theoretic retrieval of pk from I_b.

The map $\Phi_{\mathsf{rand},\overrightarrow{h}}$ is defined as follows. It maps a b-side instance I_b as above to

$$\Phi_{\mathsf{rand},\overrightarrow{h}}(I_b) = (1-b, A_{1-b}, \mathsf{pk}_b := (Q * A_b, \pi_0, \pi_1), \overrightarrow{\boldsymbol{H}_{1-b}^{(1)}}, \overrightarrow{\boldsymbol{H}_{1-b}^{(-1)}}, \overrightarrow{\boldsymbol{c}_b^{*}}, \overrightarrow{\boldsymbol{U}_b^{*(1)}}, \overrightarrow{\boldsymbol{U}_b^{*(-1)}}, \overrightarrow{\Theta})$$

where A_{1-b} is defined by $\phi_{\mathsf{pk},b}(b, A_b) = (1-b, A_{1-b})$ and satisfies $P_{1-b} = Q * A_{b-1}$. For $j \in [k+1]$ and $i \in [\lambda]$,

$$H_{1-b,i}^{(1)(j)} := \begin{cases} A_{1-b} \cdot U_{1-b,i}^{*(1)\ (j)} & \text{if } c_{1-b,i}^{*\ (j)} = -1 \\ U_{1-b,i}^{*(1)\ (j)} & \text{if } c_{1-b,i}^{*\ (j)} = 1 \end{cases},$$

$$H_{1-b,i}^{(-1)(j)} := \begin{cases} U_{1-b,i}^{*(-1)(j)} & \text{if } c_{1-b,i}^{*\ (j)} = -1 \\ A_{1-b} \cdot U_{1-b,i}^{*(-1)(j)} & \text{if } c_{1-b,i}^{*\ (j)} = 1 \end{cases},$$

$$c_{b,i}^{*\ (j)} := c_i^{*(j)} \cdot c_{1-b,i}^{*\ (j)}, \qquad \text{with } \overrightarrow{\boldsymbol{c}^*} = \overrightarrow{e}(I_b, \mathsf{rand}, \overrightarrow{h})$$

$$U_{b,i}^{*(1)(j)} := \begin{cases} A_b^{-1} \cdot H_{b,i}^{(1)(j)} & \text{if } c_{b,i}^{*\ (j)} = -1 \\ H_{b,i}^{(1)(j)} & \text{if } c_{b,i}^{*\ (j)} = 1 \end{cases},$$

$$U_{b,i}^{*(-1)(j)} := \begin{cases} H_{b,i}^{(-1)(j)} & \text{if } c_{b,i}^{*\ (j)} = -1 \\ A_b^{-1} \cdot H_{b,i}^{(-1)(j)} & \text{if } c_{b,i}^{*\ (j)} = 1 \end{cases}.$$

The definition of $\overrightarrow{\Theta}$ is implicit. Lemma 68 can be used to show that there exists a unique value for $\overrightarrow{\Theta}$ such that $\Phi_{\mathsf{rand},\overrightarrow{h}}$ maintains transcripts. We define $\overrightarrow{\Theta}$ to be exactly this value, but we will come back to this.

Lemma 72. *The map $\Phi_{\mathsf{Rand},\overrightarrow{h}}$ respects the components $\left(\overrightarrow{\boldsymbol{W}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W}_1^{(\pm 1)}}\right)$ of the first signer messages $\overrightarrow{\rho_{\mathsf{Sign}_1}}$. That is, for any instance I and for the first signer messages from the transcripts $\mathsf{trans}(I, \mathsf{rand}, \overrightarrow{h})$ and $\mathsf{trans}(\Phi_{\mathsf{rand},\overrightarrow{h}}(I), \mathsf{rand}, \overrightarrow{h})$, the components $\left(\overrightarrow{\boldsymbol{W}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W}_1^{(\pm 1)}}\right)$ are equal. Note that the computation of these components does not depend on the definition of $\overrightarrow{\Theta}$ for $\Phi_{\mathsf{rand},\overrightarrow{h}}$, which we still need to define.*

Proof. Let I_b be a b-side instance and let rand and $\overrightarrow{h}$ be fixed. The components $\left(\overrightarrow{\boldsymbol{W}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W}_1^{(\pm 1)}}\right)$ in the first signer message in the transcript $\mathsf{trans}(I_b, \mathsf{rand}, \overrightarrow{h})$ are equal to

$$W_{b,i}^{(1)(j)} := Q * H_{b,i}^{(1)(j)}, \qquad W_{1-b,i}^{(1)(j)} := \begin{cases} P_{1-b} * U_{1-b,i}^{*(1)(j)} & \text{if } c_{1-b,i}^{*(j)} = -1 \\ Q * U_{1-b,i}^{*(1)(j)} & \text{if } c_{1-b,i}^{*(j)} = 1 \end{cases},$$

$$W_{b,i}^{(-1)(j)} := Q * H_{b,i}^{(-1)(j)}, \quad W_{1-b,i}^{(-1)(j)} := \begin{cases} Q * U_{1-b,i}^{*(-1)(j)} & \text{if } c_{1-b,i}^{*(j)} = -1 \\ P_{1-b} * U_{1-b,i}^{*(-1)(j)} & \text{if } c_{1-b,i}^{*(j)} = 1 \end{cases}$$

and the components $\left(\overrightarrow{\boldsymbol{W'}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W'}_1^{(\pm 1)}}\right)$ from the first signer message in the transcript $\mathsf{trans}(\Phi_{\mathsf{rand},\overrightarrow{h}}(I_b), \mathsf{rand}, \overrightarrow{h})$ to

$$W'^{(1)(j)}_{1-b,i} := Q * H^{(1)(j)}_{1-b,i}, \qquad W'^{(1)(j)}_{b,i} := \begin{cases} P_b * U^{*(1)(j)}_{b,i} & \text{if } c^{*(j)}_{b,i} = -1 \\ Q * U^{*(1)(j)}_{b,i} & \text{if } c^{*(j)}_{b,i} = 1 \end{cases},$$

$$W'^{(-1)(j)}_{1-b,i} := Q * H^{(-1)(j)}_{1-b,i}, \quad W'^{(-1)(j)}_{b,i} := \begin{cases} Q * U^{*(-1)(j)}_{b,i} & \text{if } c^{*(j)}_{b,i} = -1 \\ P_b * U^{*(-1)(j)}_{1-b,i} & \text{if } c^{*(j)}_{b,i} = 1 \end{cases}.$$

Let us now prove that

$$W^{(1)(j)}_{b,i} = W'^{(1)(j)}_{b,i}, \qquad W^{(-1)(j)}_{b,i} = W'^{(-1)(j)}_{b,i},$$
$$W^{(1)(j)}_{1-b,i} = W'^{(1)(j)}_{1-b,i}, \qquad W^{(-1)(j)}_{1-b,i} = W'^{(-1)(j)}_{1-b,i}.$$

From the definition of $\Phi_{\mathsf{rand},\overrightarrow{h}}$,

$$H^{(j)}_{1-b,i} = \begin{cases} A_{1-b} \cdot U^{*(1)\ (j)}_{1-b,i} & \text{if } c^{*}_{1-b,i}{}^{(j)} = -1 \\ U^{*(1)\ (j)}_{1-b,i} & \text{if } c^{*}_{1-b,i}{}^{(j)} = 1, \end{cases}$$

$$\begin{aligned} W'^{(1)(j)}_{1-b,i} = Q * H^{(1)(j)}_{1-b,i} &= \begin{cases} Q * (A_{1-b} \cdot U^{*(1)\ (j)}_{1-b,i}) & \text{if } c^{*}_{1-b,i}{}^{(j)} = -1 \\ Q * U^{*(1)\ (j)}_{1-b,i} & \text{if } c^{*}_{1-b,i}{}^{(j)} = 1 \end{cases} \\ &= \begin{cases} P_{1-b} * U^{*(1)\ (j)}_{1-b,i} & \text{if } c^{*}_{1-b,i}{}^{(j)} = -1 \\ Q * U^{*(1)\ (j)}_{1-b,i} & \text{if } c^{*}_{1-b,i}{}^{(j)} = 1 \end{cases} \\ &= W^{(1)(j)}_{1-b,i}. \end{aligned}$$

From the description of $\Phi_{\mathsf{rand},\overrightarrow{h}}$,

$$U^{*(1)(j)}_{b,i} = \begin{cases} A_b^{-1} \cdot H^{(1)(j)}_{b,i} & \text{if } c^{*}_{b,i}{}^{(j)} = -1 \\ H^{(1)(j)}_{b,i} & \text{if } c^{*}_{b,i}{}^{(j)} = 1 \end{cases},$$

where $P_b = Q * A_b$. So,

$$\begin{aligned} W'^{(1)(j)}_{b,i} &= \begin{cases} P_{1-b} * U^{*(1)(j)}_{b,i} & \text{if } c^{*(j)}_{b,i} = -1 \\ Q * U^{*(1)(j)}_{b,i} & \text{if } c^{*(j)}_{b,i} = 1 \end{cases} \\ &= \begin{cases} Q * (A_b \cdot A_b^{-1} \cdot H^{(1)(j)}_{b,i}) & \text{if } c^{*(j)}_{b,i} = -1 \\ Q * H^{(1)(j)}_{b,i} & \text{if } c^{*(j)}_{b,i} = 1 \end{cases} \\ &= Q * H^{(1)(j)}_{b,i} \\ &= W^{(1)(j)}_{b,i}. \end{aligned}$$

The verification process for the equations ${W'}_{1-b,i}^{(-1)(j)} = W_{1-b,i}^{(-1)(j)}$ and ${W'}_{b,i}^{(-1)(j)} = W_{b,i}^{(-1)(j)}$ is analogous.

□

Defining $\overrightarrow{\Theta}$ to completely set $\Phi_{\mathsf{rand},\overrightarrow{h}}$. For $j \in [k+1]$ and $i \in [\lambda]$, let

$$\begin{aligned} W_{b,i}^{(1)(j)} &= {W'}_{b,i}^{(1)(j)}, & W_{b,i}^{(-1)(j)} &= {W'}_{b,i}^{(-1)(j)}, \\ W_{1-b,i}^{(1)(j)} &= {W'}_{1-b,i}^{(1)(j)}, & W_{1-b,i}^{(-1)(j)} &= {W'}_{1-b,i}^{(-1)(j)}. \end{aligned}$$

be as in the proof for Lemma 72. The value of $\overrightarrow{\Theta}$ is uniquely determined by the condition that $\forall j \in [k+1]$, $\forall \beta \in \{-1,1\}$ and $\forall i \in [\lambda]$:

1. The output of $\Pi^{\mathsf{OR}}_{Q,P_{1-b},\lambda}.\mathsf{ComPr}((W'^{(\beta)(j)}_{1-b,i}, W'^{(\beta)(j)}_{1-b,i}), (0, H^{(\beta)(j)}_{1-b,i}))$ under the randomness $\overrightarrow{\Theta}$ is equal to the output of
$$\Pi^{\mathsf{OR}}_{Q,P_{1-b},\lambda}.\mathsf{ComPr}\left(\left(W^{(\beta)}_{1-b,i}, W^{(\beta)}_{1-b,i}\right), \left(\frac{1-\beta\cdot c^*_{1-b,i}}{2}, U^{*(\beta)}_{1-b,i}\right)\right)$$
under the randomness $\overrightarrow{\Pi}$.
2. The output of $\Pi^{\mathsf{OR}}_{Q,P_b,\lambda}.\mathsf{ComPr}\left(\left(W'^{(\beta)}_{b,i}, W'^{(\beta)}_{b,i}\right), \left(\frac{1-\beta\cdot c^*_{b,i}}{2}, U^{*(\beta)}_{b,i}\right)\right)$ under the randomness $\overrightarrow{\Theta}$ equals the output of
$$\Pi^{\mathsf{OR}}_{Q,P_b,\lambda}.\mathsf{ComPr}((W^{(\beta)(j)}_{b,i}, W^{(\beta)(j)}_{b,i}), (0, H^{(\beta)(j)}_{b,i}))$$
under the randomness $\overrightarrow{\Pi}$.

This properly defines $\overrightarrow{\Theta}$ by Lemma 72 and Lemma 68. We have now fully defined $\Phi_{\mathsf{rand},\overrightarrow{h}}$ and by inspection, $\Phi_{\mathsf{rand},\overrightarrow{h}}$ maps the b-side instance I_b to a $(1-b)$-side instance.

$\Phi_{\mathsf{rand},\overrightarrow{h}}$ Maintains First Signer Messages.

Lemma 73. *Let I be an instance and fix values for rand and $\overrightarrow{h}$. Let $\overrightarrow{\rho_{\mathsf{Sign}_1}}$ be the collection of first signer messages of the transcript $\mathsf{trans}(I, \mathsf{rand}, \overrightarrow{h})$ and let $\overrightarrow{\rho'_{\mathsf{Sign}_1}}$ be the first signer messages of the transcript $\mathsf{trans}(\Phi_{\mathsf{rand},\overrightarrow{h}}(I), \mathsf{rand}, \overrightarrow{h})$. Then, $\overrightarrow{\rho_{\mathsf{Sign}_1}} = \overrightarrow{\rho'_{\mathsf{Sign}_1}}$.*

Proof. For $j \in [k+1]$, $\rho^{(j)}_{\mathsf{Sign}_1}$ and $\rho'^{(j)}_{\mathsf{Sign}_1}$ are of the form

$$\begin{aligned} \rho^{(j)}_{\mathsf{Sign}_1} &= (\boldsymbol{W}_0^{(1)}, \boldsymbol{W}_0^{(-1)}, \boldsymbol{\pi}_0^{(1)}, \boldsymbol{\pi}_0^{(-1)}, \boldsymbol{W}_1^{(1)}, \boldsymbol{W}_1^{(-1)}, \boldsymbol{\pi}_1^{(1)}, \boldsymbol{\pi}_1^{(-1)}), \\ \rho'^{(j)}_{\mathsf{Sign}_1} &= (\boldsymbol{W'}_0^{(1)}, \boldsymbol{W'}_0^{(-1)}, \boldsymbol{\pi'}_0^{(1)}, \boldsymbol{\pi'}_0^{(-1)}, \boldsymbol{W'}_1^{(1)}, \boldsymbol{W'}_1^{(-1)}, \boldsymbol{\pi'}_1^{(1)}, \boldsymbol{\pi'}_1^{(-1)}). \end{aligned}$$

By Lemma 72 it holds that ${W'}_0^{(\pm 1)} = W_0^{(\pm 1)}$ and ${W'}_1^{(\pm 1)} = W_1^{(\pm 1)}$. Furthermore, the conditions that define $\overrightarrow{\Theta}$ in the definition of $\Phi_{\mathsf{rand},\overrightarrow{h}}$ impose precisely that $\boldsymbol{\pi}_0^{(\pm 1)} = {\boldsymbol{\pi}'}_0^{(\pm 1)}$ and $\boldsymbol{\pi}_1^{(\pm 1)} = {\boldsymbol{\pi}'}_1^{(\pm 1)}$. Thus, $\overrightarrow{\rho_{\mathsf{Sign}_1}} = \overrightarrow{\rho'_{\mathsf{Sign}_1}}$.

□

Sampling of Random Triples As discussed in Section C.1, let $\mathcal{I}$ be the set of all instances, $\mathcal{R}$ be the set of all random tapes to run the adversary on and $(\{0,1\}^\lambda)^{k+1}$ denote the set of all possible combined random oracle responses.

Recall that the idea of the OMUF proof strategy is to provide a reduction to the $\mathrm{GAIP}_{\mathsf{IG}}$ problem (Section C). For the reduction, when the instance containing signer randomness is initialised, the $\mathrm{GAIP}_{\mathsf{IG}}$ challenge that we wish to solve is already known. In order for the reduction to solve this challenge, it needs to be embedded in the signers public key. For Q and a $\mathrm{GAIP}_{\mathsf{IG}}$ challenge Q' fixed, we define a distribution $\mathcal{D}$ on $\mathcal{I} \times \mathcal{R} \times (\{0,1\}^\lambda)^{k+1}$ under which we sample triples from $\mathcal{I} \times \mathcal{R} \times (\{0,1\}^\lambda)^{k+1}$ such that the instance from $\mathcal{I}$ has Q' embedded in the public key.

We define the procedure $\mathsf{Sampl}_{\mathcal{I}}$ in Figure 11 that on input $Q \in \mathcal{Q}$, a $\mathrm{GAIP}_{\mathsf{IG}}$ challenge Q' and the security parameter λ samples a random instance from $\mathcal{I}$ which has Q' embedded in one of the public key components. Additionally, a proof validating the public key components P_0 and P_1 according to the proof system $\Pi_{Q,\lambda}$ needs to be provided. To generate such a proof for the component containing Q', we use the simulator algorithm of the honest-verifier zero-knowledge proof for $\Sigma_{Q,\lambda}^{\mathrm{GAIP}}$ (Theorem 65), denoted $\mathsf{Sim}_{\Sigma_{Q,\lambda}^{\mathrm{GAIP}}}$. To ensure that the proof verifies the $\mathsf{Sampl}_{\mathcal{I}}$ algorithm is allowed to prime one query to the random oracle $\mathcal{H}$. This can be safely done as the reduction requires no random oracle queries to be made until after this step is finished.

We define $\mathcal{D}$ to be the distribution on $(I, \mathsf{rand}, \overrightarrow{h})$ when I is sampled from $\mathcal{I}$ according to $\mathsf{Sampl}_I(Q, Q', \lambda)$, rand is sampled from $\mathcal{R}$ uniformly at random and $\overrightarrow{h}$ is sampled uniformly at random from $(\{0,1\}^\lambda)^{k+1}$.

The following lemma 74 is necessary to invoke Lemma 51 as explained in Section C.1.

Lemma 74. *Let $X = (I, \mathsf{rand}, \overrightarrow{h})$ be sampled according to the distribution $\mathcal{D}$. For any successful tuple $X_0 = (I'_0, \mathsf{rand}', \overrightarrow{h'})$ and $X_1 = (I'_1, \mathsf{rand}', \overrightarrow{h'}) := \Phi(I_0, \mathsf{rand}', \overrightarrow{h'})$ its image under Φ,*

$$\Pr_{X \xleftarrow{\$} D}[X = X_0] = \Pr_{X \xleftarrow{\$} D}[X = X_1].$$

Proof. We remark that $\mathsf{rand} \xleftarrow{\$} \mathcal{R}$ and $\overrightarrow{h} \xleftarrow{\$} (\{0,1\}^\lambda)^{k+1}$ are both uniformly sampled at random and independently from I which is sampled as $(Q', A) \xleftarrow{\$} \mathsf{IG}(Q)$, $\mathcal{I} \xleftarrow{\$} \mathsf{Sampl}_{\mathcal{I}}(Q, Q', \lambda)$. The map Φ acts as the identity on its second and

$\mathsf{Sampl}_{\mathcal{I}}(Q, Q', \lambda)$

sample $b \xleftarrow{\$} \{0,1\}$

set $P_{1-b} = Q'$, $(\boldsymbol{R}, \boldsymbol{c}, \boldsymbol{V}) \xleftarrow{\$} \mathsf{Sim}_{\Sigma^{\mathrm{GAIP}}_{Q,\lambda}}(Q', \lambda)$, $\pi_{1-b} \leftarrow (\boldsymbol{R}, \boldsymbol{V})$ and $\mathcal{H}(\boldsymbol{R}) := \boldsymbol{c}$

set $(P_b, A_b) \xleftarrow{\$} \mathsf{IG}(Q)$, $\pi_b \xleftarrow{\$} \Pi_{Q,\lambda}.\mathsf{ComPr}(P_b, A_b)$, $\mathsf{pk}_{1-b} \leftarrow (P_{1-b}, \pi_0, \pi_1)$

for $j \in [k+1]$: $\boldsymbol{c}^{*(j)}_{1-b} \xleftarrow{\$} \{0,1\}^{\lambda}$

  for $i \in [\lambda]$, $\beta \in \{-1,1\}$, $\left(W^{(\beta)(j)}_{b,i}, H^{(\beta)(j)}_{b,i}\right) \xleftarrow{\$} \mathsf{IG}(Q)$,

    if $c^{*(j)}_{1-b,i} = -1$, $\left(W^{(\beta)(j)}_{1-b,i}, \boldsymbol{U}^{*(\beta)(j)}_{1-b,i}\right) \xleftarrow{\$} \mathsf{IG}(Q)$,

      else $\left(W^{(\beta)(j)}_{1-b,i}, \boldsymbol{U}^{*(\beta)(j)}_{1-b,i}\right) \xleftarrow{\$} \mathsf{IG}(P_{b-1})$

    $\widetilde{\boldsymbol{c}}^{(\beta)(j)}_{(b,i),1} \xleftarrow{\$} \{-1,1\}^{\lambda}$, $\widetilde{\delta} \leftarrow \frac{1-\beta \cdot c^{*(j)}_{1-b,i}}{2}$, $\widetilde{\boldsymbol{c}}^{(\beta)(j)}_{(1-b,i),1-\widetilde{\delta}} \xleftarrow{\$} \{-1,1\}^{\lambda}$

    for $l \in [\lambda]$: sample $\widetilde{H}^{(\beta)(j)}_{(b,i),0,l} \xleftarrow{\$} \mathsf{IG}(Q)$

      if $\widetilde{c}^{(\beta)(j)}_{(b,i),1,l} = -1$, $\widetilde{U}^{(\beta)(j)}_{(b,i),1,l} \xleftarrow{\$} \mathsf{IG}(W^{(\beta)(j)}_{b,i})$ **else** $\widetilde{U}^{(\beta)(j)}_{(b,i),1,l} \xleftarrow{\$} \mathsf{IG}(P_b)$

      if $\widetilde{\delta} = 0$, $\widetilde{H}^{(\beta)(j)}_{(1-b,i),\widetilde{\delta},l} \xleftarrow{\$} \mathsf{IG}(Q)$, **else** $\widetilde{H}^{(\beta)(j)}_{(1-b,i),\widetilde{\delta},l} \xleftarrow{\$} \mathsf{IG}(P_{1-b})$

      if $\widetilde{c}^{(\beta)(j)}_{(1-b,i),1-\widetilde{\delta},l} = -1$, sample $\widetilde{U}^{(\beta)(j)}_{(1-b,i),1-\widetilde{\delta},l} \xleftarrow{\$} \mathsf{IG}(W^{(\beta)(j)}_{1-b,i})$

      else :

        if $\widetilde{\delta} = 0$, $\widetilde{U}^{(\beta)(j)}_{(1-b,i),1-\widetilde{\delta},l} \xleftarrow{\$} \mathsf{IG}(P_{1-b})$, **else** $\widetilde{U}^{(\beta)(j)}_{(1-b,i),1-\widetilde{\delta},l} \xleftarrow{\$} \mathsf{IG}(Q)$

    set $\Pi^{(\beta)(j)}_{b,i} \leftarrow \left(\widetilde{\boldsymbol{H}}^{(\beta)(j)}_{(b,i),(0)}, \widetilde{\boldsymbol{c}}^{(\beta)(j)}_{(b,i)(1)}, \widetilde{\boldsymbol{U}}^{(\beta)(j)}_{(b,i)(1)}\right)$

    set $\Pi^{(\beta)(j)}_{1-b,i} \leftarrow \left(\widetilde{\boldsymbol{H}}^{(\beta)(j)}_{(1-b,i)\widetilde{\delta}}, \widetilde{\boldsymbol{c}}^{(\beta)(j)}_{(1-b,i),1-\widetilde{\delta}}, \widetilde{\boldsymbol{U}}^{(\beta)(j)}_{(1-b,i),1-\widetilde{\delta}}\right)$

set $\overrightarrow{\Pi} \leftarrow \left(\overrightarrow{\boldsymbol{\Pi}^{(-1)}_0}, \overrightarrow{\boldsymbol{\Pi}^{(1)}_0}, \overrightarrow{\boldsymbol{\Pi}^{(-1)}_1}, \overrightarrow{\boldsymbol{\Pi}^{(1)}_1}\right)$

return $\left(b, A_b, \mathsf{pk}_{1-b}, \overrightarrow{\boldsymbol{H}^{(1)}_b}, \overrightarrow{\boldsymbol{H}^{(-1)}_b}, \overrightarrow{\boldsymbol{c}^{*}_{1-b}}, \overrightarrow{\boldsymbol{U}^{*(1)}_{1-b}}, \overrightarrow{\boldsymbol{U}^{*(-1)}_{1-b}}, \overrightarrow{\Pi}\right)$

Figure 11. Sampling process for instances according to honest execution of the GABS signer protocol.

third argument, therefore it suffices to show that

$$\Pr_{I \xleftarrow{\$} \mathsf{Sampl}_{\mathcal{I}}(Q,Q',\lambda)}[I = I_0] = \Pr_{I \xleftarrow{\$} \mathsf{Sampl}_{\mathcal{I}}(Q,Q',\lambda)}[I = I_1].$$

where $(Q', A) \xleftarrow{\$} \mathsf{IG}(Q)$.

We consider the random variable

$$I := (b, A_b, \mathsf{pk}_{1-b}, \overrightarrow{\boldsymbol{H}_b^{(1)}}, \overrightarrow{\boldsymbol{H}_b^{(-1)}}, \overrightarrow{\boldsymbol{c}^*_{1-b}}, \overrightarrow{\boldsymbol{U}^{*(1)}_{1-b}}, \overrightarrow{\boldsymbol{U}^{*(-1)}_{1-b}}, \overrightarrow{\Pi}) \xleftarrow{\$} \mathsf{Sampl}_{\mathcal{I}}(Q, Q', \lambda)$$

for any output $(Q', A) \xleftarrow{\$} \mathsf{IG}(Q)$. Furthermore, we define $P_b := Q * A_b$, $(P_{1-b}, \pi_0, \pi_1) := \mathsf{pk}_{1-b}$, and for $\beta \in \{-1, 1\}$, $j \in [k+1]$ and $i \in [\lambda]$,

$$W_{b,i}^{(\beta)(j)} := Q * H_{b,i}^{(\beta)(j)} \text{ and } W_{1-b,i}^{(\beta)(j)} := \begin{cases} Q * U_{1-b,i}^{*(\beta)(j)} & \text{if } c_{b,i}^{*(j)} = -1 \\ P_{1-b} * U_{1-b,i}^{*(\beta)(j)} & \text{if } c_{b,i}^{*(j)} = -1 \end{cases}.$$

Define $(\boldsymbol{\pi}_0^{(\pm 1)}, \boldsymbol{\pi}_0^{(\pm 1)})$ to be the proofs testifying the honest generation of $\left(\overrightarrow{\boldsymbol{W}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W}_1^{(\pm 1)}}\right)$ generated under I, specifically using randomness $\overrightarrow{\Pi}$.

Let I'_0 be the instance from the first component of any successful tuple X_0 and we parse

$$I'_0 = (d, A'_d, \mathsf{pk}'_{1-d}, \overrightarrow{\boldsymbol{H'}^{(1)}_d}, \overrightarrow{\boldsymbol{H'}^{(-1)}_d}, \overrightarrow{\boldsymbol{c'}^*_{1-d}}, \overrightarrow{\boldsymbol{U'}^{*(1)}_{1-d}}, \overrightarrow{\boldsymbol{U'}^{*(-1)}_{1-d}}, \overrightarrow{\Pi'})$$
$$I'_1 = (1-d, A'_{1-d}, \mathsf{pk}'_d, \overrightarrow{\boldsymbol{H'}^{(1)}_{1-d}}, \overrightarrow{\boldsymbol{H'}^{(-1)}_{1-d}}, \overrightarrow{\boldsymbol{c'}^*_d}, \overrightarrow{\boldsymbol{U'}^{*(1)}_d}, \overrightarrow{\boldsymbol{U'}^{*(-1)}_d}, \overrightarrow{\Theta'}) := \Phi_{\mathsf{rand}, \overrightarrow{h}}(I'_0)$$

with $P'_d := Q * A'_d$, $P'_{1-d} := Q * A'_{1-d}$, $\mathsf{pk}'_{1-d} = (P'_{1-d}, \pi'_0, \pi'_1)$, $\mathsf{pk}'_d = (P'_d, \pi'_0, \pi'_1)$ and for $\beta \in \{-1, 1\}$, $j \in k+1$ and $i \in [\lambda]$,

$$W'^{(\beta)(j)}_{d,i} := Q * H'^{(\beta)(j)}_{d,i} = \begin{cases} Q * U'^{*(\beta)(j)}_{d,i} & \text{if } \boldsymbol{c}'_{d,i} = -1 \\ P'_d * U'^{*(\beta)(j)}_{d,i} & \text{if } \boldsymbol{c}'_{d,i} = 1 \end{cases} \text{ and}$$
$$W'^{(\beta)(j)}_{1-d,i} := Q * H'^{(\beta)(j)}_{1-d,i} = \begin{cases} Q * U'^{*(\beta)(j)}_{1-d,i} & \text{if } \boldsymbol{c}'_{1-d,i} = -1 \\ P'_{1-d} * U'^{*(\beta)(j)}_{1-d,i} & \text{if } \boldsymbol{c}'_{1-d,i} = 1. \end{cases}$$

The latter equalities hold by Lemma 72.

Define $(\boldsymbol{\pi'}_0^{(\pm 1)}, \boldsymbol{\pi'}_0^{(\pm 1)})$ to be the proofs testifying the honest generation of $\left(\overrightarrow{\boldsymbol{W'}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W'}_1^{(\pm 1)}}\right)$ generated under I'_0 using randomness $\overrightarrow{\Pi'}$, or under I'_1 using randomness $\overrightarrow{\Theta'}$. By Lemma 73 these proofs are indeed equally generated under I'_0 and I'_1. By definition of $\mathsf{Sampl}_{\mathcal{I}}$, b is sampled uniformly at at random, so that $\Pr_I[b = d] = \Pr_I[b = 1-d]$. The group elements A'_d and A'_{1-d} satisfy the relations $P'_0 = Q * A'_0$ and $P'_1 = Q * A'_1$. We observe that the tuples $(d, A'_d, P'_0, P'_1, \pi'_0, \pi'_1)$ and $(d, A'_d, \mathsf{pk}'_{1-d})$ are the same, as are $(1-d, A'_{1-d}, P'_0, P'_1, \pi'_0, \pi'_1)$ and $(1-$

$d, A'_{1-d}, \mathsf{pk}'_d)$, and $(b, A_b, P_0, P_1, \pi_0, \pi_1)$ and $(b, A_b, \mathsf{pk}_{1-b})$ contain the same information. Furthermore, the instance generator IG is such that for any $P' \in [Q]$ and any $A' \in \mathbb{G}$ such that $P' = Q * A'$,

$$\Pr_{(P,A) \xleftarrow{\$} \mathsf{IG}(Q)} [A = A' \mid P = P'] = \frac{1}{|\mathrm{Stab}_{\mathbb{G}}(Q)|}. \tag{9}$$

Note that Q' is sampled according to $\mathsf{IG}(Q)$ as is P_b. As P_{1-b} is set to equal Q', P_0 and P_1 follow the same distribution independently. Therefore, the distribution on (P_0, P_1, π_0, π_1) is independent from b, and

$$\begin{aligned}
&\Pr_I[(b, A_b, \mathsf{pk}_{1-b}) = (d, A'_d, \mathsf{pk}'_{1-d})] \\
&\quad = \Pr_I[(b, A_b, P_0, P_1, \pi_0, \pi_1) = (d, A'_d, P'_0, P'_1, \pi'_0, \pi'_1)] \\
&\quad = \Pr_I[A_b = A'_d \mid (P_0, P_1, \pi_0, \pi_1) = (P'_0, P'_1, \pi'_0, \pi'_1) \cap (b = d)] \\
&\quad \cdot \Pr_I[(P_0, P_1, \pi_0, \pi_1) = (P'_0, P'_1, \pi'_0, \pi'_1) \cap (b = d)] \\
&\quad = \frac{1}{|\mathrm{Stab}_{\mathbb{G}}(Q)|} \cdot \Pr_I[(P_0, P_1, \pi_0, \pi_1) = (P'_0, P'_1, \pi'_0, \pi'_1)] \cdot \Pr_I[b = d] \\
&\quad = \frac{1}{|\mathrm{Stab}_{\mathbb{G}}(Q)|} \cdot \Pr_I[(P_0, P_1, \pi_0, \pi_1) = (P'_0, P'_1, \pi'_0, \pi'_1)] \cdot \Pr_I[b = 1 - d] \\
&\quad = \Pr_I[A_b = A'_{1-d} \mid (P_0, P_1, \pi_0, \pi_1) = (P'_0, P'_1, \pi'_0, \pi'_1) \cap (b = 1 - d)] \\
&\quad \cdot \Pr_I[(P_0, P_1, \pi_0, \pi_1) = (P'_0, P'_1, \pi'_0, \pi'_1) \cap (b = 1 - d)] \\
&\quad = \Pr_I[(b, A_b, P_0, P_1, \pi_0, \pi_1) = (1 - d, A'_{1-d}, P'_0, P'_1, \pi'_0, \pi'_1)] \\
&\quad = \Pr_I[(b, A_b, \mathsf{pk}_{1-b}) = (1 - d, A'_{1-d}, \mathsf{pk}'_d)].
\end{aligned}$$

From now on we abbreviate the event $(b, A_b, \mathsf{pk}_{1-b}) = (d, A'_d, \mathsf{pk}'_{1-d})$ as E_d and $(b, A_b, \mathsf{pk}_{1-b}) = (1 - d, A'_{1-d}, \mathsf{pk}'_d)$ as E_{d-1}. Note that $\overrightarrow{\boldsymbol{c}^*_{1-b}} \in (\{0,1\}^\lambda)^{k+1}$ is distributed uniformly at random and independent from the other components in I, so that $\Pr_I[\overrightarrow{\boldsymbol{c}^*_{1-b}} = \overrightarrow{\boldsymbol{c}'^*_{1-d}}] = \Pr_I[\overrightarrow{\boldsymbol{c}^*_{1-b}} = \overrightarrow{\boldsymbol{c}'^*_d}]$. We abbreviate the event $\overrightarrow{\boldsymbol{c}^*_{1-b}} = \overrightarrow{\boldsymbol{c}'^*_{1-d}}$ as F_0 and $\overrightarrow{\boldsymbol{c}^*_{1-b}} = \overrightarrow{\boldsymbol{c}'^*_d}$ as F_1. The event F_0 is independent from E_0 and F_1 is independent from E_1, so $\Pr_I[E_0 \cap F_0] = \Pr_I[E_1 \cap F_1]$.

Let $T := (\overrightarrow{\boldsymbol{H}^{(1)}_b}, \overrightarrow{\boldsymbol{H}^{(-1)}_b}, \overrightarrow{\boldsymbol{U}^{*(1)}_{1-b}}, \overrightarrow{\boldsymbol{U}^{*(-1)}_{1-b}})$, $T_0 := (\overrightarrow{\boldsymbol{H}'^{(1)}_d}, \overrightarrow{\boldsymbol{H}'^{(-1)}_d}, \overrightarrow{\boldsymbol{U}'^{*(1)}_{1-d}}, \overrightarrow{\boldsymbol{U}'^{*(-1)}_{1-d}})$ and $T_1 := (\overrightarrow{\boldsymbol{H}'^{(1)}_{1-d}}, \overrightarrow{\boldsymbol{H}'^{(-1)}_{1-d}}, \overrightarrow{\boldsymbol{U}'^{*(1)}_d}, \overrightarrow{\boldsymbol{U}'^{*(-1)}_d})$. By inspection of Sampl_I, note that the vector components of $\overrightarrow{\boldsymbol{H}^{(1)}_b}$ $\overrightarrow{\boldsymbol{H}^{(-1)}_b}$, $\overrightarrow{\boldsymbol{U}^{*(1)}_{1-b}}$ and $\overrightarrow{\boldsymbol{U}^{*(-1)}_{1-b}}$ are sampled independently from one another. Provided fixed values for $b, d, Q, P_0, P_1, \overrightarrow{\boldsymbol{c}^*_{1-b}}, \overrightarrow{\boldsymbol{c}'^*_{1-d}}$ and $\overrightarrow{\boldsymbol{c}'^*_d}$, and by equation (9), for $\beta \in \{0, 1\}$,

$$\Pr_I\left[T = T_\beta \;\middle|\; \left(\overrightarrow{\boldsymbol{W}^{(\pm1)}_0}, \overrightarrow{\boldsymbol{W}^{(\pm1)}_1}\right) = \left(\overrightarrow{\boldsymbol{W}'^{(\pm1)}_0}, \overrightarrow{\boldsymbol{W}'^{(\pm1)}_1}\right)\right] = \frac{1}{|\mathrm{Stab}_{\mathbb{G}}(Q)|^{4\lambda}}.$$

Note that the distributions on $(\overrightarrow{\boldsymbol{W}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W}_1^{(\pm 1)}})$ and T are independent from the distributions on b, (A_b, pk_{1-b}) and $\overrightarrow{\boldsymbol{c}^*_{1-b}}$, and are therefore independent from the events (E_0 and F_0) and (E_1 and F_1). Lastly, observe that

$$\begin{aligned}&\Pr_I\left[\left(\overrightarrow{\boldsymbol{W}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W}_1^{(\pm 1)}}\right) = \left(\overrightarrow{\boldsymbol{W'}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W'}_1^{(\pm 1)}}\right) \middle| T = T_0\right]\\ &= \Pr_I\left[\left(\overrightarrow{\boldsymbol{W}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W}_1^{(\pm 1)}}\right) = \left(\overrightarrow{\boldsymbol{W'}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W'}_1^{(\pm 1)}}\right) \middle| T = T_1\right] = 1,\end{aligned}$$

so $\Pr_I\left[\left(\overrightarrow{\boldsymbol{W}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W}_1^{(\pm 1)}}\right) = \left(\overrightarrow{\boldsymbol{W'}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W'}_1^{(\pm 1)}}\right) \cap (T = T_\beta)\right] = \Pr_I[T = T_\beta]$, for $\beta \in \{0, 1\}$. Hence,

$$\begin{aligned}&\Pr_I[T = T_0 \cap E_0 \cap F_0]\\ &= \Pr_I[T = T_0] \cdot \Pr_I[E_0 \cap F_0]\\ &= \Pr_I\left[\left(\overrightarrow{\boldsymbol{W}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W}_1^{(\pm 1)}}\right) = \left(\overrightarrow{\boldsymbol{W'}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W'}_1^{(\pm 1)}}\right) \cap (T = T_0)\right] \cdot \Pr_I[E_0 \cap F_0]\\ &= \Pr_I\left[\left(\overrightarrow{\boldsymbol{W}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W}_1^{(\pm 1)}}\right) = \left(\overrightarrow{\boldsymbol{W'}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W'}_1^{(\pm 1)}}\right)\right]\\ &\cdot \Pr_I\left[T = T_0 \;\middle|\; \left(\overrightarrow{\boldsymbol{W}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W}_1^{(\pm 1)}}\right) = \left(\overrightarrow{\boldsymbol{W'}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W'}_1^{(\pm 1)}}\right)\right] \cdot \Pr_I[E_0 \cap F_0].\end{aligned}$$

Replacing the expression of the probability of $T = T_0$ conditioned on the W tuple, and as $\Pr_I[E_0 \cap F_0]$ is equal to $\Pr_I[E_1 \cap F_1]$,

$$\begin{aligned}&\Pr_I[T = T_0 \cap E_0 \cap F_0]\\ &= \Pr_I\left[\left(\overrightarrow{\boldsymbol{W}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W}_1^{(\pm 1)}}\right) = \left(\overrightarrow{\boldsymbol{W'}_0^{(\pm 1)}}, \overrightarrow{\boldsymbol{W'}_1^{(\pm 1)}}\right)\right] \cdot \frac{1}{|\mathrm{Stab}_{\mathbb{G}}(Q)|^{4\lambda}} \cdot \Pr_I[E_1 \cap F_1]\\ &= \Pr_I[T = T_1 \cap E_1 \cap F_1].\end{aligned}$$

Let G_0 denote the event $(b, A_b, \mathsf{pk}_{1-b}, \overrightarrow{\boldsymbol{H}_b^{(1)}}, \overrightarrow{\boldsymbol{H}_b^{(-1)}}, \overrightarrow{\boldsymbol{c}^*_{1-b}}, \overrightarrow{\boldsymbol{U}^{*(1)}_{1-b}}, \overrightarrow{\boldsymbol{U}^{*(-1)}_{1-b}}) = (d, A'_d, \mathsf{pk}'_{1-d}, \overrightarrow{\boldsymbol{H'}_d^{(1)}}, \overrightarrow{\boldsymbol{H'}_d^{(-1)}}, \overrightarrow{\boldsymbol{c'}^*_{1-d}}, \overrightarrow{\boldsymbol{U'}^{*(1)}_{1-d}}, \overrightarrow{\boldsymbol{U'}^{*(-1)}_{1-d}})$ and G_1 the event $(b, A_b, \mathsf{pk}_{1-b}, \overrightarrow{\boldsymbol{H}_b^{(1)}}, \overrightarrow{\boldsymbol{H}_b^{(-1)}}, \overrightarrow{\boldsymbol{c}^*_{1-b}}, \overrightarrow{\boldsymbol{U}^{*(1)}_{1-b}}, \overrightarrow{\boldsymbol{U}^{*(-1)}_{1-b}}) = (1-d, A'_{1-d}, \mathsf{pk}'_d, \overrightarrow{\boldsymbol{H'}_{1-d}^{(1)}}, \overrightarrow{\boldsymbol{H'}_{1-d}^{(-1)}}, \overrightarrow{\boldsymbol{c'}^*_d}, \overrightarrow{\boldsymbol{U'}^{*(1)}_d}, \overrightarrow{\boldsymbol{U'}^{*(-1)}_d})$. G_0 is equivalent to $(T = T_0 \cap E_0 \cap F_0)$ and G_1 to $(T = T_1 \cap E_1 \cap F_1)$. So far we have shown that $\Pr_I[G_0] = \Pr_I[G_1]$. We now show that $\Pr_I[\overrightarrow{\Pi} = \overrightarrow{\Pi'} \cap G_0] = \Pr_I[\overrightarrow{\Pi} = \overrightarrow{\Theta'} \cap G_1]$ which is equivalent to

$$\Pr_{I \xleftarrow{\$} \mathsf{Sampl}_{\mathcal{I}}(Q)}[I = I_0] = \Pr_{I \xleftarrow{\$} \mathsf{Sampl}_{\mathcal{I}}(Q)}[I = I_1]$$

and thus concludes the proof.

We remark that $\Pr_I[\overrightarrow{\Pi} = \overrightarrow{\Pi'} \cap G_0] = \Pr_I[\overrightarrow{\Pi} = \overrightarrow{\Pi'} \mid G_0] \cdot \Pr_I[G_0]$ and $\Pr_I[\overrightarrow{\Pi} = \overrightarrow{\Theta'} \cap G_0] = \Pr_I[\overrightarrow{\Pi} = \overrightarrow{\Theta'} \mid G_1] \cdot \Pr_I[G_1]$ with $\Pr_I[G_0] = \Pr_I[G_1]$, thus it suffices to show that $\Pr_I[\overrightarrow{\Pi} = \overrightarrow{\Pi'} \mid G_0] = \Pr_I[\overrightarrow{\Pi} = \overrightarrow{\Theta'} \mid G_1]$

By Lemma 68 we know that in the event of G_0, $\overrightarrow{\Pi} = \overrightarrow{\Pi'}$ precisely if $(\boldsymbol{\pi}_0^{(\pm 1)}, \boldsymbol{\pi}_0^{(\pm 1)}) = (\boldsymbol{\pi'}_0^{(\pm 1)}, \boldsymbol{\pi'}_0^{(\pm 1)})$ and in the event of G_1, $\overrightarrow{\Pi} = \overrightarrow{\Pi'}$ precisely if $(\boldsymbol{\pi}_0^{(\pm 1)}, \boldsymbol{\pi}_0^{(\pm 1)}) = (\boldsymbol{\pi'}_0^{(\pm 1)}, \boldsymbol{\pi'}_0^{(\pm 1)})$. Thus, $\Pr_I[\overrightarrow{\Pi} = \overrightarrow{\Pi'} \mid G_0] = \Pr_I[\overrightarrow{\Pi} = \overrightarrow{\Theta'} \mid G_1]$ is equivalent to

$$\Pr_I\left[(\boldsymbol{\pi}_0^{(\pm 1)}, \boldsymbol{\pi}_0^{(\pm 1)}) = (\boldsymbol{\pi'}_0^{(\pm 1)}, \boldsymbol{\pi'}_0^{(\pm 1)}) \mid G_0\right] = \Pr_I\left[(\boldsymbol{\pi}_0^{(\pm 1)}, \boldsymbol{\pi}_0^{(\pm 1)}) = (\boldsymbol{\pi'}_0^{(\pm 1)}, \boldsymbol{\pi'}_0^{(\pm 1)}) \mid G_1\right] \tag{10}$$

Under the instances I'_0 and I'_1 the proofs $(\boldsymbol{\pi'}_0^{(\pm 1)}, \boldsymbol{\pi'}_0^{(\pm 1)})$ are generated using the $\Pi^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$ protocol for the same statements, but provided possibly different witnesses. By Theorem 27 the $\Pi^{\mathsf{OR}}_{Q_0,Q_1,\lambda}$ proof protocol satisfies perfect honest-verifier zero knowledge, and therefore perfect witness indistinguishability. It follows that the event $(\boldsymbol{\pi}_0^{(\pm 1)}, \boldsymbol{\pi}_0^{(\pm 1)}) = (\boldsymbol{\pi'}_0^{(\pm 1)}, \boldsymbol{\pi'}_0^{(\pm 1)})$ is statistically independent from the differences between events G_0 and G_1, so that equality 10 holds. □

$\boldsymbol{\Phi}_{\mathsf{rand},\overrightarrow{h}}$ Maintains Transcripts

Lemma 75. *Let I be an instance and fix values for* rand *and $\overrightarrow{h}$. Let $\overrightarrow{\rho_{\mathsf{Use}}}$ be the collection of first user messages of the transcript* $\mathsf{trans}(I, \mathsf{rand}, \overrightarrow{h})$ *and let $\overrightarrow{\rho'_{\mathsf{Use}}}$ be the first user messages of the transcript* $\mathsf{trans}(\Phi_{\mathsf{rand},\overrightarrow{h}}(I), \mathsf{rand}, \overrightarrow{h})$. *Then, $\overrightarrow{\rho_{\mathsf{Use}}} = \overrightarrow{\rho'_{\mathsf{Use}}}$.*

Proof. The adversary computes the first user message based on the first signer messages using randomness. By Lemma 73, the first signer messages from $\mathsf{trans}(I, \mathsf{rand}, \overrightarrow{h})$ are equal to those from $\mathsf{trans}(\Phi_{\mathsf{rand},\overrightarrow{h}}(I), \mathsf{rand}, \overrightarrow{h})$. Since the randomness for the adversary is fixed, $\overrightarrow{\rho_{\mathsf{Use}}} = \overrightarrow{\rho'_{\mathsf{Use}}}$.

□

Note that the result of Lemma 75 can also be stated as $(I, \mathsf{rand}, \overrightarrow{h})$ and $(\Phi_{\mathsf{rand},\overrightarrow{h}}(I), \mathsf{rand}, \overrightarrow{h})$ inducing the same query transcript, *i. e.*

$$\overrightarrow{e}(I, \mathsf{rand}, \overrightarrow{h}) = \overrightarrow{e}(\Phi_{\mathsf{rand},\overrightarrow{h}}(I), \mathsf{rand}, \overrightarrow{h}).$$

Lemma 76. *Let I be an instance and fix values for* rand *and $\overrightarrow{h}$. Let $\overrightarrow{\rho_{\mathsf{Sign}_2}}$ be the collection of second signer messages of the transcript* $\mathsf{trans}(I, \mathsf{rand}, \overrightarrow{h})$ *and let $\overrightarrow{\rho'_{\mathsf{Sign}_2}}$ be the second messages of the transcript* $\mathsf{trans}(\Phi_{\mathsf{rand},\overrightarrow{h}}(I), \mathsf{rand}, \overrightarrow{h})$. *Then, $\overrightarrow{\rho_{\mathsf{Sign}_2}} = \overrightarrow{\rho'_{\mathsf{Sign}_2}}$.*

Proof. Let $I_b = (b, A_b, \mathsf{pk}_{1-b}, \overrightarrow{\boldsymbol{H}_b^{(1)}}, \overrightarrow{\boldsymbol{H}_b^{(-1)}}, \overrightarrow{\boldsymbol{c}^*_{1-b}}, \overrightarrow{\boldsymbol{U}^{*(1)}_{1-b}}, \overrightarrow{\boldsymbol{U}^{*(-1)}_{1-b}}, \overrightarrow{\Pi})$ be a b-side instance. We denote the image of I_b under $\Phi_{\mathsf{rand},\overrightarrow{h}}$ by

$$\Phi_{\mathsf{rand},\overrightarrow{h}}(I_b) = (1-b, A_{1-b}, \mathsf{pk}_b, \overrightarrow{\boldsymbol{H'}^{(1)}_{1-b}}, \overrightarrow{\boldsymbol{H'}^{(-1)}_{1-b}}, \overrightarrow{\boldsymbol{c'}^*_b}, \overrightarrow{\boldsymbol{U'}^{*(1)}_b}, \overrightarrow{\boldsymbol{U'}^{*(-1)}_b}, \overrightarrow{\Theta})$$

Let $j \in [k+1]$. The second signer messages $\rho^{(j)}_{\mathsf{Sign}_2}$ and $\rho'^{(j)}_{\mathsf{Sign}_2}$ are of the form

$$\rho^{(j)}_{\mathsf{Sign}_2} = (\boldsymbol{c}^*_0, \boldsymbol{U}^{*(1)}_0, \boldsymbol{U}^{*(-1)}_0, \boldsymbol{c}^*_1, \boldsymbol{U}^{*(1)}_1, \boldsymbol{U}^{*(-1)}_1),$$
$$\rho'^{(j)}_{\mathsf{Sign}_2} = (\boldsymbol{c'}^*_0, \boldsymbol{U'}^{*(1)}_0, U^{*(-1)'}_0, \boldsymbol{c'}^*_1, \boldsymbol{U'}^{*(1)}_1, \boldsymbol{U'}^{*(-1)}_1).$$

Let $\boldsymbol{c}$ be the j-th component of user messages $\overrightarrow{\rho_{\mathsf{Use}}} = \overrightarrow{\rho'_{\mathsf{Use}}}$ which are equal by Lemma 75. Note that $\boldsymbol{c}$ is also the j-th component of the query transcript $\overrightarrow{e}(I, \mathsf{rand}, \overrightarrow{h})$. We denote $\boldsymbol{H}^{(1)}_{1-b} := \boldsymbol{H}^{(1)(j)}_{1-b}$ and $\boldsymbol{H}^{(-1)}_{1-b} := \boldsymbol{H}^{(-1)(j)}_{1-b}$.

By the definition of $\Phi_{\mathsf{rand},\overrightarrow{h}}$, $c'^*_{b,i} = c_i \cdot c^*_{1-b,i}$ for $i \in [\lambda]$. The definition of the second signer algorithm for GABS states that for all $i \in [\lambda]$,

$$c^*_{b,i} = c_i \cdot c^*_{1-b,i} \text{ and } c'^*_{1-b,i} = c_i \cdot c'^*_{b,i}.$$

Recall that $c_i \in \{-1, 1\}$, so that $c_i^2 = 1$. From the equalities, $\boldsymbol{c'}^*_0 = \boldsymbol{c}^*_0$ and $\boldsymbol{c'}^*_1 = \boldsymbol{c}^*_1$. One now needs to show that $\boldsymbol{U'}^{*(1)}_b = \boldsymbol{U}^{*(1)}_b$. From the definition of the second signer algorithm for GABS, for $i \in [\lambda]$,

$$U^{*(1)}_{b,i} = \begin{cases} A_b^{-1} \cdot H^{(1)}_{b,i} & \text{if } c^*_{b,i} = -1 \\ H^{(1)}_{b,i} & \text{if } c^*_{b,i} = 1 \end{cases}.$$

From the definition of $\Phi_{\mathsf{rand},\overrightarrow{h}}$,

$$U'^*_{b,i} = \begin{cases} A_b^{-1} \cdot H^{(1)}_{b,i} & \text{if } c^*_{b,i} = -1 \\ H^{(1)}_{b,i} & \text{if } c^*_{b,i} = 1, \end{cases}$$

so $\boldsymbol{U'}^{*(1)}_b = \boldsymbol{U}^{*(1)}_b$. A similar computation shows that $\boldsymbol{U'}^{*(-1)}_b = \boldsymbol{U}^{*(-1)}_b$.

Lastly, we show that $\boldsymbol{U'}^{*(1)}_{1-b} = \boldsymbol{U}^{*(1)}_{1-b}$. From the definition of $\Phi_{\mathsf{rand},\overrightarrow{h}}$,

$$H'^{(1)}_{1-b,i} = \begin{cases} A_{1-b} \cdot U^{*(1)}_{1-b,i} & \text{if } c^*_{1-b,i} = -1 \\ U^{*(1)}_{1-b,i} & \text{if } c^*_{1-b,i} = 1. \end{cases}$$

Recall that $\boldsymbol{c'}^*_{1-b} = \boldsymbol{c}^*_{1-b}$. From the definition of the second signer algorithm for GABS, for i in $[\lambda]$,

$$\begin{aligned} U'^*_{1-b,i} &= \begin{cases} A^{-1}_{1-b} \cdot H'^{(1)}_{1-b,i} & \text{if } c'_{1-b,i} = -1 \\ H'^{(1)}_{1-b,i} & \text{if } c'_{1-b,i} = 1 \end{cases} \\ &= \begin{cases} A^{-1}_{1-b} \cdot A_{1-b} \cdot U^{*(1)}_{1-b,i} & \text{if } c_{1-b,i} = -1 \\ U^{*(1)}_{1-b,i} & \text{if } c_{1-b,i} = 1 \end{cases} \\ &= U^{*(1)}_{1-b,i}, \end{aligned}$$

so $\boldsymbol{U}'^{*(1)}_{1-b} = \boldsymbol{U}^{*(1)}_{1-b}$. Similarly, $\boldsymbol{U}^{*(-1)'}_{1-b} = \boldsymbol{U}^{*(-1)}_{1-b}$. Finally, $\overrightarrow{\rho_{\mathsf{Sign}_2}} = \overrightarrow{\rho'_{\mathsf{Sign}_2}}$.

□

Lemma 77. *Let I be an instance and fix values for* rand *and $\overrightarrow{h}$. Let $\overrightarrow{\sigma}$ be the collection of signatures in the transcript* $\mathsf{trans}(I, \mathsf{rand}, \overrightarrow{h})$ *and $\overrightarrow{\sigma'}$ the signatures from the transcript* $\mathsf{trans}(\Phi_{\mathsf{rand},\overrightarrow{h}}(I), \mathsf{rand}, \overrightarrow{h})$*. Then, $\overrightarrow{\sigma} = \overrightarrow{\sigma'}$.*

Proof. The adversary computes the forged signatures based on the signer messages using randomness. By Lemma 73 and Lemma 76, the signer messages from both transcripts are equal. Since the randomness for the adversary is fixed, $\overrightarrow{\sigma} = \overrightarrow{\sigma'}$.

□

$\Phi_{\mathsf{rand},\overrightarrow{h}}$ is a Self-inverse Bijection

Lemma 78. *The map $\Phi_{\mathsf{rand},\overrightarrow{h}}$ respects transcripts,* i. e. *for fixed* rand *and $\overrightarrow{h}$, and I an instance such that $(I, \mathsf{rand}, \overrightarrow{h}) \in \mathsf{Succ}$,*

$$\mathsf{trans}(I, \mathsf{rand}, \overrightarrow{h}) = \mathsf{trans}(\Phi_{\mathsf{rand},\overrightarrow{h}}(I), \mathsf{rand}, \overrightarrow{h}).$$

Proof. A transcript consists of signer messages, a user message and a signature. By Lemma 73, Lemma 75, Lemma 76 and Lemma 77, all of these components are equal in $\mathsf{trans}(I, \mathsf{rand}, \overrightarrow{h})$ and $\mathsf{trans}(\Phi_{\mathsf{rand},\overrightarrow{h}}(I), \mathsf{rand}, \overrightarrow{h})$.

□

Lemma 79. *Let* rand *and $\overrightarrow{h}$ be fixed. For instances I such that $(I, \mathsf{rand}, \overrightarrow{h}) \in \mathsf{Succ}$, $\Phi_{\mathsf{rand},\overrightarrow{h}}$ is a self inverse bijection.*

Proof. Let $(I, \mathsf{rand}, \overrightarrow{h}) \in \mathsf{Succ}$.

The instance I is a b-side instance for some $b \in \{0, 1\}$, and may be denoted as

$$I = (b, A_b, \mathsf{pk}_{1-b} = (P_{1-b}, \pi_0, \pi_1), \overrightarrow{\boldsymbol{H}^{(1)}_b}, \overrightarrow{\boldsymbol{H}^{(-1)}_b}, \overrightarrow{\boldsymbol{c}^*_{1-b}}, \overrightarrow{\boldsymbol{U}^{*(1)}_{1-b}}, \overrightarrow{\boldsymbol{U}^{*(-1)}_{1-b}}, \overrightarrow{\Pi}).$$

Furthermore, denote

$$\Phi_{\mathsf{rand},\overrightarrow{h}}(I) = (1-b, A_{1-b}, \mathsf{pk}_b = (P_b, \pi_0, \pi_1), \overrightarrow{\boldsymbol{H}^{(1)}_{1-b}}, \overrightarrow{\boldsymbol{H}^{(-1)}_{1-b}}, \overrightarrow{\boldsymbol{c}^*_b}, \overrightarrow{\boldsymbol{U}^{*(1)}_b}, \overrightarrow{\boldsymbol{U}^{*(-1)}_b}, \overrightarrow{\Theta})$$

and

$$\Phi_{\mathsf{rand},\overrightarrow{h}}(\Phi_{\mathsf{rand},\overrightarrow{h}}(I)) = (b', A'_b, \mathsf{pk}'_{1-b} = (P'_{1-b}, \pi'_0, \pi'_1), \overrightarrow{\boldsymbol{G}^{(1)}_b}, \overrightarrow{\boldsymbol{G}^{(-1)}_b}, \overrightarrow{\boldsymbol{d}^*_{b-1}}, \overrightarrow{\boldsymbol{V}^{*(1)}_{1-b}}, \overrightarrow{\boldsymbol{V}^{*(-1)}_{1-b}}, \overrightarrow{\Pi'}).$$

One hence has $b' = 1 - (1 - b) = b$, and

$$(b, A'_b) = \phi_{\mathsf{pk},1-b}(1-b, A_{1-b}) = \phi_{\mathsf{pk},1-b}(\phi_{\mathsf{pk},b}(b, A_b)) = (b, A_b),$$

so $A'_b = A_b$. Furthermore, from the definition of P'_{1-b}, $P'_{1-b} = Q * A_{1-b} = P_{1-b}$, and from the one of $\Phi_{\mathsf{rand},\overrightarrow{h}}$, $\pi'_0 = \pi_0$ and $\pi'_1 = \pi_1$. Hence, $\mathsf{pk}'_{1-b} = \mathsf{pk}_{1-b}$.

For $j \in [k+1]$ and $i \in [\lambda]$,

$$G_{b,i}^{(1)(j)} = \begin{cases} A_b^{-1} \cdot U_{b,i}^{*(1)(j)} & \text{if } {c_{b,i}^*}^{(j)} = -1 \\ U_{b,i}^{*(1)(j)} & \text{if } {c_{b,i}^*}^{(j)} = 1 \end{cases}$$

and

$$U_{b,i}^{*(1)(j)} = \begin{cases} A'_b \cdot H_{b,i}^{(1)(j)} & \text{if } {c_{b,i}^*}^{(j)} = -1 \\ H_{b,i}^{(1)(j)} & \text{if } {c_{b,i}^*}^{(j)} = 1. \end{cases}$$

As $A'_b = A_b$, one has $G_{b,i}^{(1)(j)} = H_{b,i}^{(1)(j)}$. Similarly, $G_{b,i}^{(-1)(j)} = H_{b,i}^{(-1)(j)}$. Additionally,

$${d_{1-b,i}^*}^{(j)} = d_i^{*(j)} \cdot {c_{b,i}^*}^{(j)} \quad \text{and} \quad {c_{b,i}^*}^{(j)} = c_i^{*(j)} \cdot {c_{1-b,i}^*}^{(j)}$$

where $\overrightarrow{\boldsymbol{d}^*} = \overrightarrow{e}(\Phi_{\mathsf{rand},\overrightarrow{h}}(I), \mathsf{rand}, \overrightarrow{h})$ and $\overrightarrow{\boldsymbol{c}^*} = \overrightarrow{e}(I, \mathsf{rand}, \overrightarrow{h})$. As $\Phi_{\mathsf{rand},\overrightarrow{h}}$ respects transcripts for successful tuples $(I, \mathsf{rand}, \overrightarrow{h}) \in \mathsf{Succ}$, $\overrightarrow{d_i^*} = \overrightarrow{c_i^*}$ and thus ${d_{1-b,i}^*}^{(j)} = {c_{1-b,i}^*}^{(j)}$.

$$V_{1-b,i}^{*(1)\,(j)} = \begin{cases} A_{1-b}^{-1} \cdot H_{1-b,i}^{(1)(j)} & \text{if } {d_{1-b,i}^*}^{(j)} = -1 \\ H_{1-b,i}^{(1)(j)} & \text{if } {d_{1-b,i}^*}^{(j)} = 1 \end{cases}$$

and

$$H_{1-b,i}^{(1)(j)} := \begin{cases} A_{1-b} \cdot U_{1-b,i}^{*(1)\,(j)} & \text{if } {c_{1-b,i}^*}^{(j)} = -1 \\ U_{1-b,i}^{*(1)\,(j)} & \text{if } {c_{1-b,i}^*}^{(j)} = 1 \end{cases}.$$

We have already established that ${d_{1-b,i}^*}^{(j)} = {c_{1-b,i}^*}^{(j)}$, so $V_{1-b,i}^{*(1)\,(j)} = U_{1-b,i}^{*(1)\,(j)}$. Similarly, $V_{1-b,i}^{*(-1)\,(j)} = U_{1-b,i}^{*(-1)\,(j)}$.

We know that I and $\Phi_{\mathsf{rand},\overrightarrow{h}}(\Phi_{\mathsf{rand},\overrightarrow{h}}(I))$ are equal in all components except possibly $\overrightarrow{\Pi}$ and $\overrightarrow{\Pi'}$ and both induce the same first signer messages under rand and $\overrightarrow{h}$ by Lemma 73. From the definition of $\Phi_{\mathsf{rand},\overrightarrow{h}}$, $\overrightarrow{\Theta}$ and $\overrightarrow{\Pi'}$ are defined uniquely by the fact that they maintain the Zero-Knowledge proofs in the first signer messages, so $\overrightarrow{\Pi} = \overrightarrow{\Pi'}$. Finally,

$$\begin{aligned} \Phi_{\mathsf{rand},\overrightarrow{h}}(\Phi_{\mathsf{rand},\overrightarrow{h}}(I)) &= (b', A'_b, \mathsf{pk}'_{1-b}, \overrightarrow{\boldsymbol{G}_b^{(1)}}, \overrightarrow{\boldsymbol{G}_b^{(-1)}}, \overrightarrow{\boldsymbol{d}_{b-1}^*}, \overrightarrow{\boldsymbol{V}_{1-b}^{*(1)}}, \overrightarrow{\boldsymbol{V}_{1-b}^{*(-1)}}, \overrightarrow{\Pi'}) \\ &= (b, A_b, \mathsf{pk}_{1-b}, \overrightarrow{\boldsymbol{H}_b^{(1)}}, \overrightarrow{\boldsymbol{H}_b^{(-1)}}, \overrightarrow{\boldsymbol{c}_{1-b}^*}, \overrightarrow{\boldsymbol{U}_{1-b}^{*(1)}}, \overrightarrow{\boldsymbol{U}_{1-b}^{*(-1)}}, \overrightarrow{\Pi}) \\ &= I, \end{aligned}$$

hence $\Phi_{\mathsf{rand},\overrightarrow{h}}$ is a bijection and self-inverse.

□

Lemma 80. *Lemma 47 is satisfied for* GABS.

Proof. We have provided a definition for $\Phi_{\mathsf{rand},\overrightarrow{h}}$ and by Lemma 78 and Lemma 79 it is a required for Lemma 47 to hold.

□

Witness extractors for GABS.

Definition 81 (Explicit witness extractors $(\mathsf{Ext}_0, \mathsf{Ext}_1)$**).** *Let I be an instance and* rand *some randomness, $(\overrightarrow{h}, \overrightarrow{h'}) \in F_j(I, \mathsf{rand})$ for some $j \in [k+1]$, and $\sigma = (\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{U}_0, \boldsymbol{U}_1)$ and $\sigma' = (\boldsymbol{c}'_0, \boldsymbol{c}'_1, \boldsymbol{U}'_0, \boldsymbol{U}'_1)$ denote the signatures that correspond to the j-th entries of $\overrightarrow{h}$, respectively $\overrightarrow{h'}$, denoted by $\boldsymbol{c}^{(j)}$ and $\boldsymbol{c'}^{(j)}$. For all $i \in [\lambda]$, $c_{0,i} \cdot c_{1,i} = c_i^{(j)}$ and $c'_{0,i} \cdot c'_{1,i} = c_i'^{(j)}$. We adapt a definition for witness extractors $(\mathsf{Ext}_0, \mathsf{Ext}_1)$ from Figure 7. For $b \in \{0,1\}$ let Ext_b be as in Figure 7, but if the output value A is unequal to $\perp$, then Ext_b outputs (b, A) instead of A.*

Lemma 82. *The witness extractors from Definition 81 satisfy the definition of witness extractors in Definition 46.*

Proof. We consider fixed values for an instance I and randomness rand. Let $(\overrightarrow{h}, \overrightarrow{h'}) \in F_j(I, \mathsf{rand})$ be a successful fork at index $j \in [k+1]$. Let $\sigma = (\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{U}_0, \boldsymbol{U}_1)$ and $\sigma' = (\boldsymbol{c}'_0, \boldsymbol{c}'_1, \boldsymbol{U}'_0, \boldsymbol{U}'_1)$ be as in Definition 81.

For $b \in \{0,1\}$ and $i \in [\lambda]$ set

$$Y_{b,i} = \begin{cases} P_b * U_{b,i} & \text{if } c_{b,i} = -1 \\ Q * U_{b,i} & \text{if } c_{b,i} = 1 \end{cases} \quad \text{and} \quad Y'_{b,i} = \begin{cases} P_b * U'_{b,i} & \text{if } c'_{b,i} = -1 \\ Q * U'_{b,i} & \text{if } c'_{b,i} = 1 \end{cases}.$$

Since both signatures are accepting, $c_i^{(j)} = c_{0,i} \cdot c_{1,i}$ and $c'^{(j)}_i = c'_{0,i} \cdot c'_{1,i}$ for all $i \in [\lambda]$, and for

$$\boldsymbol{c}^{(j)} = \mathcal{H}(\boldsymbol{Y}_0 || \boldsymbol{Y}_1 || m) \quad \text{and} \quad \boldsymbol{c'}^{(j)} = \mathcal{H}(\boldsymbol{Y}'_0 || \boldsymbol{Y}'_1 || m'),$$

the vectors $\overrightarrow{h}$ and $\overrightarrow{h'}$ agree up to the j-th entry and the randomness of the challenger and the adversary is fixed, so the inputs of the j-the queries must agree. Hence, $m = m'$, $\boldsymbol{Y}_0 = \boldsymbol{Y}'_0$ and $\boldsymbol{Y}_1 = \boldsymbol{Y}'_1$.

Let $b \in \{0,1\}$. If there is no t in $[\lambda]$ such that $c_{b,t} \neq c'_{b,t}$, $\mathsf{Ext}_b(\sigma, \sigma')$ outputs $\perp$. If there exists a value t in $[\lambda]$ such that $c_{b,t} \neq c'_{b,t}$, then from Lemma 67, $\mathsf{Ext}_b(\sigma, \sigma')$ outputs a b-side witness.

As $\boldsymbol{c}^{(j)} \neq \boldsymbol{c'}^{(j)}$, and for $i \in [\lambda]$, $c_{0,i} \cdot c_{1,i} = c_i^{(j)}$ and $c'_{0,i} \cdot c'_{1,i} = c_i'^{(j)}$, there exists an index $t \in [\lambda]$ such that $c_{0,t} \neq c'_{0,t}$ or such that $c_{1,t} \neq c'_{1,t}$. Therefore, $\mathsf{Ext}_0(\sigma, \sigma')$ or $\mathsf{Ext}_1(\sigma, \sigma')$ must output a valid secret key and

$$(\mathsf{Ext}_0(\sigma, \sigma'), \mathsf{Ext}_1(\sigma, \sigma')) \in (\mathcal{S}_{\mathsf{pk},0} \times \{\perp\}) \cup (\{\perp\} \times \mathcal{S}_{\mathsf{pk},1}) \cup (\mathcal{S}_{\mathsf{pk},0} \times \mathcal{S}_{\mathsf{pk},1}).$$

We conclude that $(\mathsf{Ext}_0, \mathsf{Ext}_1)$ are witness extractors as in Definition 46.

□

Lemma 83. *Lemma 48 is true for the witness extractors* $(\mathsf{Ext}_0, \mathsf{Ext}_1)$ *from definition 81.*

Proof. This demonstration follows the one of [KLLQ23, Lemma 4.5]. Consider a value $b \in \{0, 1\}$. The proof of Lemma 82 showed that a b-side witness can be extracted from two successful tuples $(I, \mathsf{rand}, \overrightarrow{h})$ and $(I, \mathsf{rand}, \overrightarrow{h'})$ precisely when $\boldsymbol{c}_b \neq \boldsymbol{c}'_b$, where $\sigma = (\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{U}_0, \boldsymbol{U}_1)$ and $\sigma' = (\boldsymbol{c}'_0, \boldsymbol{c}'_1, \boldsymbol{U}'_0, \boldsymbol{U}'_1)$ are the signatures corresponding to h_j and h'_j, respectively. We proceed by contradiction. Let

$$((I, \mathsf{rand}, \overrightarrow{h}), (I, \mathsf{rand}, \overrightarrow{h'}), (I, \mathsf{rand}, \overrightarrow{h''})) \in \triangle_j(I, \mathsf{rand})$$

be a triangle at index j with respect to I and rand, and $\sigma = (\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{U}_0, \boldsymbol{U}_1)$, $\sigma' = (\boldsymbol{c}'_0, \boldsymbol{c}'_1, \boldsymbol{U}'_0, \boldsymbol{U}'_1)$ and $\sigma'' = (\boldsymbol{c}''_0, \boldsymbol{c}''_1, \boldsymbol{U}''_0, \boldsymbol{U}''_1)$ denote the signatures corresponding to h_j, h'_j and h''_j, respectively. Suppose that a b-side witness can be extracted at index j from $((I, \mathsf{rand}, \overrightarrow{h}), (I, \mathsf{rand}, \overrightarrow{h'}))$ while it cannot be extracted from $((I, \mathsf{rand}, \overrightarrow{h}), (I, \mathsf{rand}\overrightarrow{h''}))$ nor $(I, \mathsf{rand}, \overrightarrow{h'}, (I, \mathsf{rand}, \overrightarrow{h''}))$. Hence $\boldsymbol{c}_b = \boldsymbol{c}''_b$ and $\boldsymbol{c}'_b = \boldsymbol{c}''_b$, so $\boldsymbol{c}_b = \boldsymbol{c}'_b$, which implies that a b-side witness cannot be extracted from $((I, \mathsf{rand}, \overrightarrow{h}), (I, \mathsf{rand}, \overrightarrow{h'}))$ at index j, resulting in a contradiction.

□

Theorem 84 (k-One-More Unforgeability). *If the* $\mathrm{GAIP}_{\mathsf{IG}}$ *is hard, then the* GABS *blind signature protocol satisfies k-one-more unforgeability for any k in $\mathbb{N}$ that is logarithmic in the security parameter. More precisely, there exist universal constants C_1, C_2 in $\mathbb{R}_+$ such that for k, λ in $\mathbb{N}$, for any PPT adversary $\mathcal{A}$ making $Q_{\mathcal{H}}$ hash queries to a random oracle, if* $\mathsf{Adv}^{k-\mathsf{OMUF}}_{\mathcal{A}}(\lambda) \geq C_1 \binom{Q_{\mathcal{H}}}{k+1} \cdot 2^{-\lambda}$*, then*

$$\mathsf{Adv}^{k-\mathsf{OMUF}}_{\mathcal{A}}(\lambda) \leq \sqrt{\binom{Q_{\mathcal{H}}}{k+1}^2 \frac{(k+1)^3}{C_2} \mathsf{Adv}^{\mathrm{GAIP}_{\mathsf{IG}}}_{\mathcal{A}}(\lambda)}.$$

Proof. By Lemma 71, the GABS construction satisfies the conditions in Definition 38. By Lemma 80, $\Phi_{\mathsf{rand}, \overrightarrow{h}}$ is as is required for Lemma 47 to hold. By Definition 81 we obtain witness extractors as in Definition 46 for which Lemma 48 is satisfied. Thus, the GABS blind signature scheme satisfies all required conditions to apply Theorem 49 and we obtain that it satisfies one-more unforgeability as in Definition 17.

□

I Proofs for the GASDVS scheme

In this section we discuss the proofs for the security of the GASDVS construction.

I.1 Correctness of GASDVS.

Theorem 85 (Correctness of GASDVS). *The* GASDVS *scheme for strong designated verifier signatures satisfies correctness;* i.e., *there exists a negligible function* $\mathsf{negl} : \mathbb{N} \to \mathbb{R}_+$ *such that for any* λ *in* $\mathbb{N}$, $(\mathsf{pk}_s, \mathsf{sk}_s) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, s)$, $(\mathsf{pk}_v, \mathsf{sk}_v) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, v)$, *and message* $m \in \{0,1\}^*$,

$$\Pr\left[\mathsf{Verify}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m, \sigma, E) = 1 \;\middle|\; (\sigma, E) \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}_s, \mathsf{pk}_s, \mathsf{pk}_v, m)\right] = 1 - \mathsf{negl}(\lambda).$$

Proof. Let $\lambda \in \mathbb{N}$ and let $(\mathsf{pk}_s, \mathsf{sk}_s) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, s)$ and let $(\mathsf{pk}_v, \mathsf{sk}_v) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, v)$. Parse the keys as $P_s \leftarrow \mathsf{pk}_s$, $A_s \leftarrow \mathsf{sk}_s$, $(P_v, \mathsf{pk}_\mathsf{K}) \leftarrow \mathsf{pk}_v$ and $(A_v, \mathsf{sk}_\mathsf{K}) \leftarrow \mathsf{sk}_v$. Let $m \in \{0,1\}^*$ and let $(\sigma, E) \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}_s, \mathsf{pk}_s, \mathsf{pk}_v, m)$. We show that

$$\mathsf{Verify}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m, \sigma, E) = 1,$$

that is, we show that $\boldsymbol{Y}'_s = \boldsymbol{Y}_s$, $\boldsymbol{Y}'_v = \boldsymbol{Y}_v$ and that for all $i \in [\lambda]$, $e_i = c_{s,i} \cdot c_{v,i}$, where $K \leftarrow \mathsf{Decaps}(\mathsf{sk}_\mathsf{K}, c)$, σ is parsed as $(\boldsymbol{c}_s, \boldsymbol{c}_v, V, \boldsymbol{Y}_s, \boldsymbol{Y}_v) \leftarrow \sigma$, $(\boldsymbol{U}_s, \boldsymbol{U}_v) \leftarrow \mathsf{Dec}(K, V)$, $\boldsymbol{Y}'_s$ and $\boldsymbol{Y}'_v$ are computed as

$$Y'_{s,i} = \begin{cases} P_s \cdot U_{s,i} & \text{if } c_{s,i} = -1 \\ Q \cdot U_{s,i} & \text{if } c_{s,i} = 1 \end{cases} \quad \text{and} \quad Y'_{v,i} = \begin{cases} P_v \cdot U_{v,i} & \text{if } c_{v,i} = -1 \\ Q \cdot U_{v,i} & \text{if } c_{v,i} = 1 \end{cases}.$$

and $\boldsymbol{e} = \mathcal{H}(m||K||E||\boldsymbol{Y}_s||\boldsymbol{Y}_v)$.

Let $\epsilon^{\mathsf{Fail}}_{\mathsf{IG}}$ denote the probability, that the instance generator IG outputs $\perp$. Note that $\epsilon^{\mathsf{Fail}}_{\mathsf{IG}}$ is negligible in λ. Let $\epsilon^{\mathsf{Fail}}_{\mathsf{KEM}}$ denote the probability that a KEM encapsulation does not decapsulate correctly and let $\epsilon^{\mathsf{Fail}}_{\mathsf{SKE}}$ denote the probability that an SKE encryption does not decrypt correctly. Both $\epsilon^{\mathsf{Fail}}_{\mathsf{KEM}}$ and $\epsilon^{\mathsf{Fail}}_{\mathsf{SKE}}$ are negligible in λ by the correctness of KEM and SKE.

If the decapsulation under KEM and the decryption under SKE are successful, then the value of $(\boldsymbol{U}_s, \boldsymbol{U}_v)$ that we obtain is equal to the value of $(\boldsymbol{U}_s, \boldsymbol{U}_v)$ as specified in the description of Sign except with probability bounded by $\epsilon^{\mathsf{Fail}}_{\mathsf{KEM}} + \epsilon^{\mathsf{Fail}}_{\mathsf{SKE}}$. If the value of $(\boldsymbol{U}_s, \boldsymbol{U}_v)$ is indeed as such, we are able to show that $\mathsf{Verify}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m, \sigma, c) = 1$, assuming that all calls to IG are successful.

Consider the computation of $\boldsymbol{Y}_s$, $\boldsymbol{Y}_v$, $\boldsymbol{c}_s$, $\boldsymbol{c}_v$, $\boldsymbol{U}_s$, $\boldsymbol{U}_v$ and $\boldsymbol{W}_s$ as specified in the GASDVS protocol for honest generation of (σ, E). We first show that

$$\boldsymbol{Y}'_s = \boldsymbol{Y}_s \quad \text{and} \quad \boldsymbol{Y}'_v = \boldsymbol{Y}_v. \tag{11}$$

Observe that for all $i \in [\lambda]$, $(Y_{s,i}, W_{s,i})$ and $(Y_{v,i}, U_{v,i})$ are outputs of $\mathsf{IG}(Q)$ or $\mathsf{IG}(\mathsf{pk}_v)$. In particular it holds that for all $i \in [\lambda]$,

$$Y_{s,i} = Q * W_{s,i} \quad \text{and} \quad Y_{v,i} = \begin{cases} P_v \cdot U_{v,i} & \text{if } c_{v,i} = -1 \\ Q \cdot U_{v,i} & \text{if } c_{v,i} = 1 \end{cases}.$$

Furthermore, for all $i \in [\lambda]$, $U_{s,i}$ is computed as

$$U_{s,i} = \begin{cases} A_s^{-1} \cdot W_{s,i} & \text{if } c_{s,i} = -1 \\ W_{s,i} & \text{if } c_{s,i} = 1 \end{cases},$$

so we also have that

$$\begin{aligned} Y_{s,i} = Q * W_{s,i} &= \begin{cases} Q * A_s \cdot A_s^{-1} \cdot W_{s,i} & \text{if } c_{s,i} = -1 \\ Q \cdot W_{s,i} & \text{if } c_{s,i} = 1 \end{cases} \\ &= \begin{cases} P_s \cdot U_{s,i} & \text{if } c_{s,i} = -1 \\ Q \cdot U_{s,i} & \text{if } c_{s,i} = 1 \end{cases}. \end{aligned}$$

It follows that the equations in (11) are satisfied.

Observe that $\boldsymbol{c}_s$ is computed based on $\boldsymbol{c}_v$ and $\boldsymbol{e} = \mathcal{H}(m||K||E||\boldsymbol{Y}_s||\boldsymbol{Y}_v)$ such that precisely $e_i = c_{s,i} \cdot c_{v,i}$ for all $i \in [\lambda]$. We conclude that $\mathsf{Verify}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m, \sigma, c) = 1$.

The generation of a GASDVS signature requires 2λ calls to IG, of which we need to consider the failure probability. It follows that

$$\begin{aligned} &\Pr\left[\mathsf{Verify}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m, \sigma, E) = 1 \,\middle|\, (\sigma, E) \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}_s, \mathsf{pk}_s, \mathsf{pk}_v, m)\right] \\ &\geq 1 - (2\lambda \cdot \epsilon_{\mathsf{IG}}^{\mathsf{Fail}} + \epsilon_{\mathsf{KEM}}^{\mathsf{Fail}} + \epsilon_{\mathsf{SKE}}^{\mathsf{Fail}}). \end{aligned}$$

□

I.2 Non-Transferability of GASDVS.

Theorem 86 (Non-Transferability of GASDVS). *The GASDVS scheme satisfies perfect non-transferability.*

Proof. Let $Q \in \mathcal{Q}$ be the same fixed value is in the GASDVS description and let $\lambda \in \mathbb{N}$ be the security parameter. We define the following PPT algorithm Sim:

$\mathsf{Sim}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m) \to (\sigma, E)$**:** The algorithm

1. parses $(A_v, \mathsf{sk}_\mathsf{K}) \leftarrow \mathsf{sk}_v$, $P_s \leftarrow \mathsf{pk}_s$, $(P_v, \mathsf{pk}_\mathsf{K}) \leftarrow \mathsf{pk}_v$;
2. sets $x = (P_s, P_v)$ and $w = (1, A_v)$,
   $(\mathsf{st}, \rho = (\boldsymbol{Y}_s, \boldsymbol{Y}_v)) \xleftarrow{\$} \Sigma_{Q,Q,\lambda}^{\mathsf{OR}}.\mathsf{Commit}(x, w)$,
   $(K, E) \xleftarrow{\$} \mathsf{KEM.Encaps}(\mathsf{pk}_\mathsf{K})$ and $\boldsymbol{e} \leftarrow \mathcal{H}(m\|K\|E\|\boldsymbol{Y}_s\|\boldsymbol{Y}_v)$;
3. computes $(\boldsymbol{c}_s, \boldsymbol{c}_v, \boldsymbol{U}_s, \boldsymbol{U}_v) \xleftarrow{\$} \Sigma_{Q,Q,\lambda}^{\mathsf{OR}}.\mathsf{Prove}(x, w, \mathsf{st}, \boldsymbol{e})$
4. sets $V \xleftarrow{\$} \mathsf{SKE.Enc}(K, (\boldsymbol{U}_s, \boldsymbol{U}_v))$, $\sigma \leftarrow (\boldsymbol{c}_s, \boldsymbol{c}_v, V, \boldsymbol{Y}_s, \boldsymbol{Y}_v)$;
5. returns (σ, E).

Let m be in $\{0,1\}^*$, and let $(\mathsf{pk}_s, \mathsf{sk}_s) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, s)$ and $(\mathsf{pk}_v, \mathsf{sk}_v) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, v)$, where we parse $P_s \leftarrow \mathsf{pk}_s$, $A_s \leftarrow \mathsf{sk}_s$, $(P_v, \mathsf{sk}_\mathsf{K}) \leftarrow \mathsf{pk}_v$ and

$(A_v, \mathsf{sk}_\mathsf{K}) \leftarrow \mathsf{sk}_v$. We show that the output distributions of $\mathsf{Sim}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m)$ and $\mathsf{Sign}(\mathsf{sk}_s, \mathsf{pk}_s, \mathsf{pk}_v, m)$ are equal.

Observe that the only difference between the algorithms Sign and Sim is in the value of the witness w for the statement $x = (P_s, P_v)$. For Sign, w takes the value $(0, A_s)$ and for Sim, w takes the value $(1, A_v)$.

The algorithms Sign and Sim both function by generating a non-interactive proof $\pi = (\boldsymbol{c}_s, \boldsymbol{c}_v, \boldsymbol{U}_s, \boldsymbol{U}_v)$ based on the $\Sigma^{\mathsf{OR}}_{Q,Q,\lambda}$ protocol for the statement witness pair (x, w). Additionally, the value of $(\boldsymbol{U}_s, \boldsymbol{U}_v)$ is encrypted and the values of $\boldsymbol{Y}_s$ and $\boldsymbol{Y}_v$, which can be directly computed from $\boldsymbol{c}_s$, $\boldsymbol{c}_v$, $\boldsymbol{U}_s$ and $\boldsymbol{U}_v$ are added to the signature. The $\Sigma^{\mathsf{OR}}_{Q,Q,\lambda}$ proof protocol satisfies perfect honest verifier zero knowledge (Theorem 27) and is therefore perfectly witness indistinguishable. It follows that the proofs generated internally by $\mathsf{Sim}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m)$ and $\mathsf{Sign}(\mathsf{sk}_s, \mathsf{pk}_s, \mathsf{pk}_v, m)$ are statistically indistinguishable. Therefore, the σ components of the signatures they generate, which are such proofs with encrypted $(\boldsymbol{U}_s, \boldsymbol{U}_v)$ values and added $(\boldsymbol{Y}_s, \boldsymbol{Y}_v)$ values, are also statistically indistinguishable.

The keys for encryption under SKE in Sign and Sim are both generated as $(K, E) \xleftarrow{\$} \mathsf{KEM.Encaps}(\mathsf{pk}_\mathsf{K})$, so the values for E and E' in $(\sigma, E) \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}_s, \mathsf{pk}_s, \mathsf{pk}_v, m)$ and $(\sigma', E') \xleftarrow{\$} \mathsf{Sim}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m)$ are also equally distributed. Therefore, the output distributions of $\mathsf{Sim}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m)$ and $\mathsf{Sign}(\mathsf{sk}_s, \mathsf{pk}_s, \mathsf{pk}_v, m)$ are equal.

□

I.3 Strong Unforgeability of GASDVS.

Theorem 87 (Strong Unforgeability of GASDVS). *If the hardness of the* $\mathrm{GAIP}_\mathsf{IG}$ *holds, then the* GASDVS *protocol satisfies strong unforgeability in the random oracle model; more precisely, given a PPT adversary* $\mathcal{A}$ *making* q *random oracle queries and security parameter* λ *in* $\mathbb{N}$*,*

$$\mathsf{Adv}^{\mathsf{SUF}}_{\mathcal{A}}(\lambda) \leq \sqrt{2q \cdot \mathsf{Adv}^{\mathrm{GAIP}_\mathsf{IG}}_{\mathcal{A}}(\lambda)}.$$

Proof. Adapting the proof strategy of [MTT25] to our KEM-based construction, given a PPT algorithm $\mathcal{A}$ that would have a non-negligible advantage $\mathsf{Adv}^{\mathsf{SUF}}_{\mathcal{A}}$ in the strong unforgeability game for GASDVS, the reduction PPT $\mathcal{R}$ described in this proof will show that it may be used to build an efficient adversary with non-negligible advantage against the $\mathrm{GAIP}_\mathsf{IG}$. For λ a security parameter in $\mathbb{N}$, let $* : \mathcal{Q} * \mathbb{G} \to \mathbb{Q}$ be the corresponding group action for IG and let $Q \in \mathcal{Q}$ be fixed. We consider the corresponding NP relation $\mathcal{R}_{Q,\lambda}$ (Definition 23).

Construction of a Simulating Algorithm $\mathcal{W}$. To use the forking Lemma 59 (Appendix F), the ordinary challenger for the strong unforgeability game will be replaced by another PPT routine $\mathcal{W}$, in an indistinguishable manner for $\mathcal{A}$, without using either the signer or the verifier's secret key. This algorithm will take as input a tuple $(x, \boldsymbol{e}_1, ..., \boldsymbol{e}_q)$, where q is the total number of queries that

$\mathcal{A}$ makes to the oracles $\mathcal{O}^{\mathsf{Sign}}$, $\mathcal{O}^{\mathsf{Verify}}$ or $\mathcal{O}^{\mathcal{H}}$, and $x \in \mathcal{Q}$ is a $\mathrm{GAIP}_{\mathsf{IG}}$ instance, generated according to IG. The routines of $\mathcal{W}$ are described hereafter.

Challenge initialisation: on input $x = Q'$, $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_q \in \{-1, 1\}^\lambda$, $\mathcal{W}$

1. sets a hash table $\mathbb{T} \leftarrow \{\}$, a list $L^{\mathsf{Sign}} \leftarrow \{\}$, $\mathsf{ctr} \leftarrow 0$ and $\alpha \xleftarrow{\$} \{0,1\}$;
2. if $\alpha = 0$, sets $P_s \leftarrow Q'$ and $((P_v, \mathsf{pk}_\mathsf{K}), (A_v, \mathsf{pk}_\mathsf{K})) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, v)$ and ; else, $P_v \leftarrow Q'$ and $((P_s, \mathsf{pk}_\mathsf{K}), (A_s, \mathsf{pk}_\mathsf{K})) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, v)$;
3. sets $\mathsf{pk}_s \leftarrow P_s$, $\mathsf{sk}_s \leftarrow A_s$, $\mathsf{pk}_v \leftarrow (P_v, \mathsf{pk}_\mathsf{K})$ and $\mathsf{sk}_v \leftarrow (A_v, \mathsf{sk}_\mathsf{K})$.
4. gives input $(\mathsf{pk}_s, \mathsf{pk}_v)$ to $\mathcal{A}$ and simulates oracles queries for $\mathcal{A}$.

Simulation of $\mathcal{O}^{\mathcal{H}}$: on input a query $(m, K, E, \boldsymbol{Y}_s, \boldsymbol{Y}_v)$ to $\mathcal{O}^{\mathcal{H}}$, $\mathcal{W}$ checks whether $\mathbb{T}[m, K, E, \boldsymbol{Y}_s, \boldsymbol{Y}_v]$ exists, and returns the corresponding value $\boldsymbol{e}$ if it does; else sets $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$, $\mathbb{T}[m, K, E, \boldsymbol{Y}_s, \boldsymbol{Y}_v] \leftarrow \boldsymbol{e}_{\mathsf{ctr}}$ and returns $\boldsymbol{e}_{\mathsf{ctr}}$.

Simulation of $\mathcal{O}^{\mathsf{Sign}}$: when receiving a query m to $\mathcal{O}^{\mathsf{Sign}}$, $\mathcal{W}$ uses the algorithm Sim from the proof for Theorem 86, replacing its execution of the hash evaluation by a query to the simulator of $\mathcal{O}^{\mathcal{H}}$.

1. If $\alpha = 0$, $\mathcal{W}$ sets $(\sigma, E) \xleftarrow{\$} \mathsf{Sim}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m)$, else, $(\sigma, E) \xleftarrow{\$} \mathsf{GASDVS.Sign}(\mathsf{sk}_s, \mathsf{pk}_s, \mathsf{pk}_v, m)$;
2. it sets $L^{\mathsf{Sign}} \leftarrow L^{\mathsf{Sign}} \cup \{(m, (\sigma, E))\}$ and returns (σ, E).

Simulation of $\mathcal{O}^{\mathsf{Verify}}$: on input a query $(m, (\sigma, E))$ for $\mathcal{O}^{\mathsf{Verify}}$, $\mathcal{W}$ runs $\mathsf{GASDVS.Verify}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m, \sigma, E)$, replacing its execution of a hash evaluation by a query to the simulator of $\mathcal{O}^{\mathcal{H}}$, and returning the output b.

Final procedure: on input $(m^*, (\sigma^*, E^*))$ received from $\mathcal{A}$, setting $K^* \leftarrow \mathsf{KEM.Decaps}(\mathsf{sk}_\mathsf{K}, E^*)$ and parsing $(\boldsymbol{c}_s, \boldsymbol{c}_v, V, \boldsymbol{Y}_s, \boldsymbol{Y}_v) \leftarrow \sigma^*$ the algorithm:

1. sets $(\boldsymbol{U}_s, \boldsymbol{U}_v) \leftarrow \mathsf{SKE.Dec}(K, V)$ and $\pi^* \leftarrow (\boldsymbol{c}_s, \boldsymbol{c}_v, \boldsymbol{U}_s, \boldsymbol{U}_v)$;
2. aborts if $\mathbb{T}[m^*, K^*, E^*, \boldsymbol{Y}_s, \boldsymbol{Y}_v]$ does not exist; else if $(m^*, (\sigma^*, E^*)) \in L^{\mathsf{Sign}}$ or $\mathsf{Verify}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m^*, \sigma^*, E^*) \neq 1$, returns $(0, \perp)$;
3. for i^* an index such that $\boldsymbol{e}_{i^*} = \mathbb{T}[m^*, K^*, E^*, \boldsymbol{Y}_s, \boldsymbol{Y}_v]$, returns $(i^*, (\pi^*, \alpha, m^*, \sigma^*, E^*, K^*))$.

Construction of $\mathcal{R}$. This algorithm will take as input a $\mathrm{GAIP}_{\mathsf{IG}}$ challenge (Q, Q') and runs $\mathsf{Fork}^{\mathcal{W}}$ to extract a solution. Note that Ext_0 and Ext_1 from Figure 7 can also be interpreted as Ext_s and Ext_v, by changing the binary indices to $\{s, v\}$ ones.

$\mathcal{R}(Q') \to A$**:** given (Q'),

1. sets $(d, \mathsf{out}, \widetilde{\mathsf{out}}) \xleftarrow{\$} \mathsf{Fork}^{\mathcal{W}}(Q')$ and aborts if $d = 0$;
2. parses $(\pi^*, \alpha, m^*, \sigma^*, E^*, K^*) \leftarrow \mathsf{out}$ and $(\widetilde{\pi}^*, \widetilde{\alpha}, m^*, \widetilde{\sigma}^*, \widetilde{E}^*, \widetilde{K}^*) \leftarrow \widetilde{\mathsf{out}}$;
3. if $\alpha = 0$, sets $A_s \leftarrow \mathsf{Ext}_s(\pi^*, \widetilde{\pi}^*)$, else, $A_v \leftarrow \mathsf{Ext}_v(\pi^*, \widetilde{\pi}^*)$;
4. returns A (or $\perp$ if this was the extractor's output).

Analysis of $\mathcal{W}$. Let us first show that $\mathcal{W}$ perfectly simulates the view of $\mathcal{A}$. For the initialisation, the input to $\mathcal{W}$ is received from the $\mathrm{GAIP}_{\mathsf{IG}}$ challenger, generating $Q' \xleftarrow{\$} \mathsf{IG}(Q)$, so all keys are distributed exactly as when running $\mathsf{GASDVS.KeyGen}$; then, in the random oracle model, the $\mathcal{W}$'s simulation of $\mathcal{O}^{\mathcal{H}}$ and $\mathcal{O}^{\mathsf{Verify}}$ is perfect; when simulating $\mathcal{O}^{\mathsf{Sign}}$, if $\alpha = 1$, the simulation uses

$\mathsf{GASDVS.Sign}$ to sign, else for $\alpha = 0$, Theorem 86 shows that $\mathsf{Sim}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m)$ perfectly simulates the signing algorithm.

The forking lemma 59's $\mathsf{acc}(\lambda)$ function corresponds to the probability that $\mathcal{W}$ output (j, out) with $j \geq 1$. This event occurs when $\mathcal{A}$ outputs a valid forgery, with at least one corresponding hash query; without such a query, its advantage is bounded by $2^{-\lambda}$, hence $\mathsf{acc}(\lambda) \geq \mathsf{Adv}^{\mathsf{SUF}}_{\mathcal{A}}(\lambda) - \frac{1}{2^\lambda}$.

Analysis of $\mathcal{R}$. Consider an output $(d, \mathsf{out}, \widetilde{\mathsf{out}}) \xleftarrow{\$} \mathsf{Fork}^{\mathcal{W}}(Q')$, parsed as in the description of $\mathcal{R}$ while additionally parsing $(\boldsymbol{c}_s, \boldsymbol{c}_v, V, \boldsymbol{Y}_s, \boldsymbol{Y}_c) \leftarrow \sigma^*$ and $(\widetilde{\boldsymbol{c}}_s, \widetilde{\boldsymbol{c}}_v, \widetilde{V}, \widetilde{\boldsymbol{Y}}_s, \widetilde{\boldsymbol{Y}}_c) \leftarrow \widetilde{\sigma}^*$. If $d = 1$, the challenges of the signatures (σ^*, E^*) and $(\widetilde{\sigma}^*, E^*)$ corresponding to out and $\widetilde{\mathsf{out}}$ differ, *i.e.* $\boldsymbol{e}^* \neq \widetilde{\boldsymbol{e}}^*$, while $\mathsf{Verify}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m^*, \sigma^*, E^*) = \mathsf{Verify}(\mathsf{sk}_v, \mathsf{pk}_s, \mathsf{pk}_v, m^*, \widetilde{\sigma}^*, E^*) = 1$, and $(m^*, (\sigma^*, E^*)) \notin L^{\mathsf{Sign}}$, $(m^*, (\widetilde{\sigma}^*, E^*)) \notin \widetilde{L}^{\mathsf{Sign}}$, with the equality of keys, message and encapsulation stemming from the fact that $\mathcal{W}$ uses the same randomness in both cases, with identical inputs until $\boldsymbol{e}^*$ and $\widetilde{\boldsymbol{e}}^*$ are queried, and thus also identical states until that point. Via similar reason it also holds that $\boldsymbol{Y}_s = \widetilde{\boldsymbol{Y}}_s$, $\boldsymbol{Y}_v = \widetilde{\boldsymbol{Y}}_v$. By validity of the forgeries it holds that for all $i \in [\lambda]$, $c_{s,i} \cdot c_{v,i} = e_i$ and $\widetilde{c}_{s,i} \cdot \widetilde{c}_{v,i} = \widetilde{e}_i$. Since $\boldsymbol{e}^* \neq \widetilde{\boldsymbol{e}}^*$ it follows that $\boldsymbol{c}_s \neq \widetilde{\boldsymbol{c}}_s$ or $\boldsymbol{c}_v \neq \widetilde{\boldsymbol{c}}_v$. Calling Ext_s and Ext_v on $(\pi^*, \widetilde{\pi}^*)$ hence produces at least one successful extraction of either A_s such that $P_s = Q * A_s$ for $P_s = \mathsf{pk}_s$ or A_v such that $P_v = Q * A_v$ for $\mathsf{pk}_v = (P_v, \mathsf{pk}_{\mathsf{K}})$, by Lemma 67.

Let us now analyse the probability that $\mathcal{R}$ solves the $\mathrm{GAIP}_{\mathsf{IG}}$ instance, that is, it outputs A such that $(Q', A) \in \mathcal{R}_{Q,\lambda}$, *i.e.* $Q' = Q * A$. If $d = 0$ the extraction fails, and if $d = 1$ the following will hold: if the extractors found A_s such that $P_s = Q * A_s$, then this is a valid witness for the $\mathrm{GAIP}_{\mathsf{IG}}$ problem if $\alpha = 0$, which occurs with probability one half; else, they found A_v such that $P_v = Q * A_v$, which is a valid witness for Q' if $\alpha = 1$. Thus, the forking lemma 59 (Appendix F) provides

$$\begin{aligned}\mathsf{Adv}^{\mathrm{GAIP}_{\mathsf{IG}}}_{\mathcal{A}}(\lambda) &\geq \frac{1}{2} \cdot \mathsf{frk}(\lambda) \geq \frac{1}{2} \cdot \mathsf{acc}(\lambda) \cdot \left(\frac{\mathsf{acc}(\lambda)}{q} - \frac{1}{2^\lambda} \right) \\ &\geq \frac{1}{2q} \cdot \left(\mathsf{Adv}^{\mathsf{SUF}}_{\mathcal{A}}(\lambda) - \frac{1}{2^\lambda} \right) \cdot \left(\mathsf{Adv}^{\mathsf{SUF}}_{\mathcal{A}}(\lambda) - \frac{q+1}{2^\lambda} \right).\end{aligned}$$

□

I.4 Privacy of Signer Identities of GASDVS.

Theorem 88 (Privacy of Signer's Identity of GASDVS). *The* GASDVS *protocol satisfies privacy of signer's identity. More precisely, in the random oracle model, for any PPT adversary* $\mathcal{A}$ *and security parameter* λ *in* $\mathbb{N}$*, denoting* Q_s *the number of queries of* $\mathcal{A}$ *to the signing oracle* $\mathcal{O}^{\mathsf{Sign}}$*,*

$$\mathsf{Adv}^{\mathsf{PSI}}_{\mathcal{A}}(\lambda) \leq 2 \cdot \mathsf{Adv}^{\mathsf{SK\text{-}CCA}}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{IND\text{-}EAV}}_{\mathcal{A}}(\lambda) + \frac{Q_s}{2^\lambda} + \mathsf{Adv}^{\mathsf{SUF}}_{\mathcal{A}}(\lambda).$$

Proof. Let $\mathcal{A}$ be a PPT adversary, and λ in $\mathbb{N}$ a security parameter. Consider the following game sequence.

Game $\mathbf{G}_0$: is the original privacy of the signer identity game. The adversary's advantage is denoted $\mathsf{Adv}^{\mathsf{PSI}}_{\mathcal{A}}(\lambda)$.

Game $\mathbf{G}_1$: is as the previous game, but the challenger $\mathcal{C}$ now does not receive $\mathsf{sk}_v, \mathsf{sk}_0$ and sk_1 at the beginning of the game, and replaces the generation of σ^* by querying $\mathcal{O}^{\mathsf{Sign}}$ on (b, m^*) and setting its output as σ^*. This is a perfectly indistinguishable change for the adversary: $\mathsf{Adv}^{\mathbf{G}_1}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{PSI}}_{\mathcal{A}}(\lambda)$;

Game $\mathbf{G}_2$: is as the previous game, but $\mathcal{C}$ now changes the simulation of the oracles for the adversary: its initiates an empty list $\mathcal{L}^{\mathsf{Sign}}$, and for each query (β, m) the adversary issues to $\mathcal{O}^{\mathsf{Sign}}$, it registers the output signature σ with $\mathcal{L}^{\mathsf{Sign}} \leftarrow \mathcal{L}^{\mathsf{Sign}} \cup \{(m, \sigma)\}$. Then, when $\mathcal{O}^{\mathsf{Verify}}$ or $\mathcal{O}^{\mathsf{Verify}^*}$ are queried with (β', m', σ'), if (m', σ') is not in $\mathcal{L}^{\mathsf{Sign}}$, it replaces the output of the original oracle by 0 (except if (m^*, σ^*) is queried to $\mathcal{O}^{\mathsf{Verify}^*}$: then is still returns $\perp$). The adversary's advantage in distinguishing between this game and the previous one is bounded by its advantage in the strong unforgeability game: $\left|\mathsf{Adv}^{\mathbf{G}_2}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathbf{G}_1}_{\mathcal{A}}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{SUF}}_{\mathcal{A}}(\lambda)$.

Game $\mathbf{G}_3$: is as the previous game, except that now, $\mathcal{O}^{\mathsf{Sign}}$ is replaced by a simulator running its normal procedure and sending not only $\sigma^* = (s^*, E^*)$ to $\mathcal{C}$ but also the session key K^* corresponding to E^*. $\mathcal{C}$ aborts if there exist m and s such that $(m, (s, E^*))$ is in $\mathcal{L}^{\mathsf{Sign}}$. If in any of the subsequent requests of $\mathcal{A}$ to $\mathcal{O}^{\mathsf{Sign}}$, the oracle outputs a signature of the form $(m, (s, E^*))$, it also aborts. The difference between this game and the previous one consists in events that would allow $\mathcal{C}$ to win the $\mathsf{SK\text{-}CCA}$ game for E^* (with (E^*, K^*) corresponding to an encapsulation generation using pk_{K} that would yield the session key corresponding to the challenge E^* received as answer to another signing request), hence $\left|\mathsf{Adv}^{\mathbf{G}_3}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathbf{G}_2}_{\mathcal{A}}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{SK\text{-}CCA}}_{\mathcal{A}}(\lambda)$.

Game $\mathbf{G}_4$: is as the previous game, except that $\mathcal{C}$ does not receive K^* anymore, which does not change the view of $\mathcal{A}$: $\mathsf{Adv}^{\mathbf{G}_4}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathbf{G}_3}_{\mathcal{A}}(\lambda)$.

Game $\mathbf{G}_5$: is as the previous game, except that the simulator of $\mathcal{O}^{\mathsf{Sign}}(b, m^*)$ now draws $\widetilde{K} \xleftarrow{\$} \{0,1\}^\lambda$ and sets $V' \xleftarrow{\$} \mathsf{SKE.Enc}(\widetilde{K}, (\boldsymbol{U}^*_s, \boldsymbol{U}^*_v))$ instead of the usual computation with the KEM session-key before returning $\sigma^* = (\boldsymbol{c}^*_s, \boldsymbol{c}^*_v, V', \boldsymbol{Y}^*_s, \boldsymbol{Y}^*_v, E^*)$ to $\mathcal{C}$. The adversary's advantage in the KEM $\mathsf{SK\text{-}CCA}$ game bounds the one it has in distinguishing the current game from the previous one: $\left|\mathsf{Adv}^{\mathbf{G}_5}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathbf{G}_4}_{\mathcal{A}}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{SK\text{-}CCA}}_{\mathcal{A}}(\lambda)$.

Game $\mathbf{G}_6$: is as the previous game, except that the simulator for $\mathcal{O}^{\mathsf{Sign}}$ now aborts if any of the session-keys on other inputs than (b, m^*) are equal to $\widetilde{K}$. Denoting Q_s the number of queries of $\mathcal{A}$ to $\mathcal{O}^{\mathsf{Sign}}$, $\left|\mathsf{Adv}^{\mathbf{G}_6}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathbf{G}_5}_{\mathcal{A}}(\lambda)\right| \leq \frac{Q_s}{2^\lambda}$.

Game $\mathbf{G}_7$: is as the previous game, except that now, the simulator of $\mathcal{O}^{\mathsf{Sign}}(b, m^*)$ for $\mathcal{C}$, denoting $\boldsymbol{I}$ the neutral element of $\mathbb{G}$, sets $\widetilde{V} \xleftarrow{\$} \mathsf{SKE.Enc}(\widetilde{K}, (\boldsymbol{I}, \boldsymbol{I}))$ and returns $\sigma^* = (\boldsymbol{c}^*_s, \boldsymbol{c}^*_v, \widetilde{V}, \boldsymbol{Y}^*_s, \boldsymbol{Y}^*_v, E^*)$ to $\mathcal{C}$. $\mathcal{A}$'s advantage in distinguish-

ing this situation from the previous game is bounded by its IND-EAV advantage: $\left|\mathsf{Adv}^{\mathbf{G}_7}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathbf{G}_6}_{\mathcal{A}}(\lambda)\right| \leq \mathsf{Adv}^{\mathsf{IND\text{-}EAV}}_{\mathcal{A}}(\lambda)$.

Game $\mathbf{G}_8$: is as the previous game, except that now $\mathcal{C}$ draws $\widetilde{\boldsymbol{c}}_s$ and $\widetilde{\boldsymbol{c}}_v$ uniformly at random from $\{0,1\}^\lambda$ and replaces these values in the challenge signature $\sigma^* = (\widetilde{\boldsymbol{c}}_s, \widetilde{\boldsymbol{c}}_v, \widetilde{V}, \boldsymbol{Y}^*_s, \boldsymbol{Y}^*_v, E^*)$ it returns to $\mathcal{A}$. In the random oracle model, this is perfectly indistinguishable from the previous game, thus $\mathsf{Adv}^{\mathbf{G}_8}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathbf{G}_7}_{\mathcal{A}}(\lambda)$.

Game $\mathbf{G}_9$: is as the previous game, except that now $\mathcal{C}$ draws $\widetilde{\boldsymbol{Y}}_s, \widetilde{\boldsymbol{Y}}_v \xleftarrow{\$} \mathsf{IG}(Q)$ and replaces these values in the challenge signature $\sigma^* = (\widetilde{\boldsymbol{c}}_s, \widetilde{\boldsymbol{c}}_v, \widetilde{V}, \widetilde{\boldsymbol{Y}}_s, \widetilde{\boldsymbol{Y}}_v, E^*)$ it returns to $\mathcal{A}$. As IG is an instance generator satisfying definition 6, this game is perfectly indistinguishable from the previous one: $\mathsf{Adv}^{\mathbf{G}_9}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathbf{G}_8}_{\mathcal{A}}(\lambda)$.

Game $\mathbf{G}_{10}$: is as the previous game, but now $\mathcal{C}$ does not query $\mathcal{O}^{\mathsf{Sign}}(b, m^*)$, sets $(\widetilde{k}, \widetilde{E}) \xleftarrow{\$} \mathsf{KEM.Encaps}(\mathsf{pk}_\mathsf{K})$, aborts if there is an element of the form $(m, (s, \widetilde{E}))$ in $\mathcal{L}^{\mathsf{Sign}}$, and sets $\sigma^* = (\widetilde{\boldsymbol{c}}_s, \widetilde{\boldsymbol{c}}_v, \widetilde{V}, \widetilde{\boldsymbol{Y}}_s, \widetilde{\boldsymbol{Y}}_v, \widetilde{E})$ before returning σ^* to $\mathcal{A}$. Then, it aborts if any of the adversary's requests to $\mathcal{O}^{\mathsf{Sign}}$ results in a new element of the form $(m, (s, \widetilde{E}))$ in $\mathcal{L}^{\mathsf{Sign}}$. The adversary's view is not modified, so $\mathsf{Adv}^{\mathbf{G}_{10}}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathbf{G}_9}_{\mathcal{A}}(\lambda)$, and moreover, in this last game, $\mathcal{C}$ never uses b, and as $\mathcal{A}$ receives no element dependent on this bit, her advantage is null.

Finally, $\mathsf{Adv}^{\mathsf{PSI}}_{\mathcal{A}}(\lambda) \leq 2 \cdot \mathsf{Adv}^{\mathsf{SK\text{-}CCA}}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{IND\text{-}EAV}}_{\mathcal{A}}(\lambda) + \frac{Q_s}{2^\lambda} + \mathsf{Adv}^{\mathsf{SUF}}_{\mathcal{A}}(\lambda)$.

□

I.5 Non-Delegatability of GASDVS.

Theorem 89 (Non-Delegatability of GASDVS). *The* GASDVS *protocol satisfies non-delegatability with knowledge error $2^{-\lambda}$ in the random oracle model if the hardness of the* $\mathrm{GAIP}_{\mathsf{IG}}$ *holds.*

Proof. Adapting the proof strategy of [MTT25] to our KEM-based construction, given a PPT forger algorithm $\mathcal{F}$ that on input m^* has success probability ϵ of outputting a valid GASDVS signature for m^*, the extractor Ext described in this proof will show that it may be used to build an efficient extractor that retrieves signer secret key $\mathsf{sk}_s = A_s$ or the A_v component of the verifier secret key.

Additionally, our hash oracle $\mathcal{O}^{\mathcal{H}}$ maintains a list of keys K that are used as input components for hash queries, so that they can be retrieved in polynomial time by searching the list. This allows us to find a key K^* that decrypts the value V as $(\boldsymbol{U}_s, \boldsymbol{U}_s)$ such that an adapted verification algorithm is satisfied.

For λ a security parameter in $\mathbb{N}$, let $\mathcal{Q}$ be the corresponding set for IG and let $Q \in \mathcal{Q}$ be fixed. Let two pairs of keys be given by $(\mathsf{pk}_s, \mathsf{sk}_s) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, s)$ and $(\mathsf{pk}_v, \mathsf{sk}_v) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, v)$, where we pare $P_s \leftarrow \mathsf{pk}_s$, $A_s \leftarrow \mathsf{sk}_s$, $(P_v, \mathsf{pk}_\mathsf{K}) \leftarrow \mathsf{pk}_v$ and $(A_v, \mathsf{sk}_\mathsf{K}) \leftarrow \mathsf{sk}_v$. Let q denote the amount of hash queries that $\mathcal{F}_{m^*}$ makes to $\mathcal{O}^{\mathcal{H}}$ and let $\kappa := 2^{-\lambda}$ where $2^{-\lambda}$ bounds the probability that the adversary

generates a forgery without having queried the hash value for the verification of that forgery. Let ϵ denote the success probability of $\mathcal{F}_{m^*}$.

In order to describe Ext, we make use of the $\mathsf{Fork}^{\mathcal{W}}$ algorithm (Figure 10). For the analysis of Ext we use the forking Lemma 59 (Appendix F). Let us first describe the algorithm $\mathcal{W}^{\mathcal{F}_{m^*}}$.

Description of $\mathcal{W}^{\mathcal{F}_{m^*}}$. On input $(m^*, \boldsymbol{e}_1, ..., \boldsymbol{e}_q)$, the algorithm $\mathcal{W}$ runs $\mathcal{F}_{m^*}$ and simulates the required has queries. The values $\boldsymbol{e}_1, ..., \boldsymbol{e}_q \in \{0,1\}^\lambda$ are output values for the simulated hash queries.

Challenge initialisation: on input $(m^*, \boldsymbol{e}_1, ..., \boldsymbol{e}_q)$, $\mathcal{W}$ initialises a hash table $\mathbb{T} \leftarrow \{\}$, a counter $\mathsf{ctr} \leftarrow 0$ and a list $\mathcal{L} \leftarrow \{\}$.

Simulation of $\mathcal{O}^{\mathcal{H}}$: on input $(m, K, E, \boldsymbol{Y}_s, \boldsymbol{Y}_v)$ the simulation is as in the proof for Theorem 87 and additionally sets $\mathcal{L} \leftarrow \mathcal{L} \cup \{K\}$.

Final procedure: $\mathcal{W}$ runs $(m^*, \sigma^*, E^*) \xleftarrow{\$} \mathcal{F}_{m^*}$; searches for $K^* \in \mathcal{L}$ such that $\widetilde{\mathsf{Verify}}(\mathsf{pk}_s, \mathsf{pk}_v, K^*, m^*, \sigma^*) = 1$, where $\widetilde{\mathsf{Verify}}$ is as $\mathsf{GASDVS.Verify}$ except that it uses K^* directly as the SKE key instead of using $\mathsf{sk_K}$ and retrieving it from decapsulation of E^*; aborts if no such K^* is found; parses $(\boldsymbol{c}_s, \boldsymbol{c}_v, V, \boldsymbol{Y}_s, \boldsymbol{Y}_v) \leftarrow \sigma^*$; if $\mathbb{T}[m^*, K^*, E^*, \boldsymbol{Y}_s, \boldsymbol{Y}_v]$ does not exist then $\mathcal{W}$ aborts; otherwise, $\mathcal{W}$ computes $(\boldsymbol{U}_s, \boldsymbol{U}_v) \leftarrow \mathsf{SKE.Dec}(K^*, V)$, sets $\pi^* \leftarrow (\boldsymbol{c}_s, \boldsymbol{c}_v, \boldsymbol{U}_s, \boldsymbol{U}_v)$ and returns $(i^*, (\pi^*, m^*, \sigma^*, E^*, K^*, \boldsymbol{Y}_s, \boldsymbol{Y}_v))$, where i^* is an index such that $\mathbb{T}[m^*, K^*, E^*, \boldsymbol{Y}_s, \boldsymbol{Y}_v] = \boldsymbol{e}_{i^*}$.

Description of Ext. This algorithm takes as input a message m^* and runs $\mathsf{Fork}^{\mathcal{W}^{\mathcal{F}_{m^*}}}$ to extract two GASDVS forgeries that correspond to the same hash query. One of the secret keys can then be extracted from these forgeries. Note that Ext_0 and Ext_1 from Figure 7 can also be interpreted as Ext_s and Ext_v, by changing the binary indices to $\{s, v\}$ ones.

$\mathsf{Ext}(m^*) \rightarrow A$**:** given m^*,

1. sets $(d, \mathsf{out}, \widetilde{\mathsf{out}}) \xleftarrow{\$} \mathsf{Fork}^{\mathcal{W}^{\mathcal{F}_{m^*}}}(m^*)$ and aborts if $d = 0$;
2. parses $(\pi^*, m^*, \sigma^*, E^*, K^*) \leftarrow \mathsf{out}$, $(\widetilde{\pi}^*, \widetilde{m}^*, \widetilde{\sigma}^*, \widetilde{E}^*, \widetilde{K}^*) \leftarrow \widetilde{\mathsf{out}}$, $(\boldsymbol{c}_s, \boldsymbol{c}_v, V, \boldsymbol{Y}_s, \boldsymbol{Y}_v) \leftarrow \sigma^*$ and $(\widetilde{\boldsymbol{c}}_s, \widetilde{\boldsymbol{c}}_v, \widetilde{V}, \widetilde{\boldsymbol{Y}}_s, \widetilde{\boldsymbol{Y}}_v) \leftarrow \widetilde{\sigma}^*$;
3. if $\boldsymbol{c}_s \neq \widetilde{\boldsymbol{c}}_s$, sets $A'_s \leftarrow \mathsf{Ext}_s(\pi^*, \widetilde{\pi}^*)$, if $\boldsymbol{c}_v \neq \widetilde{\boldsymbol{c}}_v$, sets $A'_v \leftarrow \mathsf{Ext}_v(\pi^*, \widetilde{\pi}^*)$;
4. returns A'_s if $A'_s \neq \perp$ or A'_v if $A'_v \neq \perp$ (or $\perp$ if this was both extractor's output).

Analysis of $\mathcal{W}$. In the random oracle model, $\mathcal{W}^{\mathcal{F}_{m^*}}$'s simulation of $\mathcal{O}^{\mathcal{H}}$ is perfect. Therefore, $\mathcal{W}^{\mathcal{F}_{m^*}}$ perfectly simulated the view of $\mathcal{F}_{m^*}$.

Suppose $\mathcal{F}_{m^*}$ outputs a valid forgery (σ^*, E^*). The key $K^* = \mathsf{Decaps}(\mathsf{sk_K}, E^*)$ must be in L if $\mathcal{F}_{m^*}$ made the hash query corresponding to (σ^*, E^*). The condition that $\widetilde{\mathsf{Verify}}(\mathsf{pk}_s, \mathsf{pk}_v, K^*, m^*, \sigma^*) = 1$ is clearly satisfied for $K^* = \mathsf{Decaps}(\mathsf{sk_K}, E^*)$.

The forking lemma's (Lemma 59) $\mathsf{acc}(\lambda)$ function corresponds to the probability that $\mathcal{W}^{\mathcal{F}_{m^*}}$ output (j, out) with $j \geq 1$. This event occurs when $\mathcal{F}_{m^*}$ outputs a valid forgery, with at least one corresponding hash query; without such a query, its advantage is bounded by $2^{-\lambda}$, hence $\mathsf{acc}(\lambda) \geq \epsilon - \frac{1}{2^\lambda}$.

Analysis of Ext. Consider an output $(d, \mathsf{out}, \widetilde{\mathsf{out}}) \xleftarrow{\$} \mathsf{Fork}^{\mathcal{W}^{\mathcal{F}_{m^*}}}(m^*)$, parsed as in the description of Ext. If $d = 1$, the challenges, for verification under $\widetilde{\mathsf{Verify}}$, of the signatures (m^*, σ^*, E^*) and $(m^*, \widetilde{\sigma}^*, E^*)$ differ, *i.e.* $\boldsymbol{e}^* \neq \widetilde{\boldsymbol{e}}^*$, while $\widetilde{\mathsf{Verify}}(\mathsf{pk}_s, \mathsf{pk}_v, K^*, m^*, \sigma^*, E^*) = \widetilde{\mathsf{Verify}}(\mathsf{pk}_s, \mathsf{pk}_v, \widetilde{K}^*, m^*, \widetilde{\sigma}^*, E^*) = 1$, with the equality of keys, message and encapsulation stemming from the fact that $\mathcal{W}^{\mathcal{F}_{m^*}}$ uses the same randomness in both cases, with identical inputs until $\boldsymbol{e}^*$ and $\widetilde{\boldsymbol{e}}^*$ are queried, and thus also identical states up until that point. Via similar reason it also holds that $\boldsymbol{Y}_s = \widetilde{\boldsymbol{Y}}_s$ and $\boldsymbol{Y}_v = \widetilde{\boldsymbol{Y}}_v$. It holds that $\widetilde{\mathsf{Verify}}(\mathsf{pk}_s, \mathsf{pk}_v, K^*, m^*, \sigma^*, E^*) = \widetilde{\mathsf{Verify}}(\mathsf{pk}_s, \mathsf{pk}_v, \widetilde{K}^*, m^*, \widetilde{\sigma}^*, E^*) = 1$ so that for all $i \in [\lambda]$, $c_{s,i} \cdot c_{v,i} = e_i$ and $\widetilde{c}_{s,i} \cdot \widetilde{c}_{v,i} = \widetilde{e}_i$. Since $\boldsymbol{e}^* \neq \widetilde{\boldsymbol{e}}^*$ it follows that $\boldsymbol{c}_s \neq \widetilde{\boldsymbol{c}}_s$ or $\boldsymbol{c}_v \neq \widetilde{\boldsymbol{c}}_v$. Calling Ext_s and Ext_v on $(\pi^*, \widetilde{\pi}^*)$ hence produces at least one successful extraction of either A'_s such that $P_s = Q * A'_s$ or A'_v such that $P_v = Q * A'_v$ by Lemma 67.

Let us now analyse the probability ϵ_{Ext} that Ext outputs sk'_s such that the distribution of $\mathsf{Sign}(\mathsf{sk}', \mathsf{pk}_s, \mathsf{pk}_v, \cdot)$ is equal to the one of $\mathsf{Sign}(\mathsf{sk}_s, \mathsf{pk}_s, \mathsf{pk}_v, \cdot)$. Recall that $\mathsf{pk}_s = P_s$ and $\mathsf{pk}_v = (P_v, \mathsf{pk}_\mathsf{K})$. If $d = 0$ the extraction fails, and if $d = 1$ the following will hold: the extractors will have found A such that $P_s = Q * A$ or $P_v = Q * A$, and by witness indistinguishability of $\Sigma^{\mathsf{OR}}_{Q,Q,\lambda}$ (implied by the perfect honest-verifier zero-knowledge), the distributions of its algorithms on this witness to the OR relation or another one are the same, hence $\mathsf{Sign}(A, \mathsf{pk}_s, \mathsf{pk}_v, \cdot) = \mathsf{Sign}(\mathsf{sk}_s, \mathsf{pk}_s, \mathsf{pk}_v, \cdot)$. Finally, the forking lemma 59 (Appendix F) provides

$$\begin{aligned}\epsilon_{\mathsf{Ext}} \geq \mathsf{frk}(\lambda) &\geq \mathsf{acc}(\lambda) \cdot \left(\frac{\mathsf{acc}(\lambda)}{q} - \frac{1}{2^\lambda}\right) \\ &\geq \left(\epsilon - \frac{1}{2^\lambda}\right) \cdot \left(\frac{\epsilon}{q} - \frac{1}{2^\lambda}\right).\end{aligned}$$

As q is polynomially bounded in λ and $\kappa = \frac{1}{2^\lambda}$, it follows that GASDVS satisfies non-delegatability.

□