This is the full version of a paper which appears in Cryptology and Network Security - 15th International Conference, CANS 2016, Milan, Italy, November 14-16, 2016. Proceedings, Sara Foresti and Giuseppe Persiano Eds., Springer, LNCS 10052, pages 211–227, 10.1007/978-3-319-48965-0_13.

Signer-Anonymous Designated-Verifier Redactable Signatures for Cloud-Based Data Sharing

David Derler[1,‡], Stephan Krenn[2,‡,‖], and Daniel Slamanig[1,‡]

[1] IAIK, Graz University of Technology, Graz, Austria
{david.derler|daniel.slamanig}@tugraz.at

[2] AIT Austrian Institute of Technology GmbH, Vienna, Austria
stephan.krenn@ait.ac.at

Abstract. Redactable signature schemes allow to black out predefined parts of a signed message without affecting the validity of the signature, and are therefore an important building block in privacy-enhancing cryptography. However, a second look shows, that for many practical applications, they cannot be used in their vanilla form. On the one hand, already the identity of the signer may often reveal sensitive information to the receiver of a redacted message; on the other hand, if data leaks or is sold, everyone getting hold of (redacted versions of) a signed message will be convinced of its authenticity.

We overcome these issues by providing a definitional framework and practically efficient instantiations of so called *signer-anonymous designated-verifier redactable signatures* (AD-RS). As a byproduct we also obtain the first *group redactable signatures*, which may be of independent interest. AD-RS are motivated by a real world use-case in the field of health care and complement existing health information sharing platforms with additional important privacy features. Moreover, our results are not limited to the proposed application, but can also be directly applied to various other contexts such as notary authorities or e-government services.

1 Introduction

Digitalization of data and processes as well as the use of promising IT-trends such as cloud computing is prevalent, steadily increasing and meanwhile outreaches even sensitive fields such as the health care sector.[1] Given the sensitivity of the involved data and the high demands in data correctness and quality, the health care domain is a prime example for the beneficial application of cryptographic means such as encryption and digital signatures. This work is dedicated to the development of a cryptographically enhanced solution for a real world hospital, which is currently planning to complement its existing information sharing system for electronic patient data with additional privacy features. The overall idea

‡ Supported by EU H2020 project Prismacloud, grant agreement n°644962.

‖ Supported by EU H2020 project Credential, grant agreement n°653454.

[1] See e.g., www.healthcaredive.com/news/407746/

of the system is to grant patients access to all their medical records via a cloud-based platform. The patients are then able to use this as a central hub to distribute their documents to different stakeholders, e.g., to request reimbursement by the insurance, or to forward (parts of) the documents to the family doctor for further treatment. While means for access control and data confidentiality are already in place, the system should be complemented by strong authenticity guarantees. At the same time a high degree of privacy should be maintained, i.e., by allowing the patients, on a fine-granular basis, to decide which parts of which document should be visible to which party. For instance, the family doctor might *not* need to learn the precise costs of a treatment; similarly a medical research laboratory should *not* learn the patients' identities.

From a research point of view, one motivation behind this work is to show how rather complex real world scenarios with conflicting interests and strong security and privacy requirements can be elegantly and securely realized by means of rigorous cryptographic design and analysis. More importantly, we can indeed come up with provably secure and practical solutions being well suited for real world use. Now, we discuss the motivation for our design.

Redactable Signatures. A trivial solution for the above problem would be to let the hospital cloud create a fresh signature on the information to be revealed every time the user wishes to forward authentic subsets of a document to other parties. However, this is not satisfactory as it would require strong trust assumptions into the cloud: one could not efficiently guarantee that the signed data has not been altered over time by the cloud or by a malicious intruder. It is therefore preferable to use *redactable signatures* (RS). These are signature schemes that allow to black out (redact) predefined parts of a signed message while preserving the validity of the signature, thereby guaranteeing the authenticity of the redacted message. That is, it is not necessary to let the cloud attest the authenticity of the forwarded data, as the signature on the redacted document can be extracted from the doctor's signature on the original document without requiring the doctor's secret signing key or further interaction with the doctor.

Designated Verifiers. Unfortunately, using redactable signatures in their vanilla form in our scenario would lead to severe privacy problems, i.e., everyone getting hold of a signed document would be convinced of its authenticity. In such a case, for instance, an employer who gets hold of a signed health record of an employee, might reliably learn the employee's disease, who, in further consequence, might get dismissed. What is therefore needed is a *designated verifier* for each redacted version of a document. That is, when redacting a document, the patient should be able to define the intended receiver. Then, while everybody can check the validity of a leaked document, *only* the designated verifier is convinced about its authenticity. This can be achieved by constructing the schemes in a way that the designated verifier can fake indistinguishable signatures on its own. Moreover, the public verifiability property might as well be a motivation for designated verifiers to not leak/sell documents, as this reduces the circle of possible suspects to the data owner and the designated verifier.

Group Signatures. Another problem of RS is that they only support a single signer. However, a hospital potentially employing hundreds of doctors will not use a single signing key that is shared by all its employees. By doing so, the identity of the signing doctor could not be revealed in case of a dispute, e.g., after a malpractice. However, using different keys for different doctors poses a privacy risk again. For instance, if the document was signed using an oncologist's key, one could infer sensitive information about the disease—even though the diagnosis was blacked out. What is therefore needed are features known from *group signatures*, where towards the verifier the doctor's identity remains hidden within the set of doctors in the hospital, while re-identification is still possible by a dedicated entity.

Contribution. The properties we need for our scenario are contributed by three distinct cryptographic concepts and what we actually need can be considered as a *signer-anonymous designated-verifier redactable signature scheme.* However, while a lot of existing work studies the different concepts in isolation, there is no work which aims at combining them in a way to profit from a combination of their individual properties. Trying to obtain this by simply combining them in an ad-hoc fashion, however, is dangerous. It is well known that the ad-hoc combination of cryptographic primitives to larger systems is often problematic (as subtle issues often remain hidden when omitting formal analysis) and security breaches resulting from such approaches are often seen in practice. Unlike following such an ad-hoc approach, we follow a rigorous approach and formally model what is required by the use-case, introduce a comprehensive security model and propose two (semi-)black-box constructions that are provably secure within our model. While such a (semi-)black-box construction is naturally interesting from a theoretical point of view, our second construction is also entirely practical and thus also well suited to be used within the planned system. Finally, as a contribution which may be of independent interest, we also obtain the first *group redactable signatures* as a byproduct of our definitional framework.

Technical Overview. Our constructions provably achieve the desired functionality by means of a two-tier signature approach: a message is signed using a freshly generated RS key pair where the corresponding public key of this "one-time RS" is certified using a group signature. For the designated verifier feature, we follow two different approaches. Firstly, we follow the naïve approach and use a disjunctive non-interactive proof of knowledge which either demonstrates knowledge of a valid RS signature on the message, *or* it demonstrates knowledge of a valid signature of the designated verifier on the same message. While this approach is very generic, its efficiency largely depends on the complexity to prove knowledge of an RS signature. To this end, we exploit key-homomorphic signatures, which we introduce and which seem to be of independent interest. In particular, we use the observation that a large class of RS can easily be turned into RS admitting the required key-homomorphism, to obtain a practical construction. More precisely, besides conventional group signatures and conventional redactable signatures, our approach only requires to prove a single statement demonstrating knowledge of the relation between two RS keys *or* demonstrating knowledge of the designated verifier's secret key. For instance, in the discrete logarithm setting when instantiating this proof using Fiat-Shamir transformed [FS86] Σ-protocols, they are highly efficient as they only require two group exponentiations.

Related Work. Redactable signature schemes have been independently introduced in [JMSW02] and [SBZ01]. Although such schemes suffer from the aforementioned problems, we can use them as an important building block. In particular, we will rely on the general framework for such signatures as presented in [DPSS15]. Besides that, redactable signatures with an unlinkability property have been introduced in [CDHK15, PS15].[2] Unfortunately, apart from lacking practical efficiency, even unlinkable redactable signatures are not useful to achieve the desired designated verifier functionality. There is a large body of work on signatures with designated verifiers, which are discussed subsequently. However, none of the approaches considers selective disclosure via redaction or a group signing feature.

In designated verifier (DV) signatures (or proofs) [JSI96], a signature produced by a signer can only be validated by a single user who is designated by the signer during the signing process (cf. [LWB05] for a refined security model). Designation can only be performed by the signer and verification requires the designated verifier's secret. Thus, this concept is not directly applicable to our setting. In [JSI96] also the by now well known "*OR trick*" was introduced as a DV construction paradigm.

Undeniable signatures [CA89] are signatures that can not be verified without the signer's cooperation and the signer can either prove that a signature is valid or invalid. This is not suitable for us as this is an interactive process.

Designated confirmer signatures [Cha94] introduce a third entity besides the signer and the verifier called designated confirmer. This party, given a signature, has the ability to privately verify it as well as to convince anyone of its validity or invalidity. Additionally, the designated confirmer can convert a designated confirmer signature into an ordinary signature that is then publicly verifiable. This is not suitable for our scenario, as it is exactly the opposite of what we require, i.e., here the signature for the confirmer is not publicly verifiable, but the confirmer can always output publicly verifiable versions of this signature.

Another concept, which is closer to the designation functionality that we require, are universal designated verifier (UDV) signatures introduced in [SBWP03]. They are similar to designated verifier signatures, but universal in the sense that any party who is given a publicly verifiable signature from the signer can designate the signature to any designated verifier by using the verifiers public key. Then, the designated verifier can verify that the message was signed by the signer, but is unable to convince anyone else of this fact. Like with ordinary DV signatures, UDV signatures also require the designated verifier's secret key for verification. There are some generic results for UDV signatures. In [Ver06] it was shown how to convert various pairing-based signature schemes into UDV signatures. In [SS08] it was shown how to convert a large class of signature schemes

[2] Similar to the related concept of unlinkable sanitizable signatures [BFLS10, BPS13, FKM$^+$16, LZCS16].

into UDV signatures. Some ideas in our second construction are conceptually related to this generic approach. However, as we only require to prove relations among public keys, our approach is more tailored to efficiency.

2 Preliminaries

We denote algorithms by sans-serif letters, e.g., A, B. All algorithms are assumed to return a special symbol $\perp$ on error. By $y \leftarrow \mathsf{A}(x)$, we denote that y is assigned the output of the potentially probabilistic algorithm A on input x and fresh random coins. Similarly, $y \overset{R}{\leftarrow} S$ means that y was sampled uniformly at random from a set S. We let $[n] := \{1, \ldots, n\}$. We write $\Pr[\Omega : \mathcal{E}]$ to denote the probability of an event $\mathcal{E}$ over the probability space Ω. We use $\mathcal{C}$ to denote challengers of security experiments, and $\mathcal{C}_\kappa$ to make the security parameter explicit.

A function $\varepsilon(\cdot) : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ is called negligible, iff it vanishes faster than every inverse polynomial, i.e., $\forall\ k : \exists\ n_k : \forall\ n > n_k : \varepsilon(n) < n^{-k}$.

Followingly, we recap required cryptographic building blocks. Due to space constraints we omit formal definitions for well known primitives such as a digital signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ and a (non-interactive) proof system $\Pi = (\mathsf{Setup}, \mathsf{Proof}, \mathsf{Verify})$ here, and present them in Appendix A.

Redactable Signatures. Below, we recall the generalized model for redactable signatures from [DPSS15], which builds up on [BBD+10]. As done in [DPSS15], we do not make the structure of the message explicit. That is, we assume that the message m to be signed is some arbitrarily structured data. We use $\textsc{adm}$ to denote a data structure encoding the admissible redactions of some messeage m and we use $\textsc{mod}$ to denote a data structure containing modification instructions for some message. We use $\mathring{\mathsf{m}} \overset{\textsc{adm}}{\preceq} \mathsf{m}$ to denote that a message $\mathring{\mathsf{m}}$ is derivable from a message m under $\textsc{adm}$ and $\mathring{\mathsf{m}} \overset{\textsc{mod}}{\longleftarrow} \mathsf{m}$ to denote that $\mathring{\mathsf{m}}$ is obtained by applying $\textsc{mod}$ to m. Likewise, we use $\mathring{\textsc{adm}} \overset{\textsc{mod}}{\longleftarrow} \textsc{adm}$ to denote the derivation of $\mathring{\textsc{adm}}$ from $\textsc{adm}$ with respect to $\textsc{mod}$. We use $\textsc{adm} \preceq \mathsf{m}$ to denote that $\textsc{adm}$ matches m, and $\textsc{mod} \preceq \textsc{adm}$ to denote that $\textsc{mod}$ matches $\textsc{adm}$.

Definition 1. *An* RS *is a tuple* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Redact})$ *of PPT algorithms, which are defined as follows:*

$\mathsf{KeyGen}(1^\kappa)$: *Takes a security parameter* κ *as input and outputs a keypair* $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{Sign}(\mathsf{sk}, \mathsf{m}, \textsc{adm})$: *Takes a secret key* sk*, a message* m *and admissible modifications* $\textsc{adm}$ *as input, and outputs a message-signature pair* (m, σ) *together with some auxiliary redaction information* $\textsc{red}$.[3]

$\mathsf{Verify}(\mathsf{pk}, \mathsf{m}, \sigma)$: *Takes a public key* pk*, a message* m*, and a signature* σ *as input, and outputs a bit* b.

$\mathsf{Redact}(\mathsf{pk}, \mathsf{m}, \sigma, \textsc{mod}, \textsc{red})$: *Takes a public key* pk*, a message* m*, a valid signature* σ*, modification instructions* $\textsc{mod}$*, and auxiliary redaction information* $\textsc{red}$ *as input. It returns a redacted message-signature pair* $(\mathring{\mathsf{m}}, \mathring{\sigma})$ *and an updated auxiliary redaction information* $\mathring{\textsc{red}}$.

[3] As it is common for RS, we assume that $\textsc{adm}$ can always be recovered from (m, σ).

While we omit correctness, we recall the remaining RS security definitions below.

Definition 2 (Unforgeability). *An* RS *is unforgeable, if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l}(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa), \\ (\mathsf{m}^\star,\sigma^\star) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot,\cdot)}(\mathsf{pk})\end{array} : \begin{array}{r}\mathsf{Verify}(\mathsf{pk},\mathsf{m}^\star,\sigma^\star)=1\ \wedge \\ \nexists\,(\mathsf{m},\mathsf{ADM}) \in \mathcal{Q}^{\mathsf{Sign}} : \mathsf{m}^\star \overset{\mathsf{ADM}}{\preceq} \mathsf{m}\end{array}\right] \leq \varepsilon(\kappa),$$

where the environment keeps track of the queries to the signing oracle via $\mathcal{Q}^{\mathsf{Sign}}$.

Definition 3 (Privacy). *An* RS *is private, if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l}(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa),\ b \xleftarrow{R} \{0,1\}, \\ \mathcal{O} \leftarrow \{\mathsf{Sign}(\mathsf{sk},\cdot,\cdot),\mathsf{LoRRedact}(\mathsf{sk},\mathsf{pk},\cdot,\cdot,b)\},\ :\ b=b^\star \\ b^\star \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk})\end{array}\right] \leq 1/2+\varepsilon(\kappa),$$

where $\mathsf{LoRRedact}$ *is defined as follows:*

```
$\mathsf{LoRRedact}(\mathsf{sk},\mathsf{pk},(\mathsf{m}_0,\mathsf{ADM}_0,\mathsf{MOD}_0),(\mathsf{m}_1,\mathsf{ADM}_1,\mathsf{MOD}_1),b)$:
  1: Compute $((\mathsf{m}_c,\sigma_c),\mathsf{RED}_c) \leftarrow \mathsf{Sign}(\mathsf{sk},\mathsf{m}_c,\mathsf{ADM}_c)$ for $c \in \{0,1\}$.
  2: Let $((\mathring{\mathsf{m}}_c,\mathring{\sigma}_c),\mathring{\mathsf{RED}}_c) \leftarrow \mathsf{Redact}(\mathsf{pk},\sigma_c,\mathsf{m}_c,\mathsf{MOD}_c,\mathsf{RED}_c)$ for $c \in \{0,1\}$.
  3: If $\mathring{\mathsf{m}}_0 \neq \mathring{\mathsf{m}}_1\ \vee\ \mathring{\mathsf{ADM}}_0 \neq \mathring{\mathsf{ADM}}_1$, return $\perp$.
  4: Return $(\mathring{\mathsf{m}}_b,\mathring{\sigma}_b)$.
```

Here, the admissible modifications $\mathring{\mathsf{ADM}}_0$ *and* $\mathring{\mathsf{ADM}}_1$ *corresponding to the redacted messages are implicitly defined by (and recoverable from) the tuples* $(\mathring{\mathsf{m}}_0,\mathring{\sigma}_0)$ *and* $(\mathring{\mathsf{m}}_1,\mathring{\sigma}_1)$ *and the oracle returns* $\perp$ *if any of the algorithms returns* $\perp$.

We call an RS *secure*, if it is correct, unforgeable, and private.

Group Signatures. Subsequently, we recall the established model for static group signatures from [BMW03]. Again, we slightly adapt the notation to ours.

Definition 4. *A group signature scheme* GS *is a tuple* $(\mathsf{KeyGen},\mathsf{Sign},\mathsf{Verify},\mathsf{Open})$ *of PPT algorithms which are defined as follows:*

$\mathsf{KeyGen}(1^\kappa,n)$: *Takes a security parameter* κ *and the group size* n *as input. It generates and outputs a group verification key* gpk*, a group opening key* gok*, as well as a list of group signing keys* $\mathsf{gsk}=\{\mathsf{gsk}_i\}_{i\in[n]}$.

$\mathsf{Sign}(\mathsf{gsk}_i,\mathsf{m})$: *Takes a group signing key* gsk_i *and a message* m *as input and outputs a signature* σ.

$\mathsf{Verify}(\mathsf{gpk},\mathsf{m},\sigma)$: *Takes a group verification key* gpk*, a message* m *and a signature* σ *as input, and outputs a bit* b.

$\mathsf{Open}(\mathsf{gok},\mathsf{m},\sigma)$: *Takes a group opening key* gok*, a message* m *and a signature* σ *as input, and outputs an identity* i.

The GS security properties are formally defined as follows (we omit correctness).

Definition 5 (Anonymity). *A* GS *is anonymous, if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l}(\mathsf{gpk},\mathsf{gok},\mathsf{gsk}) \leftarrow \mathsf{KeyGen}(1^\kappa, n),\\ b \xleftarrow{R} \{0,1\},\ \mathcal{O} \leftarrow \{\mathsf{Open}(\mathsf{gok},\cdot,\cdot)\},\\ (i_0^\star, i_1^\star, \mathsf{m}^\star, \mathsf{st}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk},\mathsf{gsk}),\\ \sigma \leftarrow \mathsf{Sign}(\mathsf{gsk}_{i_b^\star}, \mathsf{m}^\star),\ b^\star \leftarrow \mathcal{A}^{\mathcal{O}}(\sigma,\mathsf{st})\end{array} : \begin{array}{r} b = b^\star\ \wedge\\ (\mathsf{m}^\star,\sigma) \notin \mathcal{Q}_2^{\mathsf{Open}}\end{array}\right] \leq \varepsilon(\kappa),$$

where $\mathcal{A}$ *runs in two stages and* $\mathcal{Q}_2^{\mathsf{Open}}$ *records the* Open *queries in stage two.*

Definition 6 (Traceability). *A* GS *is traceable, if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l}(\mathsf{gpk},\mathsf{gok},\mathsf{gsk}) \leftarrow \mathsf{KeyGen}(1^\kappa, n),\\ \mathcal{O} \leftarrow \{\mathsf{Sig}(\cdot,\cdot),\mathsf{Key}(\cdot)\},\\ (\mathsf{m}^\star,\sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk},\mathsf{gok}),\\ i \leftarrow \mathsf{Open}(\mathsf{gok},\mathsf{m}^\star,\sigma^\star)\end{array} : \begin{array}{r}\mathsf{Verify}(\mathsf{gpk},\mathsf{m}^\star,\sigma^\star) = 1\ \wedge\\ (i = \perp\ \vee\ (i \notin \mathcal{Q}^{\mathsf{Key}}\ \wedge\\ (i,\mathsf{m}^\star) \notin \mathcal{Q}^{\mathsf{Sig}}))\end{array}\right] \leq \varepsilon(\kappa),$$

where $\mathsf{Sig}(i,\mathsf{m})$ *returns* $\mathsf{Sign}(\mathsf{gsk}_i,\mathsf{m})$*,* $\mathsf{Key}(i)$ *returns* gsk_i*, and* $\mathcal{Q}^{\mathsf{Sig}}$ *and* $\mathcal{Q}^{\mathsf{Key}}$ *record the queries to the signing and key oracle respectively.*

We call a GS *secure*, if it is correct, anonymous and traceable.

3 Security Model

Now we formally define signer-anonymous designated-verifier redactable signature schemes (AD-RS). To obtain the most general result, we follow [DPSS15] and do not make the structure of the messages to be signed explicit. Inspired by [MPV09], we view signatures output by Sign as being of the form $\sigma = (\underline{\sigma}, \overline{\sigma})$. That is, signatures are composed of a public signature component $\underline{\sigma}$ and a private signature component $\overline{\sigma}$, where $\underline{\sigma}$ may also be empty. For the sake of simple presentation we model our system for static groups, since an extension to dynamic groups [BSZ05] is straight forward.

Definition 7 (AD-RS). *An* AD-RS *is a tuple* (Setup*,* DVGen*,* Sign*,* GVerify*,* Open*,* Redact*,* Verify*,* Sim) *of PPT algorithms, which are defined as follows.*

$\mathsf{Setup}(1^\kappa, n)$: *Takes a security parameter* κ *and the group size* n *as input. It generates and outputs a group public key* gpk*, a group opening key* gok*, and a list of group signing keys* $\mathsf{gsk} = \{\mathsf{gsk}_i\}_{i\in[n]}$*.*

$\mathsf{DVGen}(1^\kappa)$: *Takes a security parameter* κ *as input and outputs a designated verifier key pair* $(\mathsf{vsk}_j, \mathsf{vpk}_j)$*.*

$\mathsf{Sign}(\mathsf{gsk}_i, \mathsf{m}, \mathsf{ADM})$: *Takes a group signing key* gsk_i*, a message* m*, and admissible modifications* ADM *as input, and outputs a signature* σ*.*

$\mathsf{GVerify}(\mathsf{gpk}, \mathsf{m}, \sigma)$: *Takes a group public key* gpk*, a message* m*, and a signature* σ *as input, and outputs a bit* b*.*

$\mathsf{Open}(\mathsf{gok}, \mathsf{m}, \sigma)$: *Takes a group opening key* gok*, a message* m*, and a valid signature* σ *as input, and outputs an identity* i*.*

$\mathsf{Redact}(\mathsf{gpk}, \mathsf{vpk}_j, \mathsf{m}, \sigma, \textsf{MOD})$: *Takes a group public key* gpk*, a designated-verifier public key* vpk_j*, a message* m*, a valid signature* σ*, and modification instructions* $\textsf{MOD}$ *as input, and returns a designated-verifier message-signature pair* $(\mathring{\mathsf{m}}, \rho)$*.*

$\mathsf{Verify}(\mathsf{gpk}, \mathsf{vpk}_j, \mathsf{m}, \rho)$: *Takes a group public key* gpk*, a designated-verifier public key* vpk_j*, a message* m*, and a designated-verifier signature* ρ*. It returns a bit* b*.*

$\mathsf{Sim}(\mathsf{gpk}, \mathsf{vsk}_j, \mathsf{m}, \textsf{ADM}, \textsf{MOD}, \underline{\sigma})$**:** *Takes a group public key* gpk*, a designated-verifier secret key* vsk_j*, a message* m*, admissible modifications* $\textsf{ADM}$*, modification instructions* $\textsf{MOD}$*, and a valid public signature component* $\underline{\sigma}$ *as input and outputs a designated-verifier message signature pair* $(\mathring{\mathsf{m}}, \rho)$*.*

Oracles. We base our security notions on the following oracles and assume that $(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk})$ generated in the experiments are implicitly available to them. The environment stores a list $\mathtt{DVK}$ of designated-verifier key pairs, and a set of public signature components $\mathtt{SIG}$. Each list entry and each set is initially set to $\perp$.

$\mathsf{Key}(i)$: This oracle returns gsk_i.

$\mathsf{DVGen}(j)$: If $\mathtt{DVK}[j] \neq \perp$ this oracle returns $\perp$. Otherwise, it runs $(\mathsf{vsk}_j, \mathsf{vpk}_j) \leftarrow \mathsf{DVGen}(1^\kappa)$, sets $\mathtt{DVK}[j] \leftarrow (\mathsf{vsk}_j, \mathsf{vpk}_j)$, and returns vpk_j.

$\mathsf{DVKey}(j)$: This oracle returns vsk_j.

$\mathsf{Sig}(i, \mathsf{m}, \textsf{ADM})$: This oracle runs $\sigma = (\underline{\sigma}, \overline{\sigma}) \leftarrow \mathsf{Sign}(\mathsf{gsk}_i, \mathsf{m}, \textsf{ADM})$, sets $\mathtt{SIG} \leftarrow \mathtt{SIG} \cup \{\underline{\sigma}\}$ and returns σ.

$\mathsf{Open}(\mathsf{m}, \sigma)$: This oracle runs $i \leftarrow \mathsf{Open}(\mathsf{gok}, \mathsf{m}, \sigma)$ and returns i.

$\mathsf{Sim}(j, \mathsf{m}, \textsf{ADM}, \textsf{MOD}, \underline{\sigma})$: If $\underline{\sigma} \notin \mathtt{SIG}$, this oracle returns $\perp$. Otherwise, it runs $(\mathring{\mathsf{m}}, \rho) \leftarrow \mathsf{Sim}(\mathsf{gpk}, \mathsf{vsk}_j, \mathsf{m}, \textsf{ADM}, \textsf{MOD}, \underline{\sigma})$ and returns $(\mathring{\mathsf{m}}, \rho)$.

$\mathsf{RoS}(b, j, \mathsf{m}, \textsf{ADM}, \textsf{MOD}, \sigma)$: If $b = 0$, this oracle runs $(\mathring{\mathsf{m}}, \rho) \leftarrow \mathsf{Redact}(\mathsf{gpk}, \mathsf{vpk}_j, \mathsf{m}, \sigma, \textsf{MOD})$ and returns $(\mathring{\mathsf{m}}, \rho)$. Otherwise, it uses the Sim oracle to obtain $(\mathring{\mathsf{m}}, \rho) \leftarrow \mathsf{Sim}(j, \mathsf{m}, \textsf{ADM}, \textsf{MOD}, \underline{\sigma})$ and returns $(\mathring{\mathsf{m}}, \rho)$.

$\mathsf{Ch}(i, j, (\mathsf{m}_0, \textsf{ADM}_0, \textsf{MOD}_0), (\mathsf{m}_1, \textsf{ADM}_1, \textsf{MOD}_1), b)$: This oracle runs $\sigma_c \leftarrow \mathsf{Sign}(\mathsf{gsk}_i, \mathsf{m}_c, \textsf{ADM}_c)$, $(\mathring{\mathsf{m}}_c, \rho_c) \leftarrow \mathsf{Redact}(\mathsf{vpk}_j, \mathsf{m}_c, \sigma_c, \textsf{MOD}_c)$, for $c \in \{0, 1\}$. If $\mathring{\mathsf{m}}_0 \neq \mathring{\mathsf{m}}_1 \;\vee\; \mathring{\textsf{ADM}}_0 \neq \mathring{\textsf{ADM}}_1$, it returns $\perp$ and $(\mathring{\mathsf{m}}_b, \underline{\sigma}_b, \rho_b)$ otherwise.[4]

The environment stores the oracle queries in lists. In analogy to the oracle labels, we use $\mathcal{Q}^{\mathsf{Key}}$, $\mathcal{Q}^{\mathsf{DVGen}}$, $\mathcal{Q}^{\mathsf{DVKey}}$, $\mathcal{Q}^{\mathsf{Sig}}$, $\mathcal{Q}^{\mathsf{Open}}$, $\mathcal{Q}^{\mathsf{Sim}}$, $\mathcal{Q}^{\mathsf{RoS}}$, and $\mathcal{Q}^{\mathsf{Ch}}$ to denote them.

Security Notions. We require AD-RS to be correct, group unforgeable, designated-verifier unforgeable, simulatable, signer anonymous, and private.

Correctness guarantees that all honestly computed signatures verify correctly.

Formally, we require that for all $\kappa \in \mathbb{N}$, for all $n \in \mathbb{N}$, for all $(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}) \leftarrow \mathsf{Setup}(1^\kappa, n)$, for all $(\mathsf{vsk}_j, \mathsf{vpk}_j) \leftarrow \mathsf{DVGen}(1^\kappa)$, for all $(\mathsf{vsk}_\ell, \mathsf{vpk}_\ell) \leftarrow \mathsf{DVGen}(1^\kappa)$, for all $(\mathsf{m}, \textsf{ADM}, \textsf{MOD})$ where $\textsf{MOD} \preceq \textsf{ADM} \;\wedge\; \textsf{ADM} \preceq \mathsf{m}$, for all $(\mathsf{m}', \textsf{ADM}', \textsf{MOD}')$ where $\textsf{MOD}' \preceq \textsf{ADM}' \;\wedge\; \textsf{ADM}' \preceq \mathsf{m}'$ for all $i \in [n]$, for all $\sigma = (\underline{\sigma}, \overline{\sigma}) \leftarrow \mathsf{Sign}(\mathsf{gsk}_i, \mathsf{m}, \textsf{ADM})$,

[4] Here $\mathring{\textsf{ADM}}_0$ and $\mathring{\textsf{ADM}}_1$ are derived from $\textsf{ADM}_0$ and $\textsf{ADM}_1$ with respect to $\textsf{MOD}_0$ and $\textsf{MOD}_1$.

for all $u \leftarrow \mathsf{Open}(\mathsf{gok}, \mathsf{m}, \sigma)$, for all $(\mathring{\mathsf{m}}, \rho) \leftarrow \mathsf{Redact}(\mathsf{gpk}, \mathsf{vpk}_j, \mathsf{m}, \sigma, \textsc{mod})$, for all $(\mathring{\mathsf{m}}', \rho') \leftarrow \mathsf{Sim}(\mathsf{gpk}, \mathsf{vsk}_\ell, \mathsf{m}', \textsc{adm}', \textsc{mod}', \underline{\sigma})$, it holds with overwhelming probability that $\mathsf{GVerify}(\mathsf{gpk}, \mathsf{m}, \sigma) = 1 \;\wedge\; i = u \;\wedge\; \mathsf{Verify}(\mathsf{gpk}, \mathsf{vpk}_j, \mathring{\mathsf{m}}, \rho) = 1 \;\wedge\; \mathsf{Verify}(\mathsf{gpk}, \mathsf{vpk}_\ell, \mathring{\mathsf{m}}', \rho') = 1$ and that $\mathring{\mathsf{m}} \xleftarrow{\textsc{mod}} \mathsf{m} \;\wedge\; \mathring{\mathsf{m}}' \xleftarrow{\textsc{mod}'} \mathsf{m}'$.

Group unforgeability captures the intuition that the only way of obtaining valid signatures on messages is by applying "allowed" modifications to messages which were initially signed by a group member. Moreover, this property guarantees that every valid signature can be linked to the original signer by some authority.

Technically, the definition captures the traceability property of group signatures while simultaneously taking the malleability of RS into account.

Definition 8. *An* AD-RS *is group unforgeable, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that*

$$\Pr\left[\begin{array}{l}(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}) \leftarrow \mathsf{Setup}(1^\kappa, n),\\ \mathcal{O} \leftarrow \{\mathsf{Sig}(\cdot,\cdot,\cdot), \mathsf{Key}(\cdot)\},\\ (\mathsf{m}^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk}, \mathsf{gok}),\\ u \leftarrow \mathsf{Open}(\mathsf{gok}, \mathsf{m}^\star, \sigma^\star)\end{array} : \begin{array}{r}\mathsf{GVerify}(\mathsf{gpk}, \mathsf{m}^\star, \sigma^\star) = 1 \,\wedge\\ (u = \bot \;\vee\; (u \notin \mathcal{Q}^{\mathsf{Key}} \,\wedge\\ \nexists (u, \mathsf{m}, \textsc{adm}) \in \mathcal{Q}^{\mathsf{Sig}} : \mathsf{m}^\star \overset{\textsc{adm}}{\preceq} \mathsf{m}))\end{array}\right] \leq \varepsilon(\kappa).$$

Designated-verifier unforgeability models the requirement that a designated-verifier signature can only be obtained in two ways: either by corretly redacting a signature (which can be done by everybody having access to the latter), or by having access to the secret key of the designated verifier. The former option would be chosen whenever a signature is to be legitimately forwarded to a receiver, while the latter enables the designated verifier to fake signatures.

Together with the previous definition, designated-verifier unforgeability guarantees that no adversary can come up with a designated-verifier signature for a foreign public key: by Definition 8 it is infeasible to forge a signature—and Definition 9 states that the only way of generating a designated-verifier signature for somebody else is to know a valid signature to start from.

Definition 9. *An* AD-RS *is designated-verifier unforgeable, if there exists a PPT opener $\mathsf{O} = (O_1, O_2)$ such that for every PPT adversary $\mathcal{A}$ there is a negligible function $\varepsilon_1(\cdot)$ such that*

$$\left|\begin{array}{l}\Pr\left[(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}) \leftarrow \mathsf{Setup}(1^\kappa, n) : \mathcal{A}(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}) = 1\right] \;-\\ \Pr\left[(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}, \tau) \leftarrow O_1(1^\kappa, n) : \mathcal{A}(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}) = 1\right]\end{array}\right| \leq \varepsilon_1(\kappa),$$

and for every PPT adversary $\mathcal{A}$ there is a negligible function $\varepsilon_2(\cdot)$ such that

$$\Pr\left[\begin{array}{l}(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}, \tau) \leftarrow O_1(1^\kappa, n),\\ \mathcal{O} \leftarrow \{\mathsf{Sig}(\cdot,\cdot,\cdot), \mathsf{Key}(\cdot),\\ \mathsf{DVGen}(\cdot), \mathsf{DVKey}(\cdot),\\ \mathsf{Sim}(\cdot,\cdot,\cdot,\cdot,\cdot)\},\\ (\mathsf{m}^\star, \rho^\star, v^\star) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk}, \mathsf{gok}),\\ u \leftarrow O_2(\tau, \mathtt{DVK}, \mathsf{m}^\star, \rho^\star, v^\star)\end{array} : \begin{array}{r}\mathsf{Verify}(\mathsf{gpk}, \mathsf{vpk}_{v^\star}, \mathsf{m}^\star, \rho^\star) = 1 \,\wedge\\ v^\star \notin \mathcal{Q}^{\mathsf{DVKey}} \,\wedge\\ \wedge\; (u = \bot \;\vee\; (u \notin \mathcal{Q}^{\mathsf{Key}} \,\wedge\\ \nexists (u, \mathsf{m}, \textsc{adm}) \in \mathcal{Q}^{\mathsf{Sig}} : \mathsf{m}^\star \overset{\textsc{adm}}{\preceq} \mathsf{m})) \,\wedge\\ \nexists (v^\star, \mathsf{m}, \textsc{adm}, \cdot, \cdot) \in \mathcal{Q}^{\mathsf{Sim}} : \mathsf{m}^\star \overset{\textsc{adm}}{\preceq} \mathsf{m})\end{array}\right] \leq \varepsilon_2(\kappa).$$

In our definition, we assume a simple key registration for designated verifiers to ensure that all designated-verifier key pairs have been honestly created and thus an adversary is not able to mount rogue key attacks. In practice, this requirement can often be alleviated by introducing an option to check the honest generation of the keys (cf. [RY07]), which we omit for simplicity.

Simulatability captures that designated verifiers can simulate signatures on arbitrary messages which are indistinguishable from honestly computed signatures.

Definition 10. *An* AD-RS *satisfies the simulatability property, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that it holds that*

$$\Pr\left[\begin{array}{l} (\mathsf{gpk},\mathsf{gok},\mathsf{gsk}) \leftarrow \mathsf{Setup}(1^\kappa, n),\ \ b \xleftarrow{R} \{0,1\}, \\ \mathcal{O} \leftarrow \{\mathsf{DVGen}(\cdot), \mathsf{DVKey}(\cdot)\}, \\ ((\mathsf{m}_0, \mathsf{ADM}_0, \mathsf{MOD}_0), (\mathsf{m}_1, \mathsf{ADM}_1), \\ i^\star, j^\star, \mathsf{st}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk},\mathsf{gok},\mathsf{gsk}), \\ \sigma = (\underline{\sigma}, \overline{\sigma}) \leftarrow \mathsf{Sign}(\mathsf{gsk}_{i^\star}, \mathsf{m}_b, \mathsf{ADM}_b), \\ (\mathring{\mathsf{m}}_0, \rho) \leftarrow \mathsf{RoS}(b, j^\star, \mathsf{m}_0, \mathsf{ADM}_0, \mathsf{MOD}_0, \sigma), \\ b^\star \leftarrow \mathcal{A}^{\mathcal{O}}(\underline{\sigma}, \mathring{\mathsf{m}}_0, \rho, \mathsf{st}) \end{array} : \begin{array}{r} b = b^\star\ \wedge \\ \mathsf{ADM}_0 \preceq \mathsf{m}_0\ \wedge \\ \mathsf{ADM}_1 \preceq \mathsf{m}_1 \end{array}\right] \leq 1/2 + \epsilon(\kappa).$$

As mentioned earlier, we assume that signatures consist of a private and a public component (the latter being denoted by $\underline{\sigma}$). To eliminate potential privacy issues associated with a public $\underline{\sigma}$, we also give $\underline{\sigma}$ as input to the simulator and the adversary, and require that the adversary cannot tell real and faked signatures apart *even when knowing* $\underline{\sigma}$. This way, our definitional framework guarantees that these parts do not contain any sensitive information.

In a realization of the system, the public parts of all signatures issued by the hospital would be made publicly available (without further meta-information).

Signer anonymity requires that only the opening authority can determine the identity of a signer.

Definition 11. *An* AD-RS *is signer anonymous, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that*

$$\Pr\left[\begin{array}{l} (\mathsf{gpk},\mathsf{gok},\mathsf{gsk}) \leftarrow \mathsf{Setup}(1^\kappa, n), \\ b \xleftarrow{R} \{0,1\}, \mathcal{O} \leftarrow \{\mathsf{Open}(\cdot,\cdot)\}, \\ (i_0^\star, i_1^\star, \mathsf{m}^\star, \mathsf{ADM}^\star, \mathsf{st}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk},\mathsf{gsk}), \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{gsk}_{i_b^\star}, \mathsf{m}^\star, \mathsf{ADM}^\star), \\ b^\star \leftarrow \mathcal{A}^{\mathcal{O}}(\sigma, \mathsf{st}) \end{array} : \begin{array}{r} b = b^\star\ \wedge \\ \nexists(\mathsf{m}, (\underline{\sigma}, \cdot)) \in \mathcal{Q}_2^{\mathsf{Open}} : \\ \mathsf{m} \overset{\mathsf{ADM}}{\preceq} \mathsf{m}^\star \end{array}\right] \leq 1/2 + \varepsilon(\kappa),$$

and $\mathcal{A}$ runs in two stages and $\mathcal{Q}_2^{\mathsf{Open}}$ records queries to oracle Open *in stage two.*

The definition guarantees that—no matter how many signatures already have been opened—the signers' identities for all other signatures remain secret. The formulation is, up to the last clause of the winning condition, similar to the anonymity definition of group signature schemes (cf. Definition 5). We, however, need to adapt the last clause because Definition 5 requires signatures to be non-malleable. In contrast, our signatures are malleable by definition. However, we can still require parts of the signature, and in particular the public part, to be non-malleable. By doing so, we can achieve a strong notion that resembles anonymity in the sense of group signatures whenever honestly generated signatures have different public components with overwhelming probability. This is in particular the case for our instantiations provided in the next sections.

Privacy guarantees that a redacted designated-verifier signature does not leak anything about the blacked-out parts of the original message.

Definition 12. *An* AD-RS *is private, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that*

$$\Pr\left[\begin{array}{l}(\mathsf{gpk},\mathsf{gok},\mathsf{gsk}) \leftarrow \mathsf{Setup}(1^\kappa, n),\ b \xleftarrow{R} \{0,1\},\\ \mathcal{O} \leftarrow \{\mathsf{Sig}(\cdot,\cdot,\cdot), \mathsf{Ch}(\cdot,\cdot,\cdot,\cdot,b)\},\\ b^\star \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk},\mathsf{gok},\mathsf{gsk})\end{array} : b = b^\star\right] \leq 1/2 + \varepsilon(\kappa).$$

We call an AD-RS *secure*, if it is correct, group unforgeable, designated-verifier unforgeable, simulatable, signer anonymous, and private.

Group Redactable Signatures. When omitting the DV-related notions and oracles, one directly obtains a definition of group redactable signatures, which may also be useful for applications that require revocable signer-anonymity.

4 A Generic Construction

Now we present a simple generic construction which can be built by combining any GS, any RS, and any Π that admits proofs of knowledge in a black-box way. In Scheme 1 we present our construction which follows the intuition given in the introduction. We use Π to prove knowledge of a witness for the following **NP** relation R required by the verification of designated-verifier signatures.

$$\begin{aligned}((\mathsf{m}, \mathsf{pk}, \mathsf{vpk}_j),\ (\sigma_{\mathsf{R}}, \sigma_{\mathsf{V}})) \in R \iff \\ \mathsf{RS}.\mathsf{Verify}(\mathsf{pk}, \mathsf{m}, \sigma_{\mathsf{R}}) = 1 \ \vee\ \Sigma.\mathsf{Verify}(\ vpk_j, \mathsf{m}, \sigma_{\mathsf{V}}) = 1.\end{aligned}$$

The rationale behind choosing R in this way is that this yields the most general result. That is, no further assumptions on RS or Σ are required.

Theorem 1 (proven in Appendix B). *If* GS*,* RS*, and* Σ *are secure and* Π *is witness indistinguishable and admits proofs of knowledge, then Scheme 1 is secure.*

For an instantiation of our construction we can use standard GS and standard RS, where multiple practically efficient instantiations exist. Thus, the time required for signature creation/verification is mainly determined by the cost of the proof of knowledge of the RS signature σ_{R}. We, however, want to emphasize that—depending on the concrete RS—this proof can usually be instantiated by means of relatively cheap Σ-protocols. Ultimately, as we will show below, we can replace this proof with a much cheaper proof by exploiting properties of the used RS.

$\mathsf{Setup}(1^\kappa, n)$: Run $(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}) \leftarrow \mathsf{GS.KeyGen}(1^\kappa, n)$, $\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$, set $\mathsf{gpk}' \leftarrow (\mathsf{gpk}, \mathsf{crs})$ and return $(\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk})$.

$\mathsf{DVGen}(1^\kappa)$: Run $(\mathsf{vsk}_j, \mathsf{vpk}_j) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$ and return $(\mathsf{vsk}_j, \mathsf{vpk}_j)$.

$\mathsf{Sign}(\mathsf{gsk}_i, \mathsf{m}, \textsc{adm})$: Run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{RS.KeyGen}(1^\kappa)$ and return $\sigma = (\underline{\sigma}, \overline{\sigma}) \leftarrow ((\mathsf{pk}, \sigma_\mathsf{G}), (\sigma_\mathsf{R}, \textsc{red}))$, with

$$\sigma_\mathsf{G} \leftarrow \mathsf{GS.Sign}(\mathsf{gsk}_i, \mathsf{pk}), \text{ and } ((\mathsf{m}, \sigma_\mathsf{R}), \textsc{red}) \leftarrow \mathsf{RS.Sign}(\mathsf{sk}, \mathsf{m}, \textsc{adm}).$$

$\mathsf{GVerify}(\mathsf{gpk}, \mathsf{m}, \sigma)$: Parse σ as $((\mathsf{pk}, \sigma_\mathsf{G}), (\sigma_\mathsf{R}, \cdot))$ and return 1 if the following holds and 0 otherwise:

$$\mathsf{GS.Verify}(\mathsf{gpk}, \mathsf{pk}, \sigma_\mathsf{G}) = 1 \quad \wedge \quad \mathsf{RS.Verify}(\mathsf{pk}, \mathsf{m}, \sigma_\mathsf{R}) = 1.$$

$\mathsf{Open}(\mathsf{gok}, \mathsf{m}, \sigma)$: Parse σ as $((\mathsf{pk}, \sigma_\mathsf{G}), \overline{\sigma})$ and return $\mathsf{GS.Open}(\mathsf{gok}, \mathsf{pk}, \sigma_\mathsf{G})$.

$\mathsf{Redact}(\mathsf{gpk}, \mathsf{vpk}_j, \mathsf{m}, \sigma, \textsc{mod})$: Parse σ as $((\mathsf{pk}, \sigma_\mathsf{G}), (\sigma_\mathsf{R}, \textsc{red}))$ and return $(\mathring{\mathsf{m}}, \rho)$, where

$$((\mathring{\mathsf{m}}, \mathring{\sigma}_\mathsf{R}), \cdot) \leftarrow \mathsf{RS.Redact}(\mathsf{pk}, \mathsf{m}, \sigma_\mathsf{R}, \textsc{mod}, \textsc{red}),$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathring{\mathsf{m}}, \mathsf{pk}, \mathsf{vpk}_j), (\mathring{\sigma}_\mathsf{R}, \perp)), \text{ and}$$
$$\rho \leftarrow ((\mathsf{pk}, \sigma_\mathsf{G}), \pi).$$

$\mathsf{Verify}(\mathsf{gpk}, \mathsf{vpk}_j, \mathsf{m}, \rho)$: Parse ρ as $((\mathsf{pk}, \sigma_\mathsf{G}), \pi)$ and return 1 if the following holds, and 0 otherwise:

$$\mathsf{GS.Verify}(\mathsf{gpk}, \mathsf{pk}, \sigma_\mathsf{G}) = 1 \quad \wedge \quad \Pi.\mathsf{Verify}(\mathsf{crs}, (\mathsf{m}, \mathsf{pk}, \mathsf{vpk}_j), \pi) = 1.$$

$\mathsf{Sim}(\mathsf{gpk}, \mathsf{vsk}_j, \mathsf{m}, \textsc{adm}, \textsc{mod}, \underline{\sigma})$: If $\textsc{mod} \preceq \textsc{adm} \wedge \textsc{adm} \preceq \mathsf{m}$, parse $\underline{\sigma}$ as $(\mathsf{pk}, \sigma_\mathsf{G})$, run $\mathring{\mathsf{m}} \xleftarrow{\textsc{mod}} \mathsf{m}$, and return $(\mathring{\mathsf{m}}, \rho)$, where

$$\sigma_\mathsf{V} \leftarrow \Sigma.\mathsf{Sign}(\mathsf{vsk}_j, \mathring{\mathsf{m}}),$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathring{\mathsf{m}}, \mathsf{pk}, \mathsf{vpk}_j), (\perp, \sigma_\mathsf{V})), \text{ and}$$
$$\rho \leftarrow (\underline{\sigma}, \pi).$$

Otherwise, return $\perp$.

Scheme 1: Black-Box AD-RS

5 Boosting Efficiency via Key-Homomorphisms

In [DPSS15] it is shown that RS can be generically constructed from any EUF-CMA secure signature scheme and indistinguishable accumulators [DHS15]. In our setting it is most reasonable to consider messages as an (ordered) sequence of message blocks. A straight forward solution would thus be to build upon [DPSS15, Scheme 2], which is tailored to signing ordered sequences of messages $\mathsf{m} = (m_1, \ldots, m_n)$. Unfortunately, this construction aims to conceal the number of message blocks in the original message, and the positions of the redactions. This can be dangerous in our setting, since it might allow to completely change the document semantics. Besides that, it inherently requires a more complex construction.

To this end, we pursue a different direction and require another message representation: we make the position i of the message blocks m_i in the message explicit and represent messages as sets $\mathsf{m} = \{1||m_1, \ldots, n||m_n\}$. Besides solving the aforementioned issues, it also allows us to build upon the (simpler) RS paradigm for sets [DPSS15, Scheme 1]. This paradigm subsumes the essence of many existing RSs and works as follows. Secret keys, public keys, and signatures are split into two parts each. One corresponds to the signature scheme Σ, and one corresponds to the accumulator Λ. Then, Λ is used to encode the message, whereas Σ is used to sign the encoded message. Consequently, we can look at RS key pairs and signatures as being of the form $(\mathsf{sk}, \mathsf{pk}) = ((\mathsf{sk}_\Sigma, \mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda), (\mathsf{pk}_\Sigma, \mathsf{pk}_\Lambda))$ and $\sigma_\mathsf{R} = (\sigma_\Sigma, \sigma_\Lambda)$ where the indexes denote their respective types. We emphasize that for accumulators it holds by definition that sk_Λ is an optional trapdoor which may enable more efficient computations, but all algorithms also run without sk_Λ and the output distribution of the algorithms does not depend on whether the algorithms are executed with or without sk_Λ [DHS15, DPSS15]. We require this property to be able to create designated verifier signatures (cf. Sim) and use $(\mathsf{sk}_\Sigma, \perp, \mathsf{pk}_\Lambda)$ to denote an RS secret key without sk_Λ.

RS following this paradigm only require Σ (besides correctness) to be $\mathsf{EUF\text{-}CMA}$ secure. We observe that additional constraints on Σ—and in particular the key-homomorphism as we define it below—does not influence RS security, while it enables us to design the relation R such that it admits very efficient proofs.

Key-Homomorphic Signatures. Informally, we require signature schemes where, for a given public key and a valid signature under that key, one can adapt the public key and the signature so that the resulting signature is valid with respect to the initial message under the new public key. Moreover, adapted signatures need to be identically distributed as fresh signatures under the secret key corresponding to the adapted public key.

Key-malleability in the sense of adapting given signatures to other signatures under related keys has so far mainly been studied in context of related-key attacks (RKAs) [BCM11], where one aims to rule out such constructions. Signatures with re-randomizable keys which allow to consistently update secret and public keys, but without considering adaption of existing signatures, have recently been introduced and studied in [FKM$^+$16]. As we are not aware of any constructive use of and definitions for the functionality we require, we define key-homomorphic signatures inspired by key-homomorphic symmetric encryption (cf. [AHI11]).

Let $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be an $\mathsf{EUF\text{-}CMA}$ secure signature scheme where the secret and public keys live in groups $(\mathbb{H}, +)$ and $(\mathbb{G}, \cdot)$, respectively. Inspired by the definition for encryption schemes in [TW14], we define the following.

Definition 13 (Secret-Key to Public-Key Homomorphism). *A signature scheme Σ provides a secret-key to public-key homomorphism, if there exists an efficiently computable map $\mu : \mathbb{H} \rightarrow \mathbb{G}$ such that for all $\mathsf{sk}, \mathsf{sk}' \in \mathbb{H}$ it holds that $\mu(\mathsf{sk} + \mathsf{sk}') = \mu(\mathsf{sk}) \cdot \mu(\mathsf{sk}')$, and for all $(\mathsf{sk}, \mathsf{pk})$ output by KeyGen, it holds that $\mathsf{pk} = \mu(\mathsf{sk})$.*

Now, we define key-homomorphic signatures, where we focus on the class of functions Φ^+ representing linear shifts. We stress that Φ^+ is a finite set of functions, all with the same domain and range, and, in our case depends on the public key of the signature scheme (which is not made explicit). Moreover, Φ^+ admits an efficient membership test and its functions are efficiently computable.

Definition 14 (Φ^+-Key-Homomorphic Signatures). *A signature scheme is called Φ^+-key-homomorphic, if it provides a secret-key to public-key homomorphism and an additional PPT algorithm* Adapt*, defined as:*

$\mathsf{Adapt}(\mathsf{pk}, \mathsf{m}, \sigma, \Delta)$: *Takes a public key* pk*, a message* m*, a signature* σ*, and a function* $\Delta \in \Phi^+$ *as input, and outputs a public key* pk' *and a signature* σ'*,*

where for all $\Delta \in \Phi^+$*, all* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$*, all messages* m*, all* $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{m})$*, all* $(\mathsf{pk}', \sigma') \leftarrow \mathsf{Adapt}(\mathsf{pk}, \mathsf{m}, \sigma, \Delta)$ *it holds that* $\mathsf{Verify}(\mathsf{pk}', \mathsf{m}, \sigma') = 1$ *and* $\mathsf{pk}' = \Delta(\mathsf{pk})$*.*

For simplicity we sometimes identify a function $\Delta \in \Phi^+$ with its "shift amount" $\Delta \in \mathbb{H}$. To model that freshly generated signatures look identical as adapted signatures on the same message, we introduce the following additional property.

Definition 15 (Adaptability of Signatures). *A Φ^+-key-homomorphic signature scheme provides adaptability of signatures, if for every $\kappa \in \mathbb{N}$, and every message* m*, it holds that* $\mathsf{Adapt}(\mathsf{pk}, \mathsf{m}, \mathsf{Sign}(\mathsf{sk}, \mathsf{m}), \Delta)$ *and* $(\mathsf{pk} \cdot \mu(\Delta), \mathsf{Sign}(\mathsf{sk} + \Delta, \mathsf{m}))$ *as well as* $(\mathsf{sk}, \mathsf{pk})$ *and* $(\mathsf{sk}', \mu(\mathsf{sk}'))$ *are identically distributed, where* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$*,* $\mathsf{sk}' \xleftarrow{R} \mathbb{H}$*, and* $\Delta \xleftarrow{R} \Phi^+$*.*

For an in-depth treatment and examples of key-homomorphic signatures, we refer the reader to a more recent work [DS16b]. The important bottom-line here is that there are various efficient schemes that satisfy Definition 15. For instance, Schnorr signatures [Sch91], BLS signatures [BLS04], the recent scheme by Pointcheval and Sanders [PS16] or Waters signatures [Wat05].

Φ^+-Key-Homomorphic Redactable Signature Schemes. When instantiating the RS construction paradigm from [DPSS15] (as outlined above) with a Φ^+-key-homomorphic signature scheme, the key homomorphism of the signature scheme straight-forwardly carries over to the RS and we can define Adapt as follows.

$\mathsf{Adapt}(\mathsf{pk}, \mathsf{m}, \sigma, \Delta)$: Parse pk as $(\mathsf{pk}_\Sigma, \mathsf{pk}_\Lambda)$ and σ as $(\sigma_\Sigma, \sigma_\Lambda)$, run $(\mathsf{pk}'_\Sigma, \sigma'_\Sigma) \leftarrow \mathsf{Adapt}(\mathsf{pk}_\Sigma, \Lambda(\mathsf{m}), \sigma_\Sigma, \Delta)$ and return $(\mathsf{pk}', \sigma') \leftarrow ((\mathsf{pk}'_\Sigma, \mathsf{pk}_\Lambda), (\sigma'_\Sigma, \sigma_\Lambda))$.

This allows us to concisely present our construction in Scheme 2. The **NP** relation, which needs to be satisfied by valid designated-verifier signatures is as follows.

$$((\mathsf{pk}, \mathsf{vpk}_j), (\mathsf{sk}, \mathsf{vsk}_j)) \in R \iff \mathsf{pk} = \mu(\mathsf{sk}) \ \vee \ \Sigma.\mathsf{VKey}(\mathsf{vsk}_j, \mathsf{vpk}_j) = 1.$$

In the discrete logarithm setting such a proof requires an OR-Schnorr proof of two discrete logs, i.e., only requires *two group exponentiations.*

$\mathsf{Redact}(\mathsf{gpk}, \mathsf{vpk}_j, \mathsf{m}, \sigma, \textsc{mod})$: Parse σ as $((\mathsf{pk}, \sigma_\mathsf{G}), (\sigma_\mathsf{R}, \textsc{red}))$ and return $(\mathring{\mathsf{m}}, \rho)$, where

$$\mathsf{sk}' \xleftarrow{R} \mathbb{H},\ \mathsf{pk}' \leftarrow \mu(\mathsf{sk}'),\ (\mathsf{pk}_\mathsf{R}, \sigma'_\mathsf{R}) \leftarrow \mathsf{Adapt}(\mathsf{pk}, \mathsf{m}, \sigma_\mathsf{R}, \mathsf{sk}'),$$
$$((\mathring{\mathsf{m}}, \mathring{\sigma}'_\mathsf{R}), \cdot) \leftarrow \mathsf{RS.Redact}(\mathsf{pk}_\mathsf{R}, \mathsf{m}, \sigma'_\mathsf{R}, \textsc{mod}, \textsc{red}),$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}_j), (\mathsf{sk}', \bot)),\ \text{and}\ \rho \leftarrow ((\mathsf{pk}, \sigma_\mathsf{G}), \mathsf{pk}', \mathring{\sigma}'_\mathsf{R}, \pi).$$

$\mathsf{Verify}(\mathsf{gpk}, \mathsf{vpk}_j, \mathsf{m}, \rho)$: Parse ρ as $((\mathsf{pk}, \sigma_\mathsf{G}), \mathsf{pk}', \mathring{\sigma}'_\mathsf{R}, \pi)$, let $\mathsf{pk} = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Lambda)$, compute $\mathsf{pk}_\mathsf{R} \leftarrow (\mathsf{pk}_\Sigma \cdot \mathsf{pk}', \mathsf{pk}_\Lambda)$ and return 1 if the following holds, and 0 otherwise:

$$\mathsf{GS.Verify}(\mathsf{gpk}, \mathsf{pk}, \sigma_\mathsf{G}) = 1 \quad \wedge \quad \Pi.\mathsf{Verify}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}_j), \pi) = 1$$
$$\wedge \quad \mathsf{RS.Verify}(\mathsf{pk}_\mathsf{R}, \mathsf{m}, \mathring{\sigma}'_\mathsf{R}) = 1.$$

$\mathsf{Sim}(\mathsf{gpk}, \mathsf{vsk}_j, \mathsf{m}, \textsc{adm}, \textsc{mod}, \underline{\sigma})$: If $\textsc{mod} \preceq \textsc{adm}\ \wedge\ \textsc{adm} \preceq \mathsf{m}$, parse $\underline{\sigma}$ as $((\mathsf{pk}_\Sigma, \mathsf{pk}_\Lambda), \sigma_\mathsf{G})$ and return $(\mathring{\mathsf{m}}, \rho)$, where

$$\mathsf{sk}^\Sigma_\mathsf{R} \xleftarrow{R} \mathbb{H},\ \mathsf{pk}^\Sigma_\mathsf{R} \leftarrow \mu(\mathsf{sk}^\Sigma_\mathsf{R}),\ \mathsf{pk}' \leftarrow \mathsf{pk}_\Sigma^{-1} \cdot \mathsf{pk}^\Sigma_\mathsf{R},$$
$$((\mathsf{m}, \sigma'_\mathsf{R}), \textsc{red}) \leftarrow \mathsf{RS.Sign}((\mathsf{sk}^\Sigma_\mathsf{R}, \bot, \mathsf{pk}_\Lambda), \mathsf{m}, \textsc{adm}),$$
$$((\mathring{\mathsf{m}}, \mathring{\sigma}'_\mathsf{R}), \cdot) \leftarrow \mathsf{RS.Redact}((\mathsf{pk}^\Sigma_\mathsf{R}, \mathsf{pk}_\Lambda), \mathsf{m}, \sigma'_\mathsf{R}, \textsc{mod}, \textsc{red}),$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}_j), (\bot, \mathsf{vsk}_j)),\ \text{and}\ \rho \leftarrow (\underline{\sigma}, \mathsf{pk}', \mathring{\sigma}'_\mathsf{R}, \pi).$$

Scheme 2: Semi-Black-Box AD-RS where Setup, DVGen, Sign, GVerify, and Open are as in Scheme 1.

Theorem 2 (proven in Appendix C). *If* GS *is secure,* RS *is an adaptable* RS *following [DPSS15, Scheme 1],* Σ *is secure, and* Π *is witness indistinguishable and admits proofs of knowledge, then Scheme 2 is also secure.*

5.1 Performance Overview

In this section we evaluate the practical efficiency of Scheme 2. We first assess the practicality of the underlying components and then analyze the overhead imposed by the provably provided strong security guarantees.

Group Signatures. It is well known that there exist multiple practically efficient group signature schemes for non-constrained devices such as standard PCs or even more powerful machines in the cloud. Yet, to adequately protect the doctor's group signing key—which is the only key that persists over multiple signing operations—it might make sense to compute the doctor's group signature σ_G on the one-time RS public key pk upon Sign on some dedicated signature token such as a smart card or smart phone. Using the estimations in [DS16a], such a signature can be computed in ≈ 1s on an ARM Cortex-M0+, a processor that is small enough to be employed in smart cards. While this is already acceptable, the performance on smart phones will even be significantly better. For instance, [CDDT12] report execution times of approximately 150ms for the computation of a group signature with the well-known BBS [BBS04] scheme on a by now rather outdated smart phone.

Key-Homomorphic Redactable Signatures. We first note that the RS keys are freshly generated and the secret keys can be deleted after each signing operation. The respective operations can therefore be directly executed on the doctor's PC, potentially even in parallel to the computation of the group signature. Since we are not aware of any performance evaluation of RS on standard PCs, we implemented one possible instantiation of [DPSS15, Scheme 1]. In particular, we based our RS implementation on Schnorr signatures and the indistinguishable t-SDH accumulator from [DHS15] without further optimizations regarding efficiency. In Table 1, we present our performance results on an Intel Core i7-4790 @ 3.60GHz with 8GB of RAM, running Java 1.8.0_91 on top of Ubuntu 16.04. Each value represents the mean of 100 consecutive executions. These results confirm that the required RS paradigm is perfectly suited for our application.

Sign	Verify	Redact	Verify (after Redact)
73.1ms	886.3ms	0.1ms	450.3ms

Table 1. RS timings in milliseconds, with a number of $n = 100$ message blocks, 50% admissibly redactable blocks and 25% of the blocks being redacted upon Redact.

Additional Computations. Using Schnorr signatures, one only needs two group exponentiations for the proof of knowledge; the adaption of the signature only requires a $\mathbb{Z}_p$ operation, which, compared to the group exponentiations, can be neglected. All in all, the additional computations can thus be ignored compared to those of GS and RS[5], even on very constrained devices such as [UW14].

Signature Size. Regarding signature size, the dominant part is the size of the RS public key and signature, respectively, which is in turn determined by the choice of the accumulator. In particular, when instantiating the RS with an accumulator having constant key size and supporting batch verification, one can even obtain constant size signatures. We refer the reader to [DPSS15] for a discussion on RS signature sizes and [DHS15] for an overview of suitable accumulators.

6 Conclusion

We introduce the notion of signer-anonymous designated-verifier redactable signatures, extending redactable signatures in their vanilla form in several important directions. These additional features are motivated by a real world use-case in the health care field, demonstrating its practical relevance. Besides rigorously modelling this primitive, we provide two instantiations. While both are interesting from a theoretical point of view, the latter is also interesting in practice. In particular, due to using key-homomorphic signatures as we introduce them in this paper, we obtain a simple and practically efficient solution.

[5] This is underpinned by the results in Table 1, where $\mathcal{O}(n)$ exponentiations happen.

References

[AHI11] Benny Applebaum, Danny Harnik, and Yuval Ishai. Semantic security under related-key attacks and applications. In *ICS*, 2011.

[BBD$^+$10] Christina Brzuska, Heike Busch, Özgür Dagdelen, Marc Fischlin, Martin Franz, Stefan Katzenbeisser, Mark Manulis, Cristina Onete, Andreas Peter, Bertram Poettering, and Dominique Schröder. Redactable signatures for tree-structured data: Definitions and constructions. In *ACNS*, 2010.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO*, 2004.

[BCM11] Mihir Bellare, David Cash, and Rachel Miller. Cryptography secure against related-key attacks and tampering. In *ASIACRYPT*, 2011.

[BFLS10] Christina Brzuska, Marc Fischlin, Anja Lehmann, and Dominique Schröder. Unlinkability of sanitizable signatures. In *PKC*, 2010.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4), 2004.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT*, 2003.

[BPS13] Christina Brzuska, Henrich Christopher Pöhls, and Kai Samelin. Efficient and perfectly unlinkable sanitizable signatures without group signatures. In *EuroPKI 2013*, 2013.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA*, 2005.

[CA89] David Chaum and Hans Van Antwerpen. Undeniable signatures. In *CRYPTO*, 1989.

[CDDT12] Sébastien Canard, Nicolas Desmoulins, Julien Devigne, and Jacques Traoré. On the implementation of a pairing-based cryptographic protocol in a constrained device. In *Pairing*, 2012.

[CDHK15] Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In *ASIACRYPT*, 2015.

[Cha94] David Chaum. Designated confirmer signatures. In *EUROCRYPT*, 1994.

[DHS15] David Derler, Christian Hanser, and Daniel Slamanig. Revisiting cryptographic accumulators, additional properties and relations to other primitives. In *CT-RSA*, 2015.

[DPSS15] David Derler, Henrich C. Pöhls, Kai Samelin, and Daniel Slamanig. A general framework for redactable signatures and new constructions. In *ICISC*, 2015.

[DS16a] David Derler and Daniel Slamanig. Fully-anonymous short dynamic group signatures without encryption, 2016.

[DS16b] David Derler and Daniel Slamanig. Key-homomorphic signatures and applications to multiparty signatures. *IACR Cryptology ePrint Archive*, 2016:792, 2016.

[FKM$^+$16] Nils Fleischhacker, Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin. Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. In *PKC*, 2016.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO '86*, volume 263, pages 186–194, 1986.

[JMSW02] Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic signature schemes. In *CT-RSA*, 2002.

[JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT*, 1996.

[LWB05] Helger Lipmaa, Guilin Wang, and Feng Bao. Designated verifier signature schemes: Attacks, new security notions and a new construction. In *ICALP*, 2005.

[LZCS16] Russell W. F. Lai, Tao Zhang, Sherman S. M. Chow, and Dominique Schröder. Efficient sanitizable signatures without random oracles. In *ESORICS*, 2016.

[MPV09] Jean Monnerat, Sylvain Pasini, and Serge Vaudenay. Efficient deniable authentication for signatures. In *ACNS*, 2009.

[PS15] Henrich C. Pöhls and Kai Samelin. Accountable redactable signatures. In *ARES*, 2015.

[PS16] David Pointcheval and Olivier Sanders. Short randomizable signatures. In *CT-RSA*, 2016.

[RY07] Thomas Ristenpart and Scott Yilek. The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In *EUROCRYPT*, 2007.

[SBWP03] Ron Steinfeld, Laurence Bull, Huaxiong Wang, and Josef Pieprzyk. Universal designated-verifier signatures. In *ASIACRYPT*, 2003.

[SBZ01] Ron Steinfeld, Laurence Bull, and Yuliang Zheng. Content extraction signatures. In *ICISC*, 2001.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[SS08] Siamak Fayyaz Shahandashti and Reihaneh Safavi-Naini. Construction of universal designated-verifier signatures and identity-based signatures from standard signatures. In *PKC*, 2008.

[TW14] Stefano Tessaro and David A. Wilson. Bounded-collusion identity-based encryption from semantically-secure public-key encryption: Generic constructions with short ciphertexts. In *PKC*, 2014.

[UW14] Thomas Unterluggauer and Erich Wenger. Efficient pairings and ECC for embedded systems. In *CHES*, 2014.

[Ver06] Damien Vergnaud. New extensions of pairing-based signatures into universal designated verifier signatures. In *ICALP*, 2006.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, 2005.

A Definitions and Security Notions

Digital Signatures. Subsequently, we recall a definition of signatures.

Definition 16. *A signature scheme* Σ *is a triple* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *of PPT algorithms, which are defined as follows:*

$\mathsf{KeyGen}(1^\kappa)$: *This algorithm takes a security parameter* κ *as input and outputs a secret (signing) key* sk *and a public (verification) key* pk *with associated message space* $\mathcal{M}$ *(we may omit to mention the message space* $\mathcal{M}$*).*

$\mathsf{Sign}(\mathsf{sk}, m)$: *This algorithm takes a secret key* sk *and a message* $m \in \mathcal{M}$ *as input and outputs a signature* σ*.*

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: *This algorithm takes a public key* pk*, a message* $m \in \mathcal{M}$ *and a signature* σ *as input and outputs a bit* $b \in \{0, 1\}$.

In addition, we require an algorithm $\mathsf{VKey}(\cdot, \cdot)$, which checks whether a key pair is a valid output of KeyGen, i.e., for any $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$ we have $\mathsf{VKey}(\mathsf{sk}, \mathsf{pk}) = 1$. Besides correctness, Σ needs to be existentially unforgeable under adaptively chosen message attacks ($\mathsf{EUF\text{-}CMA}$). Subsequently, we formally recall the definition of $\mathsf{EUF\text{-}CMA}$ security.

Definition 17 **($\mathsf{EUF\text{-}CMA}$).** *A signature scheme* Σ *is* $\mathsf{EUF\text{-}CMA}$ *secure, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\left[\begin{matrix} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa), \\ (m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(\mathsf{pk}) \end{matrix} : \begin{matrix} \mathsf{Verify}(\mathsf{pk}, m^\star, \sigma^\star) = 1 \ \wedge \\ m^\star \notin \mathcal{Q}^{\mathsf{Sign}} \end{matrix}\right] \leq \varepsilon(\kappa),$$

where the environment keeps track of the queries to the signing oracle via $\mathcal{Q}^{\mathsf{Sign}}$.

We call a signature scheme *secure*, if it is correct and provides $\mathsf{EUF\text{-}CMA}$ security.

Non-Interactive Proof Systems. Now, we recall a standard definition of non-interactive proof systems (Π). Therefore, let L_R be an $\mathbf{NP}$-language with witness relation R defined as $L_R = \{x \mid \exists\ w : R(x, w) = 1\}$.

Definition 18. *A non-interactive proof system* Π *is a tuple of algorithms* (Setup, Proof, Verify)*, which are defined as follows:*

$\mathsf{Setup}(1^\kappa)$: This PPT algorithm takes a security parameter κ as input, and outputs a common reference string crs.
$\mathsf{Proof}(\mathsf{crs}, x, w)$: This algorithm takes a common reference string crs, a statement x, and a witness w as input, and outputs a proof π.
$\mathsf{Verify}(\mathsf{crs}, x, \pi)$: This PPT algorithm takes a common reference string crs, a statement x, and a proof π as input, and outputs a bit $b \in \{0, 1\}$.

If, in addition, if algorithm Proof runs in polynomial time, then we talk about a non-interactive witness-indistinguishable argument system. We require Π to be complete, sound, and adaptively witness-indistinguishable. Subsequently, we recall formal definition of those properties.

Definition 19 (Completeness). *A non-interactive proof system* Π *is* complete*, if for every adversary* $\mathcal{A}$ *it holds that*

$$\Pr\left[\begin{matrix} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa),\ (x^\star, w^\star) \leftarrow \mathcal{A}(\mathsf{crs}), \\ \pi \leftarrow \mathsf{Proof}(\mathsf{crs}, x^\star, w^\star) \end{matrix} : \begin{matrix} \mathsf{Verify}(\mathsf{crs}, x^\star, \pi) = 1 \\ \wedge\ (x^\star, w^\star) \in R \end{matrix}\right] = 1.$$

Definition 20 (Soundness). *A non-interactive proof system* Π *is* sound*, if for every PPT adversary* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa),\ (x^\star, \pi^\star) \leftarrow \mathcal{A}(\mathsf{crs}) : \begin{matrix} \mathsf{Verify}(\mathsf{crs}, x^\star, \pi^\star) = 1 \\ \wedge\ x^\star \notin L_R \end{matrix}\right] \leq \varepsilon(\kappa).$$

If $\varepsilon = 0$, we have perfect soundness.

Definition 21 (Adaptive Witness-Indistinguishability). *A non-interactive proof system* Π *is* adaptively witness-indistinguishable*, if for every PPT adversary* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa),\ b \xleftarrow{R} \{0,1\},\ b^\star \leftarrow \mathcal{A}^{\mathcal{P}(\mathsf{crs},\cdot,\cdot,\cdot,b)}(\mathsf{crs}) : b = b^\star\right] \leq \varepsilon(\kappa),$$

where $\mathcal{P}(\mathsf{crs}, x, w_0, w_1, b) := \mathsf{Proof}(\mathsf{crs}, x, w_b)$*, and* $\mathcal{P}$ *returns* $\bot$ *if* $(x, w_0) \notin R \ \vee\ (x, w_1) \notin R$*.*

If $\varepsilon = 0$, we have perfect adaptive witness-indistinguishability. Furthermore, we require Π to admit proofs of knowledge, which are defined as follows.

Definition 22 (Proof of Knowledge). *A non-interactive proof system* Π *admits proofs of knowledge, if there exists a PPT extractor* $\mathsf{E} = (E_1, E_2)$ *such that for every PPT adversary* $\mathcal{A}$ *there is a negligible function* $\varepsilon_1(\cdot)$ *such that*

$$\left| \begin{array}{l} \Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa) : \mathcal{A}(\mathsf{crs}) = 1\right]\ - \\ \Pr\left[(\mathsf{crs}, \tau) \leftarrow E_1(1^\kappa) : \mathcal{A}(\mathsf{crs}) = 1\right] \end{array} \right| \leq \varepsilon_1(\kappa),$$

and for every PPT adversary $\mathcal{A}$ *there is a negligible function* $\varepsilon_2(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l} (\mathsf{crs}, \tau) \leftarrow E_1(1^\kappa),\ (x^\star, \pi^\star) \leftarrow \mathcal{A}(\mathsf{crs}), \\ w \leftarrow E_2(\mathsf{crs}, \tau, x^\star, \pi^\star) \end{array} : \begin{array}{r} \mathsf{Verify}(\mathsf{crs}, x^\star, \pi^\star) = 1\ \wedge \\ (x^\star, w) \notin R \end{array}\right] \leq \varepsilon_2(\kappa).$$

B Proof of Theorem 1

We show that Theorem 1 holds by proving Lemma 1-6.

Lemma 1. *If* GS *is correct,* RS *is correct,* Σ *is correct, and* Π *is complete, then Scheme 1 is also correct.*

Lemma 1 straight-forwardly follows from inspection; the proof is omitted.

Lemma 2. *If* GS *is traceable and* RS *is unforgeable, then Scheme 1 is group unforgeable.*

Proof. We construct efficient reductions $\mathcal{R}^{\mathsf{t}}$ and $\mathcal{R}^{\mathsf{u}}$ turning an efficient group unforgeability adversary $\mathcal{A}^{\mathsf{gu}}$, into an efficient adversary (1) $\mathcal{A}^{\mathsf{t}}$ against traceability of GS, or (2) $\mathcal{A}^{\mathsf{u}}$ against unforgeability of the RS.

(1) $\mathcal{R}^{\mathsf{t}}$ obtains $(\mathsf{gpk}, \mathsf{gok})$ from a GS traceability challenger $\mathcal{C}^{\mathsf{t}}_\kappa$, completes the setup as in the real game, and starts $\mathcal{A}^{\mathsf{gu}}$ on $(\mathsf{gpk}', \mathsf{gok})$. Sig queries are simulated by obtaining the group signature σ_{G} using the Sig oracle provided by $\mathcal{C}^{\mathsf{t}}_\kappa$ and running the remaining Sign algorithm as in the original protocol. Key queries are simply forwarded to $\mathcal{C}^{\mathsf{t}}_\kappa$. Eventually, $\mathcal{A}^{\mathsf{gu}}$ outputs a forgery $(\mathsf{m}^\star, \sigma^\star)$ which is opened to signer index u (recall that $\sigma^\star = ((\mathsf{pk}, \sigma_{\mathsf{G}}), (\sigma_{\mathsf{R}}, \mathsf{RED}))$. If u exists ($u \neq \bot$), and $\mathcal{A}^{\mathsf{gu}}$ either requested gsk_u or a group signature on pk for u (i.e., $u \in \mathcal{Q}^{\mathsf{Key}} \ \vee\ (u, \mathsf{pk}) \in \mathcal{Q}^{\mathsf{Sign}}$), we abort as we are in the other case. Otherwise, we output $(\mathsf{pk}, \sigma_{\mathsf{G}})$ as a valid forgery for traceability of GS.

(2) $\mathcal{R}^{\mathsf{u}}$ runs the setup as in the real game and starts $\mathcal{A}^{\mathsf{gu}}$ on $(\mathsf{gpk}', \mathsf{gok})$. On each Sig query, $\mathcal{R}^{\mathsf{u}}$ engages with an RS unforgeability challenger $\mathcal{C}^{\mathsf{u}}_{\kappa}$, obtains pk and computes the RS signature using the Sign oracle provided by $\mathcal{C}^{\mathsf{u}}_{\kappa}$. The remaining simulation is performed as in the original scheme. Eventually, $\mathcal{A}^{\mathsf{gu}}$ outputs a forgery $(\mathsf{m}^\star, \sigma^\star)$ which is opened to signer index u (recall that $\sigma^\star = ((\mathsf{pk}, \sigma_{\mathsf{G}}), (\sigma_{\mathsf{R}}, \mathsf{RED}))$. If u does not exist ($u = \bot$), or u exists and $\mathcal{A}^{\mathsf{gu}}$ neither requested gsk_u nor a group signature on pk for u (i.e., $u \notin \mathcal{Q}^{\mathsf{Key}} \wedge (u, \mathsf{pk}) \notin \mathcal{Q}^{\mathsf{Sign}}$) we abort as we are in the other case. Otherwise, we know that we have a valid RS signature on $\mathsf{m}^\star$ under pk which is not derivable from any queried message (i.e., $\nexists\ (u, \mathsf{m}, \mathsf{ADM}) \in \mathcal{Q}^{\mathsf{Sig}} : \mathsf{m}^\star \overset{\mathsf{ADM}}{\preceq} \mathsf{m}$) and we can output $(\mathsf{m}^\star, \sigma_{\mathsf{R}})$ as an RS forgery.

Overall Bound. The simulations in both reductions are indistinguishable from a real game. In front of an adversary we randomly guess the adversary's strategy, inducing a loss of $1/2$. Our reduction for a Type 1 forger always succeeds if the adversary succeeds, whereas our reduction for Type 2 succeeds with a probability of $1/q$, where $q \leq \mathsf{poly}(\kappa)$ is the number of queries to the Sig oracle. Overall, this means that the probability to break group unforgeability is upper-bounded by $2 \cdot \max\{\varepsilon_{\mathsf{gu}}(\kappa), q \cdot \varepsilon_{\mathsf{p}}(\kappa)\}$. $\square$

Lemma 3. *If* Π *admits proofs of knowledge,* Σ *is* $\mathsf{EUF\text{-}CMA}$ *secure, and Scheme 1 is group unforgeable, then Scheme 1 is also designated-verifier unforgeable.*

Proof. We bound the probability to break designated-verifier unforgeability. We start by defining our opener $\mathsf{O} = (O_1, O_2)$.

$O_1(1^\kappa, n)$: Run $(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}) \leftarrow \mathsf{GS.KeyGen}(1^\kappa, n)$, $(\mathsf{crs}, \tau) \leftarrow \Pi.E_1(1^\kappa)$, set $\mathsf{gpk}' \leftarrow (\mathsf{gpk}, \mathsf{crs})$, $\tau' \leftarrow (\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk}, \tau)$ and return $(\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk}, \tau')$.
$O_2(\tau, \mathtt{DVK}, \mathsf{m}^\star, \rho^\star)$: Parse σ as $((\mathsf{pk}, \sigma_{\mathsf{G}}), \overline{\sigma})$ and return $u \leftarrow \mathsf{GS.Open}(\mathsf{gok}, \mathsf{pk}, \sigma_{\mathsf{G}})$.

The tuple $(\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk})$ contained in the output of O_1 is computationally indistinguishable from the output of Setup under the extraction-CRS indistinguishability of the proof system.

What remains is to show that—using O—the success probability of every PPT adversary in the designated-verifier unforgeability game is negligible in the security parameter. We do so by using a sequence of games, where we let $q \leq \mathsf{poly}(\kappa)$ be the number of queries to the DVGen oracle.

Game 0: The original designated-verifier-unforgeability game.
Game 1: As Game 0, but we modify O_1 as follows. We use $\mathcal{C}^{\mathsf{gu}}_{\kappa}$ to denote a group unforgeability challenger.

$O_1(1^\kappa, n)$: Run $\boxed{(\mathsf{gpk}, \mathsf{gok}) \leftarrow \mathcal{C}^{\mathsf{gu}}_{\kappa}}$, set $\boxed{\mathsf{gsk} \leftarrow \bot}$, $(\mathsf{crs}, \tau) \leftarrow \Pi.E_1(1^\kappa)$, set $\mathsf{gpk}' \leftarrow (\mathsf{gpk}, \mathsf{crs})$, $\tau' \leftarrow (\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk}, \tau)$ and return $(\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk}, \tau')$.

Then, we simulate all queries to Sig and Key by forwarding them to $\mathcal{C}^{\mathsf{gu}}_{\kappa}$.
Transition - Game 0 → Game 1: This change is conceptual, i.e., $\Pr[S_0] = \Pr[S_1]$.

Game 2: As Game 1, but we guess the index $v^\star$ that the adversary will attack.
Transition - Game 1 → Game 2: The success probability in Game 2 is the same as in Game 1, unless our guess is wrong. That is $\Pr[S_2] = \Pr[S_1] \cdot {}^1/_q$.
Game 3: As Game 2, but in the query to DVGen for user $v^\star$ we engage with an $\mathsf{EUF\text{-}CMA}$ challenger $\mathcal{C}^{\mathsf{f}}_\kappa$, obtain a public key pk and return $\mathsf{vpk}_{v^\star} \leftarrow \mathsf{pk}$. Furthermore, the queries to Sim for user $v^\star$ are simulated without $\mathsf{vsk}_{v^\star}$ by using the Sign oracle provided by $\mathcal{C}^{\mathsf{f}}_\kappa$.
Transition - Game 2 → Game 3: This change is conceptual, i.e., $\Pr[S_3] = \Pr[S_2]$.
Game 4: As Game 3, but for every output of the adversary, we obtain $(\sigma_{\mathsf{R}}, \sigma_{\mathsf{V}}) \leftarrow \Pi.E_2(\mathsf{crs}, \tau, (\mathsf{m}^\star, \mathsf{pk}, \mathsf{vpk}_{v^\star}), \pi)$. If the extractor fails, we abort.
Transition - Game 3 → Game 4: The success probability in Game 2 is the same as in Game 1, unless the extractor fails, i.e., $\Pr[S_4] = \Pr[S_3] \cdot (1 - \varepsilon_{\mathsf{ext2}}(\kappa))$.

In Game 4 we have two possibilities if $\mathcal{A}$ outputs a valid forgery.

1. We extract a signature σ_{R} such that $\mathsf{RS.Verify}(\mathsf{pk}, \mathsf{m}^\star, \sigma_{\mathsf{R}}) = 1$. Since, our implementation of O_1 does the same as what is done in Open and we have that $(u = \perp \vee (u \notin \mathcal{Q}^{\mathsf{Key}} \wedge \nexists (u, \mathsf{m}, \mathsf{ADM}) \in \mathcal{Q}^{\mathsf{Sig}} : \mathsf{m}^\star \overset{\mathsf{ADM}}{\preceq} \mathsf{m}))$ by definition, we can compose $\sigma \leftarrow ((\mathsf{pk}, \sigma_{\mathsf{G}}), (\sigma_{\mathsf{R}}, \perp))$ and return $(\mathsf{m}^\star, \sigma)$ to $\mathcal{C}^{\mathsf{gu}}_\kappa$ as a forgery for the group unforgeability game.
2. We extract a signature σ_{V} such that $\Sigma.\mathsf{Verify}(\mathsf{vpk}_{v^\star}, \mathsf{m}^\star, \sigma_{\mathsf{V}}) = 1$. By definition, we have that $v^\star \notin \mathcal{Q}^{\mathsf{DVKey}} \wedge \nexists (v^\star, \mathsf{m}, \mathsf{ADM}, \cdot, \cdot) \in \mathcal{Q}^{\mathsf{Sim}} : \mathsf{m}^\star \overset{\mathsf{ADM}}{\preceq} \mathsf{m})$. Since $\mathsf{m}^\star \overset{\mathsf{ADM}}{\preceq} \mathsf{m}$ also includes the identity, i.e., the case where $\mathsf{m}^\star = \mathsf{m}$, we know that $\mathsf{m}^\star$ was never queried to the signing oracle provided by $\mathcal{C}^{\mathsf{f}}_\kappa$ and we can output $(\mathsf{m}^\star, \sigma_{\mathsf{V}})$ as a valid $\mathsf{EUF\text{-}CMA}$ forgery.

The union bound yields $\Pr[S_4] \leq \varepsilon_{\mathsf{gu}}(\kappa) + \varepsilon_{\mathsf{f}}(\kappa)$. Furthermore, we have that $\Pr[S_4] = \Pr[S_3] \cdot (1 - \varepsilon_{\mathsf{ext2}}(\kappa))$, that $\Pr[S_3] = \Pr[S_2] = \Pr[S_1] \cdot {}^1/_q$, and that $\Pr[S_0] = \Pr[S_1]$. All in all this yields $\Pr[S_0] \leq q \cdot \frac{\varepsilon_{\mathsf{gu}}(\kappa) + \varepsilon_{\mathsf{f}}(\kappa)}{1 - \varepsilon_{\mathsf{ext2}}(\kappa)}$, which proves the lemma. □

Lemma 4. *If* Π *is witness indistinguishable, then Scheme 1 is simulatable.*

Proof. We show that the output in the simulatability game is (computationally) independent of the bit b.

Game 0: The original simulatability game ($\underline{\sigma}$ is already independent of b).
Game 1: As Game 0, but we obtain crs for the Π upon Setup from a witness indistinguishability challenger $\mathcal{C}^{\mathsf{wi}}_\kappa$ instead of internally generating it.
Transition - Game 0 → Game 1: This change is conceptual, i.e., $\Pr[S_0] = \Pr[S_1]$.
Game 2: As Game 1, but instead of executing Redact inside RoS, we execute the modified algorithm Redact' with additional input vsk_j, which additionally computes $\boxed{\sigma_{\mathsf{V}} \leftarrow \Sigma.\mathsf{Sign}(\mathsf{vsk}_j, \mathring{\mathsf{m}})}$ and then computes π as $\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathring{\mathsf{m}}, \mathsf{pk}, \mathsf{vpk}_j), \boxed{(\perp, \sigma_{\mathsf{V}})})$.
Transition - Game 1 → Game 2: A distinguisher $\mathcal{D}^{1 \rightarrow 2}$ is a distinguisher for adaptive witness indistinguishability of the Π, i.e., $|\Pr[S_3] - \Pr[S_2]| \leq \varepsilon_{\mathsf{wi}}(\kappa)$.

In Game 2, Redact' and Sim are identical, i.e., RoS is independent of b. Thus, the adversary has no advantage in winning the game, i.e., $\Pr[S_2] = 1/2$, which yields $\Pr[S_0] \leq 1/2 + \varepsilon_{\mathsf{wi}}(\kappa)$. $\square$

Lemma 5. *If* GS *is anonymous and* RS *is unforgeable, then Scheme 1 is signer anonymous.*

Proof. We construct an efficient reduction $\mathcal{R}$ which turns an efficient signer-anonymity adversary $\mathcal{A}^{\mathsf{sa}}$ into an efficient adversary $\mathcal{A}$ against anonymity of the underlying GS. $\mathcal{R}$ obtains $(\mathsf{gpk}, \mathsf{gsk})$ from the challenger $\mathcal{C}^{\mathsf{a}}_{\kappa}$ of the anonymity game of GS and completes the setup as in the original scheme. $\mathcal{R}$ simulates the Open oracle by using the Open oracle provided by $\mathcal{C}^{\mathsf{a}}_{\kappa}$ and starts $\mathcal{A}^{\mathsf{sa}}$ on $(\mathsf{gpk}', \mathsf{gsk})$. If $\mathcal{A}^{\mathsf{sa}}$ eventually outputs $b^\star$, then $\mathcal{R}$ outputs $b^\star$ to $\mathcal{C}^{\mathsf{a}}_{\kappa}$. By the RS unforgeability, the simulation of the Open oracle is computationally indistinguishable from a real game. The reduction succeeds with non-negligible probability whenever $\mathcal{A}^{\mathsf{sa}}$ succeeds with non-negligible probability. $\square$

Lemma 6. *If* RS *is private, then Scheme 1 is private.*

Proof. We prove privacy using a sequence of games, where we let $q \leq \mathsf{poly}(\kappa)$ be the number of queries to the Ch oracle.

Game 0: The privacy game with bit $b = 0$.

Game 1_ℓ $(1 \leq \ell \leq q)$: As Game 0, but we set $b = 1$ for the first ℓ queries to Ch.

Transition - Game 0 → Game 1_1: A distinguisher between Game 0 and Game 1_1 is a distinguisher for the RS privacy game. To show this, we engage with an RS privacy challenger $\mathcal{C}^{\mathsf{p}}_{\kappa}$ in the first call to Ch, obtain pk, compute $\sigma_{\mathsf{G}} \leftarrow \mathsf{GS.Sign}(\mathsf{gsk}_i, \mathsf{pk})$, $(\mathring{\mathsf{m}}, \mathring{\sigma}_{\mathsf{R}}) \leftarrow \mathcal{C}^{\mathsf{p}}_{\kappa}.\mathsf{LoRRedact}((\mathsf{m}_0, \textsf{MOD}_0, \textsf{ADM}_0), (\mathsf{m}_1, \textsf{MOD}_1, \textsf{ADM}_1))$, as well as $\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathring{\mathsf{m}}, \mathsf{pk}, \mathsf{vpk}_j), (\mathring{\sigma}_{\mathsf{R}}, \perp))$, and return $(\mathring{\mathsf{m}}, \underline{\sigma}, \rho) = (\mathsf{m}, (\mathsf{pk}, \sigma_{\mathsf{G}}), ((\mathsf{pk}, \sigma_{\mathsf{G}}), \pi))$. Depending on the bit chosen by $\mathcal{C}^{\mathsf{p}}_{\kappa}$, we either simulate Game 0 or Game 1_1.

Transition - Game $1_\ell \to 1_{\ell+1}$ $(1 \leq \ell < q)$: The answers of the Ch oracle for the first ℓ queries are already simulated for $b = 1$. As above, a distinguisher between Game 1_ℓ and Game $1_{\ell+1}$ is a RS privacy distinguisher.

In Game 1_q we have a simulation for bit $b = 1$. We can bound probability to distinguish the simulations for $b = 0$ and $b = 1$ by $|\Pr[S_{1_q}] - \Pr[S_0]| \leq q \cdot \varepsilon_{\mathsf{p}}(\kappa)$, which shows that the advantage to win the privacy game is bounded by $1/2 + q \cdot \varepsilon_{\mathsf{p}}(\kappa)$. $\square$

C Proof of Theorem 2

Subsequently, we show that Theorem 2 holds by proving Lemma 7-12.

Lemma 7. *If* GS *is correct,* RS *is correct and adapts signatures,* Σ *is correct, and* Π *is complete, then Scheme 2 is also correct.*

Lemma 7 straight-forwardly follows from inspection; the proof is omitted.

Lemma 8. *If* GS *is traceable and* RS *is unforgeable, then Scheme 2 is group unforgeable.*

Lemma 8 can be proven identically as group unforgeability is proven in the previous section and is therefore omitted.

Lemma 9. *If* Π *admits proofs of knowledge,* Σ *is* $\mathsf{EUF\text{-}CMA}$ *secure,* RS *adapts signatures, Scheme 2 is group unforgeable, and the DL assumption holds in* $\mathbb{G}$*, then Scheme 2 is also designated-verifier unforgeable.*

Proof. We subsequently prove designated-verifier unforgeability by distinguishing two cases. In front of $\mathcal{A}^{\mathsf{dv}}$, we guess its strategy uniformly at random and either follow (1) or (2).

(1) We define the opener $\mathsf{O} = (O_1, O_2)$ as follows.

$O_1(1^\kappa, n)$: Run $(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}) \leftarrow \mathsf{GS.KeyGen}(1^\kappa, n)$, $(\mathsf{crs}, \tau) \leftarrow \Pi.E_1(1^\kappa)$, set $\mathsf{gpk}' \leftarrow (\mathsf{gpk}, \mathsf{crs})$, $\tau' \leftarrow (\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}, \tau)$, and return $(\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk}, \tau')$.
$O_2(\tau, \mathtt{DVK}, \mathsf{m}^\star, \rho^\star)$: Parse σ as $((\mathsf{pk}, \sigma_{\mathsf{G}}), \overline{\sigma})$ and return $u \leftarrow \mathsf{GS.Open}(\mathsf{gok}, \mathsf{pk}, \sigma_{\mathsf{G}})$.

The tuple $(\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk})$ contained in the output of O_1 is computationally indistinguishable from the output of Setup under the extraction-CRS indistinguishability of the proof system.

What remains is to show that—using O—the success probability of every PPT adversary following strategy (1) in the designated-verifier unforgeability game is negligible in the security parameter. We do so by using a sequence of games where we let $q_{\mathsf{Sim}} \leq \mathsf{poly}(\kappa)$ be the number of queries to the Sim oracle.

Game 0: The original designated-verifier unforgeability game.
Game 1: As Game 0, but we modify O_1 as follows, where $\mathcal{C}^{\mathsf{gu}}_\kappa$ denotes a group-unforgeability challenger (note that we can assume that all required accumulator public keys are obtained from collision freeness challengers):

$O_1(1^\kappa, n)$: Run $\boxed{(\mathsf{gpk}, \mathsf{gok}) \leftarrow \mathcal{C}^{\mathsf{gu}}_\kappa}$, set $\boxed{\mathsf{gsk} \leftarrow \perp}$, $(\mathsf{crs}, \tau) \leftarrow \Pi.E_1(1^\kappa)$, $\mathsf{gpk}' \leftarrow (\mathsf{gpk}, \mathsf{crs})$, $\tau' \leftarrow (\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}, \tau)$, and return $(\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk}, \tau')$.

Furthermore, whenever the adversary queries Sig or Key, we use the oracles provided by $\mathcal{C}^{\mathsf{gu}}_\kappa$ to obtain the required group signatures and keys, respectively.

Transition - Game 0 → Game 1: This game change is conceptual and $\Pr[S_1] = \Pr[S_0]$.

Game 2: As Game 1, but whenever the adversary outputs a forgery so that $\mathsf{pk}_\Sigma \cdot \mathsf{pk}'$ corresponds to a key $\mathsf{pk}^\Sigma_{\mathsf{R}}$ used in a Sim oracle call we check whether the signed accumulator value (used to encode the message) is still the same as the one signed in Sim and abort if so.

Transition - Game 1 → Game 2: If we abort, we have a collision for one of the accumulators. That is, $\Pr[S_2] = (1 - \varepsilon_{\mathsf{cf}}(\kappa))^{q_{\mathsf{Sim}}} \cdot \Pr[S_1]$.

Game 3: As Game 2, but inside Sim we obtain $\mathsf{pk}_{\mathsf{R}}^{\Sigma}$ from an $\mathsf{EUF\text{-}CMA}$ challenger of Σ and obtain the required signatures inside $\mathsf{RS.Sign}$ using the Sign oracle provided by the challenger.

Transition - Game 2 → Game 3: This change is conceptual: $\Pr[S_2] = \Pr[S_3]$.[6]

Game 4: As Game 3, but for every output of the adversary, we obtain $(\mathsf{sk}', \mathsf{vsk}_{v^\star}) \leftarrow \Pi.E_2(\mathsf{crs}, \tau, (\mathsf{pk}', \mathsf{vpk}_{v^\star}), \pi)$. If the extractor fails, we abort.

Transition - Game 3 → Game 4: The success probability in Game 3 is the same as in Game 4, unless the extractor fails, i.e., $\Pr[S_4] = \Pr[S_3] \cdot (1 - \varepsilon_{\mathsf{ext2}}(\kappa))$.

If the adversary outputs a forgery $(\mathsf{m}^\star, \rho^\star, v^\star)$, where $\rho^\star = (((\mathsf{pk}_\Sigma, \mathsf{pk}_\Lambda), \sigma_{\mathsf{G}}), \mathsf{pk}', \mathring{\sigma}'_{\mathsf{R}}, \pi)$, in Game 4, we check whether we have extracted a secret key sk' such that $\mathsf{RS.VKey}(\mathsf{pk}', \mathsf{sk}') = 0$ and abort if so (then we are in the other case). Otherwise, we have that $\mathsf{Verify}(\mathsf{gpk}, \mathsf{vpk}_{v^\star}, \mathsf{m}^\star, \rho^\star) = 1$ by definition. If $\mathsf{pk}_\Sigma \cdot \mathsf{pk}'$ corresponds to a key $\mathsf{pk}_{\mathsf{R}}^{\Sigma}$ used in Sim, we can output the Σ-signature σ on the accumulator together with the accumulator as an $\mathsf{EUF\text{-}CMA}$ forgery to one of the challengers from Sim. If not, we can obtain $(\mathsf{pk}, \mathring{\sigma}''_{\mathsf{R}}) \leftarrow \mathsf{RS.Adapt}(\mathsf{pk}_\Sigma \cdot \mathsf{pk}', \mathsf{m}^\star, \mathring{\sigma}'_{\mathsf{R}}, -\mathsf{sk}')$ and output $(\mathsf{m}^\star, \sigma) = (\mathsf{m}^\star, (((\mathsf{pk}_\Sigma, \mathsf{pk}_\Lambda), \sigma_{\mathsf{G}}), (\mathring{\sigma}''_{\mathsf{R}}, \bot)))$ to break group unforgeability. Note that our implementation of O_2 does the same as what is done in Open and we have that $(u = \bot \ \vee \ (u \notin \mathcal{Q}^{\mathsf{Key}} \ \wedge \ \nexists(u, \mathsf{m}, \mathsf{ADM}) \in \mathcal{Q}^{\mathsf{Sig}} : \mathsf{m}^\star \overset{\mathsf{ADM}}{\preceq} \mathsf{m}))$ by definition. Also note that Game 2 and Game 3 resemble the proof strategy for RS constructions following the paradigm from [DPSS15]. Taking the union bound, the success probability of $\mathcal{A}^{\mathsf{dv}}$ in Game 4 is bounded by $\Pr[S_4] \leq \varepsilon_{\mathsf{gu}}(\kappa) + q_{\mathsf{Sim}} \cdot \varepsilon_{\mathsf{f}}(\kappa)$. We have that $\Pr[S_0] = \frac{\Pr[S_3]}{(1-\varepsilon_{\mathsf{cf}}(\kappa))^{q_{\mathsf{Sim}}}}$ and that $\Pr[S_3] = \frac{\Pr[S_4]}{(1-\varepsilon_{\mathsf{ext2}}(\kappa))}$. All in all, this yields, $\Pr[S_0] \leq \frac{\varepsilon_{\mathsf{gu}}(\kappa) + q_{\mathsf{Sim}} \cdot \varepsilon_{\mathsf{f}}(\kappa)}{(1-\varepsilon_{\mathsf{cf}}(\kappa))^{q_{\mathsf{Sim}}} \cdot (1-\varepsilon_{\mathsf{ext2}}(\kappa))}$.

(2) We define the opener $\mathsf{O} = (O_1, O_2)$ as follows, where $\mathcal{C}_\kappa^{\mathsf{wi}}$ denotes a witness indistinguishability challenger.

$O_1(1^\kappa, n)$: Run $(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}) \leftarrow \mathsf{GS.KeyGen}(1^\kappa, n)$, $\mathsf{crs} \leftarrow \mathcal{C}_\kappa^{\mathsf{wi}}$, set $\mathsf{gpk}' \leftarrow (\mathsf{gpk}, \mathsf{crs})$, $\tau' \leftarrow (\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk})$, and return $(\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk}, \tau')$.

$O_2(\tau, \mathtt{DVK}, \mathsf{m}^\star, \rho^\star)$: Parse σ as $((\mathsf{pk}, \sigma_{\mathsf{G}}), \overline{\sigma})$ and return $u \leftarrow \mathsf{GS.Open}(\mathsf{gok}, \mathsf{pk}, \sigma_{\mathsf{G}})$.

The tuple $(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk})$ contained in the output of O_1 is identically distributed as a tuple $(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk})$ returned by Setup.

What remains is to show that—using O—the success probability of every PPT adversary following strategy (2) in the designated-verifier unforgeability game is negligible in the security parameter. We do so by using a sequence of games.

Game 0: The original designated-verifier unforgeability game.

[6] Note that the changes in Game 2 and Game 3 resemble the unforgeability proof strategy of [DPSS15, Scheme 1]. For further details, and, in particular for the collision freeness notion of accumulators, see [DPSS15].

Game 1: As Game 0, but we modify the Sig oracle so that it records the Σ-component of the used RS secret key sk_Σ in a list $\mathtt{KEY}$ indexed by the corresponding public key pk_Σ, i.e., $\mathtt{KEY}[\mathsf{pk}_\Sigma] = \mathsf{sk}_\Sigma$.

Transition - Game 0 → Game 1: This change is conceptual, i.e., $\Pr[S_0] = \Pr[S_1]$.

Game 2: As Game 1, but we modify the Sim oracle by executing the following modified Sim algorithm Sim', which runs on additional inputs vpk_j and $\mathtt{KEY}$.

$\mathsf{Sim}'(\mathsf{gpk}, \mathsf{vsk}_j, \mathsf{m}, \mathsf{ADM}, \mathsf{MOD}, \underline{\sigma}, \mathsf{vpk}_j, \mathtt{KEY})$: If $\mathsf{MOD} \preceq \mathsf{ADM} \wedge \mathsf{ADM} \preceq \mathsf{m}$, parse $\underline{\sigma}$ as $((\mathsf{pk}_\Sigma, \mathsf{pk}_\Lambda), \sigma_\mathsf{G})$ and return $(\mathring{\mathsf{m}}, \rho)$, where

$\boxed{\mathsf{sk}' \xleftarrow{R} \mathbb{H},\ \mathsf{pk}' \leftarrow \mu(\mathsf{sk}'),\ \mathsf{sk}^\Sigma_\mathsf{R} \leftarrow \mathtt{KEY}[\mathsf{pk}_\Sigma] + \mathsf{sk}',\ \mathsf{pk}^\Sigma_\mathsf{R} \leftarrow \mu(\mathsf{sk}^\Sigma_\mathsf{R})}$,

$((\mathsf{m}, \sigma'_\mathsf{R}), \mathsf{RED}) \leftarrow \mathsf{RS.Sign}((\mathsf{sk}^\Sigma_\mathsf{R}, \perp, \mathsf{pk}_\Lambda), \mathsf{m}, \mathsf{ADM})$,

$((\mathring{\mathsf{m}}, \mathring{\sigma}'_\mathsf{R}), \cdot) \leftarrow \mathsf{RS.Redact}((\mathsf{pk}^\Sigma_\mathsf{R}, \mathsf{pk}_\Lambda), \mathsf{m}, \sigma'_\mathsf{R}, \mathsf{MOD}, \mathsf{RED})$,

$\pi \leftarrow \mathsf{\Pi.Proof}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}_j), (\perp, \mathsf{vsk}_j))$, and $\rho \leftarrow (\underline{\sigma}, \mathsf{pk}', \mathring{\sigma}'_\mathsf{R}, \pi)$.

Transition - Game 1 → Game 2: Under adaptability of the RS, Game 1 and Game 2 are perfectly indistinguishable, i.e., $\Pr[S1] = \Pr[S_2]$.

Game 3: As Game 2, but we further modify Sim', which now additionally runs without vsk_j and computes π as $\pi \leftarrow \mathsf{\Pi.Proof}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}_j), \boxed{(\mathsf{sk}', \perp)})$.

Transition - Game 2 → Game 3: A distinguisher $\mathcal{D}^{2\to3}$ is a distinguisher for adaptive witness indistinguishability of Π, i.e., $|\Pr[S_3] - \Pr[S_2]| \leq \varepsilon_\mathsf{wi}(\kappa)$.

Game 4: As Game 3, but we guess the index $v^\star$ that the adversary will attack. If our guess is wrong, we abort.

Transition - Game 3 → Game 4: The success probability in Game 4 is the same as in Game 3, unless our guess is wrong. That is $\Pr[S_4] = \Pr[S_3] \cdot {}^1\!/_q$.

Game 5: As Game 4, but in the query to DVGen for user $v^\star$ we engage with an $\mathsf{EUF\text{-}CMA}$ challenger $\mathcal{C}^\mathsf{f}_\kappa$, obtain a public key pk and return $\mathsf{vpk}_{v^\star} \leftarrow \mathsf{pk}$.

Transition - Game 4 → Game 5: This change is conceptual, i.e., $\Pr[S_4] = \Pr[S_5]$.

Game 6: As Game 5, but we modify O_1 as follows:

$O_1(1^\kappa, n)$: Run $(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}) \leftarrow \mathsf{GS.KeyGen}(1^\kappa, n)$, $\boxed{(\mathsf{crs}, \tau) \leftarrow \mathsf{\Pi}.E_1(1^\kappa)}$, set $\mathsf{gpk}' \leftarrow (\mathsf{gpk}, \mathsf{crs})$, $\tau' \leftarrow (\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}, \boxed{\tau})$, and return $(\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk}, \tau')$.

Transition - Game 5 → Game 6: A distinguisher $\mathcal{D}^{5\to6}$ distinguishes an extraction CRS from an honest CRS, i.e., $|\Pr[S_6] - \Pr[S_5]| \leq \varepsilon_\mathsf{ext1}(\kappa)$.

Game 7: As Game 6, but for every output of the adversary, we obtain $(\mathsf{sk}', \mathsf{vsk}_{v^\star}) \leftarrow \mathsf{\Pi}.E_2(\mathsf{crs}, \tau, (\mathsf{pk}', \mathsf{vpk}_{v^\star}), \pi)$. If the extractor fails, we abort.

Transition - Game 6 → Game 7: The success probability in Game 7 is the same as in Game 6, unless the extractor fails, i.e., $\Pr[S_7] = \Pr[S_6] \cdot (1 - \varepsilon_\mathsf{ext2}(\kappa))$.

If $\mathcal{A}^\mathsf{dv}$ outputs a valid forgery $(\mathsf{m}^\star, \rho^\star, v^\star)$, where $\rho^\star = ((\mathsf{pk}, \sigma_\mathsf{G}), \mathsf{pk}', \mathring{\sigma}'_\mathsf{R}, \pi)$, we check whether we have extracted $\mathsf{vsk}_{v^\star}$ such that $\Sigma.\mathsf{VKey}(\mathsf{vpk}_{v^\star}, \mathsf{vsk}_{v^\star}) = 0$ and abort if so. Otherwise, we choose a random message m in the message space of Σ, compute $\sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{vsk}_{v^\star}, m)$ and output (m, σ) as an $\mathsf{EUF\text{-}CMA}$ forgery for Σ. The success probability for $\mathcal{A}^\mathsf{dv}$ in Game 6 is thus bounded by $\varepsilon_\mathsf{f}(\kappa)$. Furthermore, we have that $\Pr[S_0] = \Pr[S_1] = \Pr[S_2] \leq \Pr[S_3] + \varepsilon_{\mathsf{wi}}(\kappa)$, that $\Pr[S_4] = 1/q \cdot \Pr[S_3]$, that $\Pr[S_4] = \Pr[S_5] \leq \Pr[S_6] + \varepsilon_{\mathsf{ext1}(\kappa)}$, and that $\Pr[S_6] = \frac{\Pr[S_7]}{1-\varepsilon_{\mathsf{ext2}}(\kappa)}$. Taking all together, we obtain that $\Pr[S_0] \leq q \cdot \big(\frac{\varepsilon_{\mathsf{f}}(\kappa)}{1-\varepsilon_{\mathsf{ext2}}(\kappa)} + \varepsilon_{\mathsf{ext1}}(\kappa)\big) + \varepsilon_{\mathsf{wi}}(\kappa)$.

Overall Bound. Taking (1) and (2) together, we obtain

$$2 \cdot \max\left\{\frac{\varepsilon_{\mathsf{gu}}(\kappa) + q_{\mathsf{Sim}} \cdot \varepsilon_{\mathsf{f}}(\kappa)}{(1-\varepsilon_{\mathsf{cf}}(\kappa))^{q_{\mathsf{Sim}}} \cdot (1-\varepsilon_{\mathsf{ext2}}(\kappa))}, q \cdot \big(\frac{\varepsilon_{\mathsf{f}}(\kappa)}{1-\varepsilon_{\mathsf{ext2}}(\kappa)} + \varepsilon_{\mathsf{ext1}}(\kappa)\big) + \varepsilon_{\mathsf{wi}}(\kappa)\right\}$$

as an upper bound for the success probability for the designated-verifier unforgeability game, which proves Lemma 9. □

Lemma 10. *If* Π *is witness indistinguishable, and* RS *adapts signatures, then Scheme 2 is simulatable.*

Proof. We prove that the output in the simulatability game is (computationally) independent of the bit b.

Game 0: The original simulatability game ($\underline{\sigma}$ is already independent of b).

Game 1: As Game 0, but we obtain crs for the Π upon Setup from a witness indistinguishability challenger $\mathcal{C}_{\kappa}^{\mathsf{wi}}$ instead of internally generating it.

Transition - Game 0 → Game 1: This change is conceptual, i.e., $\Pr[S_0] = \Pr[S_1]$.

Game 2: As Game 1, but instead of Redact inside RoS we execute the modified algorithm Redact′ which runs on additional input vsk_j and computes π as $\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}_j), \boxed{(\bot, \mathsf{vsk}_j)})$.

Transition - Game 1 → Game 2: A distinguisher $\mathcal{D}^{1\rightarrow 2}$ is a distinguisher for adaptive witness indistinguishability of Π, i.e., $|\Pr[S_2] - \Pr[S_1]| \leq \varepsilon_{\mathsf{wi}}(\kappa)$.

Game 3: As Game 2, but we further modify Redact′ so that it additionally takes ADM as input and works as follows.

```
$\mathsf{Redact}'(\mathsf{gpk}, \mathsf{vpk}_j, \mathsf{m}, \sigma, \mathsf{MOD}, \mathsf{vsk}_j, \mathsf{ADM})$ : Parse $\sigma$ as $((\mathsf{pk}, \sigma_{\mathsf{G}}), (\sigma_{\mathsf{R}}, \mathsf{RED}))$
    and return $(\mathring{\mathsf{m}}, \rho)$, where
        $\boxed{\mathsf{sk_R}} \xleftarrow{R} \mathbb{H}$, $\boxed{\mathsf{pk_R}} \leftarrow \mu(\mathsf{sk_R})$, $\boxed{\mathsf{pk}' \leftarrow \mathsf{pk}^{-1} \cdot \mu(\mathsf{sk_R})}$
        $\boxed{((\mathsf{m}, \sigma'_{\mathsf{R}}), \mathsf{RED}) \leftarrow \mathsf{RS.Sign}((\mathsf{sk_R}, \bot, \mathsf{pk}_{\Lambda}), \mathsf{m}, \mathsf{ADM})}$,
        $((\mathring{\mathsf{m}}, \mathring{\sigma}'_{\mathsf{R}}), \cdot) \leftarrow \mathsf{RS.Redact}(\mathsf{pk_R}, \mathsf{m}, \sigma'_{\mathsf{R}}, \mathsf{MOD}, \mathsf{RED})$,
        $\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}_j), (\bot, \mathsf{vsk}_j))$, and $\rho \leftarrow (\underline{\sigma}, \mathsf{pk}', \mathring{\sigma}'_{\mathsf{R}}, \pi)$.
```

Transition - Game 2 → Game 3: Under adaptability of the RS, Game 2 and Game 3 are perfectly indistinguishable, i.e., $\Pr[S_3] = \Pr[S_2]$.

In Game 3, Redact′ and Sim are identical; RoS is thus independent of b. Thus, the adversary has no advantage in winning the game, i.e., $\Pr[S_3] = 1/2$. Further, we have that $\Pr[S_0] = \Pr[S_1] \leq \Pr[S_2] + \varepsilon_{\mathsf{wi}}(\kappa)$, and that $\Pr[S_3] = \Pr[S_2]$, which yields $\Pr[S_0] \leq 1/2 + \varepsilon_{\mathsf{wi}}(\kappa)$. □

Lemma 11. *If* GS *is anonymous, then Scheme 2 is signer anonymous.*

The proof is identical to the proof of Lemma 5 and therefore not restated here.

Lemma 12. *If* RS *is private and adapts signatures, then Scheme 2 is private.*

Proof. The proof strategy is identical to the privacy proof in the previous section. We however, use the following hybrid to interpolate between the games: We engage with an RS privacy challenger $\mathcal{C}^{\mathsf{p}}_{\kappa}$ in the $\ell + 1$st call to Ch, obtain pk, compute $\sigma_{\mathsf{G}} \leftarrow \mathsf{GS.Sign}(\mathsf{gsk}_i, \mathsf{pk})$, $(\mathring{\mathsf{m}}, \mathring{\sigma}_{\mathsf{R}}) \leftarrow \mathcal{C}^{\mathsf{p}}_{\kappa}.\mathsf{LoRRedact}((\mathsf{m}_0, \textsf{MOD}_0, \textsf{ADM}_0), (\mathsf{m}_1, \textsf{MOD}_1, \textsf{ADM}_1))$, $\mathsf{sk}' \xleftarrow{R} \mathbb{H}$, $\mathsf{pk}' \leftarrow \mu(\mathsf{sk}')$, $(\mathsf{pk}_{\mathsf{R}}, \mathring{\sigma}'_{\mathsf{R}}) \leftarrow \mathsf{RS.Adapt}(\mathsf{pk}, \mathring{\mathsf{m}}, \mathring{\sigma}_{\mathsf{R}}, \mathsf{sk}')$, as well as $\pi \leftarrow \mathsf{\Pi.Proof}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}_j), (\mathsf{sk}', \bot))$, and return $(\mathring{\mathsf{m}}, \underline{\sigma}, \rho) = (\mathring{\mathsf{m}}, (\mathsf{pk}, \sigma_{\mathsf{G}}), ((\mathsf{pk}, \sigma_{\mathsf{G}}), \mathsf{pk}', \mathring{\sigma}'_{\mathsf{R}}, \pi))$ Then, depending on the bit chosen by $\mathcal{C}^{\mathsf{p}}_{\kappa}$, we either simulate Game 0 or Game 1_1 (resp. 1_ℓ or Game $1_{\ell+1}$). $\square$