Composable & Modular Anonymous Credentials: Definitions and Practical Constructions

Jan Camenisch[1], Maria Dubovitskaya[1], Kristiyan Haralambiev*[1], and Markulf Kohlweiss[2]

[1]*IBM Research – Zurich*
[2]*Microsoft Research*

Abstract

It takes time for theoretical advances to get used in practical schemes. Anonymous credential schemes are no exception. For instance, existing schemes suited for real-world use lack formal, composable definitions, partly because they do not support straight-line extraction and rely on random oracles for their security arguments.

To address this gap, we propose *unlinkable redactable signatures* (URS), a new building block for privacy-enhancing protocols, which we use to construct the first efficient UC-secure anonymous credential system that supports multiple issuers, selective disclosure of attributes, and pseudonyms. Our scheme is one of the first such systems for which both the size of a credential and its presentation proof are independent of the number of attributes issued in a credential. Moreover, our new credential scheme does not rely on random oracles.

As an important intermediary step, we address the problem of building a functionality for a complex credential system that can cover many different features. Namely, we design a core building block for a single issuer that supports credential issuance and presentation with respect to pseudonyms and then show how to construct a full-fledged credential system with multiple issuers in a modular way. We expect this flexible definitional approach to be of independent interest.

Keywords: (Fully) structure preserving signatures, vector commitments, anonymous credentials, universal composability, Groth-Sahai proofs.

1 Introduction

Digital signature schemes are a fundamental cryptographic primitive. Besides their use for signing digital items, they are used as building blocks in more complex cryptographic schemes, such as blind signatures [46, 6], group signatures [15, 56], direct anonymous attestation [21], electronic cash [44], voting schemes [52], adaptive oblivious transfer [33, 25], and anonymous credentials [12].

The efficient construction of such protocols, however, demands special properties of a signature scheme, in particular when the protocol needs to achieve strong privacy properties. The most important such properties seem to be that the issuance of a signature and its later use in a protocol is *unlinkable* as well as that the scheme is able to sign *multiple messages* (without employing a hash function). The first signature scheme that met these requirements is a blind signature scheme by Brands [19]. The drawback of blind signatures, however, is that when using the signature later on in a higher-level protocol it must typically be revealed so that a third party can be convinced

*Work done while at IBM Research – Zurich of its validity. Thus, these signatures can be used only once, which turns out to be quite limiting for applications such as group signatures, multi-show anonymous credentials, and compact e-cash [27].

Camenisch and Lysyanskaya [31, 32] were the first to design signature schemes (CL-signatures) overcoming these drawbacks. Their schemes are secure under the Strong RSA or the q-SDH assumption, respectively, and allow for an alternative approach when using a signature in a protocol: Instead of revealing the signature to a third party, the user employs zero-knowledge proofs to convince the third party that she possesses a valid signature from the signer. While in theory this is possible for any signature scheme, CL-signatures were the first that enabled efficient proofs using so-called generalized Schnorr proofs of knowledge. This is due to the algebraic properties of CL-signatures: in particular, no hash function is applied to the message, and the signature and message values are either exponents or group elements.

With the advent of CL-signatures, the area of privacy preserving protocols flourished considerably and numerous new protocols based on them have been proposed. This has also made it apparent, however, that CL-signatures still have a number of drawbacks:

(1) *Random oracles.* To make generalized Schnorr proofs of knowledge non-interactive (which is often required in applications), one needs to resort to the Fiat-Shamir heuristic, that is, to the random oracle model and thus looses all formal provable security guarantees when the random oracle is instantiated by a hash function [37].

(2) *Straight-line extraction.* When designing a protocol to be secure in the Universal Composability model [36] no rewinding can be used to prove security. As a result, witnesses in generalized Schnorr proofs of knowledge need to be encrypted under a public key encoded in the common reference string (CRS). As the witnesses (messages signed with CL-signatures) are discrete logarithms, such encryption is rather expensive [11, 28] and may render the overall protocol unpractical.

(3) *Linear size.* When proving ownership of a CL-signature on many messages, all these messages are needed for the verification of the signature and therefore the proof of possession of a signature will be linear in the number of messages.

A promising ingredient to overcome these drawbacks is the work by Groth and Sahai [50], who for the first time constructed efficient non-interactive zero-knowledge proofs (NIZK) without using random oracles, albeit for a limited set of languages. Indeed, the set of languages covered by these so-called GS-proofs does not include the ones covered by generalized Schnorr protocols and therefore many authors started to look for a compatible CL-signatures replacement, i.e., structure-preserving signature schemes [3, 4, 2]. Together with GS-proofs, these new schemes can also be used as signatures of knowledge [42] and thus are applicable in scenarios previously addressed with CL-signatures.

However, structure-preserving signatures still suffer in terms of performance when signing multiple messages, i.e., drawback (3), which is a typical requirement in applications such as anonymous credentials. Indeed, like for CL-signatures, the size of proofs with structure-preserving signatures grows linearly with the number of messages. As the constant factor for GS-proofs is larger than for generalized Schnorr proofs, structure-preserving signatures loose their attractiveness as a building block for such applications.

Our contribution. In this paper, our goal is to address the three drawbacks of CL-signatures discussed above. To this end, we propose a new type of signature scheme, *unlinkable redactable signatures* (URS), in which one can redact message-signature pairs and reveal only their relevant parts each time they are used. Moreover, signatures in URS are unlinkable and the same message-signature pair can be redacted and revealed multiple times without being linked back to its origin. The real-world efficiency of URS is comparable to that of CL-signatures and, as they overcome the bottleneck of drawback (3), greatly surpasses them in performance when the number of signed messages grows. We view our contribution as threefold:

First, in §2, we formally define URS. We present property-based security definitions for *unlinkability* and *unforgeability* and also a UC functionality for URS. We show that what distinguishes the strength of our property-based and functionality-based definitions is careful key management. Namely, the latter requires a key registration process that allows for the extraction of signing keys. We validate our definitions by showing that an URS scheme satisfying strengthened property-based security definitions with key extraction securely implements our UC functionality.

Second, in §3, we construct an efficient URS scheme from vector commitments [55, 59, 38], structure-preserving signatures [3, 4], and (a minimal dose of) non-interactive proofs of knowledge (NIPoK), which in practice can be instantiated by witness-indistinguishable (!) Groth-Sahai proofs [50]. As we are interested in practical efficiency, we deliberately instantiate our scheme with concrete building blocks, and rely on stronger assumptions (see §4.3).[1] We show how to make use of algebraic properties in our building blocks to minimize the witness size of the NIPoK. For key management, we employ a novel cryptographic primitive, *fully* structure preserving signatures [1]. Our main target is Groth-Sahai proofs, but we also discuss extensions for optimizing the efficiency of generalized Schnorr proofs of knowledge in case one is willing to accept random oracles.

Third, in §4, to demonstrate the versatility of our URS scheme as a CL-signature 'replacement', we employ it to design the first efficient universally composable anonymous credential system that supports multiple issuers, pseudonyms, and selective disclosure of attributes.

Anonymous credential systems usually need to support an ecosystem of different features. Therefore, a single ideal functionality providing all the features, such as pseudonyms, selective attribute disclosure, predicates over attributes, revocation, inspection, etc. would be very complex and hard to both create and use in a modular way—not to mention credible security proofs.

Nevertheless, ideal functionalities are very attractive for modeling the complexity of anonymous credential schemes. Indeed an early seminal paper [30] attempted exactly this, but was foiled by drawback (2)—as well as by the immaturity of the UC framework at the time. To overcome this complexity, we present a flexible and modular approach for constructing UC-secure anonymous credentials. Namely, we design a core building block for a single issuer that supports credential issuance and presentation with respect to pseudonyms. We then show how to compose multiple such blocks to construct in a modular way a full-fledged credential system with multiple issuers.

Besides being composable, our system is also arguably one of the first schemes to support efficient non-interactive attribute disclosure with cost independent of the number of attributes issued without having to rely on random oracles. Even in the random oracle model this has been an elusive goal. Therefore, because of the composability and efficient selective disclose, our scheme is very attractive and quickly surpasses schemes based on blind signatures and CL-signatures [20, 32] when the number of attributes grows.

Related work. We compare our signatures and credential schemes with other related work. As there are a multitude of papers on redactable, quotable, and sanitizable signatures [51, 22, 61, 7], we focus on the most influential definitional work and the most promising approaches in terms of efficiency. Further comparison can be found in §5.

A variety of signature schemes with flexible signing capabilities and strong privacy properties have been proposed [41, 18, 8, 7, 10, 14, 35]. While these works provide a fresh definitional approach, their schemes are very inefficient, especially when redacting a message vector with a large number of attributes. Some schemes built on vector commitments [55, 58] achieve bet-

[1]Our modular approach facilitate the construction of 'paranoid' instantiations, e.g., using the CDH-based vector commitment scheme of [38].

ter efficiency but only consider one-time-show credentials, and while the former is not defined formally, the latter involves interaction.

The first efficient multi-show anonymous credential scheme is [30]. It was extended with efficient attribute disclosure [26] and had real-world exposure [34, 21]. It can, however, only be non-interactive in the random oracle model. Non-interactive credentials in the standard model have been built from P-signatures [12, 13]. An instantiation of our URS scheme, however, is almost twice more efficient than [12] despite the fact that the latter does not support signing multiple messages. Izabachène et al. [54] extends the work of [12] with vector commitments; their scheme is, however, not secure under our definitions. In independent work, Hanser and Slamanig [62] present a credential system with efficient (independent of the number of attributes) attribute disclosure. However, their system is only secure in the generic group model [48]. Furthermore, it uses hash function to encode attributes and thus does not enable efficient protocol design. None of these schemes is (universally) composable.

An important factor that is often neglected is the compatibility of schemes with zero-knowledge proofs to enable efficient protocol design. Because of its compatibility with Groth-Sahai proofs, efficiency and composability, immediate further applications of our URS scheme include efficient e-cash, credential-based key exchange, e-voting, auditing, and others.

2 Definitions of Unlinkable Redactable Signatures

Redactable signatures are an instance of homomorphic [7] or controlled-malleable signatures [41]. For our credentials application the most useful redaction operation is to selectively white-list or quote a subset of messages and their positions from a message vector of length n ([7] consider the quoting of sub-sentences). We denote the message space of all valid message vectors as $\mathcal{M}$. We also refer to the redacted message as a quote of the original message. To distinguish the original vector from the quote of all messages we denote the original vector as $\vec{m} = (1, m_1, \ldots, m_n)$ and a quote as $\vec{m}_I = (2, m'_1, \ldots, m'_n)$. We represent each valid quoting transformation by a set $I \subseteq [1, n]$ of message positions and denote quoting either by $I(\vec{m})$ or $\vec{m}_I$. We denote the i$^{\text{th}}$ message element either by $\vec{m}[i]$ or m_i. A quote $\vec{m}_I$ from $\vec{m}$ is of the form

$$\vec{m}_I[i] = m'_i = \begin{cases} m_i & i \in I \\ \perp & \text{otherwise} \end{cases} .$$

Note that the message itself already reveals whether it is a quote. Chase et al. [41] call such a scheme tiered and we refer to the vectors $\vec{m}$ and $\vec{m}_I$ as Tier 1 and Tier 2 vectors respectively. The vector $\vec{m}_I$ can be sparse and can have a much shorter encoding than $\vec{m}$.

2.1 Property-Based Definitions for Unlinkable Redactable Signatures

One can define the security of redactable signatures by instantiating controlled-malleable signature definitions for simulatability, simulation unforgeability, and simulation context-hiding of Chase et al. [41] with the quoting transformation class $\mathcal{T} = \{I(\cdot) | I \subseteq [1..n]\}$ above. We prefer, however, to give our own unforgeability and unlinkability definitions that are more specific and do not rely on simulation and extraction. This makes them simpler and easier to prove, and thus more efficiently realizable. Together with key extractability they are nevertheless sufficient to realize the strong guarantees of our UC functionality.

Definition 1 (Unlinkable Redactable Signatures)**.** *An unlinkable redactable signature scheme* URS *consists of the following algorithms:*

$\mathsf{URS.SGen}(1^\kappa) \rightarrow SP$. SGen *takes the security parameter* 1^κ *as input and outputs the system parameters* SP.

$\mathsf{URS.Kg}(SP) \rightarrow (pk, sk)$. Kg *takes the system parameters* SP *as an input and outputs public verification and private signing keys* (pk, sk). *The verification key* pk *defines the message space* $\mathcal{M}$.

$\mathsf{URS.Sign}(sk, \vec{m}) \rightarrow \sigma$. Sign *takes the signing key* sk *and a message* $\vec{m} \in \mathcal{M}$ *as input and produces a signature* σ.

$\mathsf{URS.Derive}(pk, I, \vec{m}, \sigma) \rightarrow \sigma_I$. Derive *takes the public key* pk*, a selection vector* I*, a message* $\vec{m}$ *and a signature* σ *(both of Tier 1) as input. It produces a Tier 2 signature* σ_I *for* $\vec{m}_I$.

$\mathsf{URS.Verify}(pk, \sigma, \vec{m}) \rightarrow 0/1$. Verify *takes the verification key* pk*, a signature* σ*, and a message* $\vec{m}$ *of Tier 1 or Tier 2 as input and checks the signature.*

We omit the URS qualifier when it is clear from the context.

Correctness. Informally, correctness requires that for honestly generated keys, both honestly generated and honestly derived signatures must always verify.

Unforgeability. Unforgeability captures the requirement that an attacker, who is given Tier 1 and Tier 2 signatures on messages of his choice, should not be able to produce a signature on a message that is not derivable from the set of signed messages in his possession. More formally:

Definition 2 (Unforgeability)**.** *Let* H *be a stateful handle generator. For a redactable signature scheme* $\mathsf{URS}.\{\mathsf{SGen}, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Derive}, \mathsf{Verify}\}$*, tables* Q_1, Q_2, Q_3*, and an adversary* $\mathcal{A}$*, consider the following game:*

- *Step 1.* $SP \leftarrow \mathsf{SGen}(1^k); (pk, sk) \xleftarrow{\$} \mathsf{Kg}(SP)$*;* $Q_1, Q_2, Q_3 \leftarrow \emptyset$.
- *Step 2.* $(\vec{m}_I^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}(\cdot), \mathcal{O}_{\mathsf{Derive}}(\cdot,\cdot), \mathcal{O}_{\mathsf{Reveal}}(\cdot)}(pk)$*, where* $\mathcal{O}_{\mathsf{Sign}}$*,* $\mathcal{O}_{\mathsf{Derive}}$*, and* $\mathcal{O}_{\mathsf{Reveal}}$ *behave as follows:*

```
$\mathcal{O}_{\mathsf{Sign}}(\vec{m})$
$h \leftarrow \mathsf{H}$; $\sigma \leftarrow \mathsf{Sign}(sk, \vec{m})$
add $(h, \vec{m}, \sigma)$ to $Q_1$
return $h$

$\mathcal{O}_{\mathsf{Derive}}(h, I)$
if $(h, \vec{m}, \sigma) \in Q_1$
    $\sigma' \leftarrow \mathsf{Derive}(pk, I, \vec{m}, \sigma)$
    add $\vec{m}_I$ to $Q_2$; return $\sigma'$

$\mathcal{O}_{\mathsf{Reveal}}(h)$
if $(h, \vec{m}, \sigma) \in Q_1$
    add $\vec{m}$ to $Q_3$
    return $\sigma$
```

A signature scheme URS *satisfies* unforgeability *if for all such PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that in the above game the probability (over the random choices of* Kg*,* Sign*,* Derive *and* $\mathcal{A}$*) that* $\mathsf{Verify}(pk, \sigma^*, \vec{m}_I^*) = 1$ *and* $\forall \vec{m} \in Q_3$*,* $\vec{m}_I^* \neq \vec{m}_I$*, and* $\vec{m}_I^* \notin Q_2$ *is less than* $\nu(\kappa)$.

Note that we do not consider a Tier 1 signature itself as a forgery. However, if the adversary manages to produce a valid Tier 1 signature on a message $\vec{m}$ without calling $\mathsf{Sign}(\vec{m})$ and either $\mathsf{Reveal}(h)$ or $\mathsf{Derive}(h, I)$ on all subsets $I \subseteq [1..n]$ for the corresponding handle h, he can use this Tier 1 signature to break unforgeability by deriving a Tier 2 signature from it.

Unlinkability. Informally, unlinkability ensures that an adversarial signer cannot distinguish which of two Tier 1 signatures of his choosing was used to derive a Tier 2 signature. More formally:

Definition 3 (Unlinkability)**.** *For the signature scheme* $\mathsf{URS}.\{\mathsf{SGen}, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Derive}, \mathsf{Verify}\}$ *and a stateful adversary* $\mathcal{A}$*, consider the following game:*

- *Step 1.* $SP \leftarrow \mathsf{SGen}(1^k)$.

- *Step 2.* $(pk, I, \vec{m}^{(0)}, \vec{m}^{(1)}, \sigma^{(0)}, \sigma^{(1)}) \xleftarrow{\$} \mathcal{A}(SP)$*, where* $\vec{m}_I^{(0)} = \vec{m}_I^{(1)}$*,* $\mathsf{Verify}(pk, \sigma^{(0)}, \vec{m}^{(0)}) = 1$*, and* $\mathsf{Verify}(pk, \sigma^{(1)}, \vec{m}^{(1)}) = 1$.
- *Step 3. Pick* $b \leftarrow \{0,1\}$ *and form* $\sigma_I^{(b)} \xleftarrow{\$} \mathsf{Derive}(pk, I, \vec{m}^{(b)}, \sigma^{(b)})$.
- *Step 4.* $b' \xleftarrow{\$} \mathcal{A}(\sigma_I^{(b)})$.

The signature scheme URS *is unlinkable if for any polynomial time adversary* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that* $\Pr[b = b'] < \frac{1+\nu(\kappa)}{2}$.

Note that this definition is very strong, as the adversary can even pick pk.

2.2 Ideal Functionality for Unlinkable Redactable Signatures

We now give an alternative characterization of unlinkable redactable signatures using an ideal functionality $\mathcal{F}_{\mathsf{URS}}$ defined as follows:

Functionality $\mathcal{F}_{\mathsf{URS}}$

The functionality maintains tables $\mathcal{K}$ and $\mathcal{Q}$ initialized to $\emptyset$ and flags kg and $keyleak$ which are initially unset.

- On input $(\mathtt{keygen}, sid)$ from S, verify that $sid = (S, sid')$ for some sid' and that flag kg is unset. If not, then return $\perp$. Else, send $(\mathtt{initF}, sid)$ to $\mathcal{SIM}$ and wait for a message $(\mathtt{initF}, sid, SP, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Derive}, \mathsf{Verify})$ from $\mathcal{SIM}$, where Kg, Sign, and Derive are PPT algorithms and Verify is a deterministic algorithm. Then, store SP, Kg, Sign, Derive, and Verify, run $(pk, sk) \leftarrow \mathsf{Kg}(SP)$, set flag kg, store (pk, sk), and return $(\mathtt{verificationKey}, sid, pk)$ to S.
- On input $(\mathtt{checkPK}, sid, pk')$ from some party P, verify that the flag kg is set. Check whether $pk' = pk$ or whether (pk', sk') for some sk' was stored in $\mathcal{K}$. In this case, return $(\mathtt{checkedPK}, sid, true)$. Else, if $(pk', \perp)$ was stored in $\mathcal{K}$ return $(\mathtt{checkedPK}, sid, false)$. Else, send $(\mathtt{checkPK}, sid, pk')$ to $\mathcal{SIM}$, wait for $(\mathtt{checkedPK}, sid, sk')$ from $\mathcal{SIM}$, add (pk', sk') to $\mathcal{K}$. If $sk' \neq \perp$, return $(\mathtt{checkedPK}, sid, \mathsf{true})$ to P. Otherwise, return $(\mathtt{checkedPK}, sid, \mathsf{false})$ to P.
- On input $(\mathtt{leakSK}, sid)$ from S verify that $sid = (S, sid')$ for some sid'. If not, return $\perp$. Else, if flag kg is set, set flag $keyleak$ and return $(\mathtt{leakSK}, sid, sk)$.
- On input $(\mathtt{sign}, sid, \vec{m})$ from S, verify that $sid = (S, sid')$ for some sid' and that the flag kg is set. If not, return $\perp$. Else, run $\sigma \leftarrow \mathsf{Sign}(sk, \vec{m})$ and $\mathsf{Verify}(pk, \sigma, \vec{m})$. If Verify is successful, return $(\mathtt{signature}, sid, \vec{m}, \sigma)$ to S and add $\vec{m}$ to $\mathcal{Q}$, otherwise return $\perp$.
- On input $(\mathtt{derive}, sid, pk', I, \vec{m}, \sigma)$ from some party P, run $\mathsf{Derive}(pk', I, \vec{m}, \sigma)$ and if it fails, return $\perp$. Otherwise, if the flag kg is set and $pk = pk'$ then set $sk_{tmp} = sk$. If there is an entry $(pk', sk') \in \mathcal{K}$ recorded, set $sk_{tmp} = sk'$. If sk_{tmp} was set run $\sigma' \leftarrow \mathsf{Sign}(sk_{tmp}, \mathsf{Zero}(\vec{m}, I))$ and return $\mathsf{Derive}(pk', I, \mathsf{Zero}(\vec{m}, I), \sigma')$. Otherwise, return the output of $\mathsf{Derive}(pk', I, \vec{m}, \sigma)$.
- On input $(\mathtt{verify}, sid, pk', \sigma, \vec{m}_I)$ from some party P, compute $result \leftarrow \mathsf{Verify}(pk', \sigma, \vec{m}_I)$. If the flag kg is set, $pk' = pk$, flag $keyleak$ is not set, and $\nexists\ \vec{m} \in \mathcal{Q}$ such that $\vec{m}_I = I(\vec{m})$, then output $(\mathtt{verified}, sid, \vec{m}_I, 0)$. Otherwise, output $(\mathtt{verified}, sid, \vec{m}_I, result)$.

We point out some aspects of the ideal functionality. The functionality needs to output concrete values as signatures of messages and redacted signatures, as well as key material. To generate and verify these values, $\mathcal{F}_{\mathsf{URS}}$ requires the adversary/simulator $\mathcal{SIM}$ to provide it with a number of polynomial-time algorithms. This is in line with how ideal functionalities for signatures, and in particular blind signatures, have been defined before [6, 36, 46, 53, 57]. We consider static corruptions of protocol machines, but allow the simulator to request the signing key at any time by sending the `leakSK` message. This allows us to ensure that the privacy properties for users are still enforced even if the signer leaks its secret key. The functional and security properties are enforced by the functionality no matter how these (adversarial) algorithms compute the values. Unforgeability is enforced by the fact that $\mathcal{F}_{\mathsf{URS}}$ will output false (0) for verification queries for which the message (or a corresponding original message) has not been signed, provided that the signer is not corrupted and the signing key not leaked. (If the signer is corrupted statically, $(\texttt{keygen}, sid)$ will not be sent and hence kg not set and unforgeability not enforced.) Unlinkability of redacted signature is enforced by $\mathcal{F}_{\mathsf{URS}}$ as follows. It generates a fresh redacted signature only from those parts of the original message that are quoted, i.e., the hidden message parts are set to zero, and thus redacted signatures from $\mathcal{F}_{\mathsf{URS}}$ do not contain any information about the hidden parts of messages. More precisely, this is enforced for the keys generated by $\mathcal{F}_{\mathsf{URS}}$, as well as for any keys that an honest party successfully checked before generating a redacted signature. Unless mentioned otherwise, the reply of the functionality upon a failed check or verification is $\perp$. Finally, we define $\mathsf{Zero}(\vec{m}, I) = (1, \tilde{m}_1, \ldots, \tilde{m}_n)$, with $\tilde{m}_j = m_j$ for $j \in I$ and $\tilde{m}_j = 0$ for $j \notin I$. This should not be confused with the operator I that outputs a Tier 2 message with $\perp$ elements on certain positions.

2.3 Key Registration and UC Realizability

We now want to construct a protocol $\mathcal{R}_{\mathsf{URS}}$ that realizes $\mathcal{F}_{\mathsf{URS}}$ using a URS scheme in the $\mathcal{F}_{\mathsf{CRS}}$-hybrid model where SP is the reference string and each call to $\mathcal{F}_{\mathsf{URS}}$ is essentially replaced by running one of the algorithms of URS. While this can be done (the detailed description of $\mathcal{R}_{\mathsf{URS}}$ is given in §A), there are a number of hurdles that need to be overcome. These hurdles are quite typical and include, e.g., that we need to be able to extract the secret keys from the adversary to be able to simulate properly. They are, however, often treated only informally in security proofs. Here we want to make them explicit and treat them formally correct. So our goal is to prove a statement (Theorem) of the form:

> *If* URS *is correct, unforgeable, unlinkable, and X then $\mathcal{R}_{\mathsf{URS}}$ securely realizes $\mathcal{F}_{\mathsf{URS}}$ in the $\mathcal{F}_{\mathsf{CRS}}$-hybrid model.*

What do we have to require from X to make this theorem true? To prove the theorem we have to show indistinguishability between the ideal world and the real world. In the ideal world, an environment $\mathcal{Z}$ interacts with the simulator $\mathcal{SIM}$ and functionality $\mathcal{F}_{\mathsf{URS}}$. In the real world, the environment $\mathcal{Z}$ interacts with the real adversary $\mathcal{A}$ and the protocol $\mathcal{R}_{\mathsf{URS}}$.

We provide a tentative description of $\mathcal{SIM}$ in the ideal world: when receiving the $(\texttt{initF}, sid)$ message from $\mathcal{F}_{\mathsf{URS}}$, it generates a trapdoor td (in addition to SP) and returns $(\texttt{initF}, sid, SP, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Derive}, \mathsf{Verify})$. On receiving the $(\texttt{checkPK}, sid, pk)$ message, is uses the trapdoor to extract the secret key sk and returns sk to $\mathcal{F}_{\mathsf{URS}}$.

To make this work, we extend URS with several algorithms: CheckPK is run by $\mathcal{R}_{\mathsf{URS}}$ on receiving a message $(\texttt{checkPK}, sid, pk)$. SGenT and ExtractKey are the trapdoored parameter generation and key extraction algorithm for $\mathcal{SIM}$. CheckKeys is used to define what it means to extract a valid key.

URS.CheckPK$(pk) \to 0/1$. CheckPK *is a deterministic algorithm that takes a public key pk as an input and checks that it is correctly formed. It outputs* 1 *if pk is correct, and* 0 *otherwise.*

URS.SGenT$(1^\kappa) \to (SP, td)$. SGenT *is a system parameters generation algorithm that takes the security parameter* 1^κ *as input and outputs the system parameters SP and a trapdoor td for the key extraction algorithm.*

URS.ExtractKey$(pk, td) \to sk$. ExtractKey *is an algorithm that takes a public key pk and a trapdoor td as input. It extracts the corresponding secret key sk.*

URS.CheckKeys$(pk, sk) \to 0/1$. CheckKeys *is an algorithm that takes a public pk and a private sk keys and checks if they constitute a valid signing key pair. It outputs* 1 *if they do, and* 0 *otherwise.*

Strengthened Correctness requires that honestly generated keys, but also keys for which predicate CheckKeys(pk, sk) holds can be used to create signatures that will verify. It moreover guarantees that $\mathsf{CheckPK}(pk)$ holds for honest public keys.

Definition 4 (Correctness)**.** *An unlinkable redactable signature scheme* $\mathsf{URS}.\{\mathsf{SGen}, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Derive}, \mathsf{Verify}\}$ *with additional algorithm* $\mathsf{CheckPK}$ *is correct if there exists a negligible function* $\nu(\cdot)$ *such that for any message* $\vec{m} \in \mathcal{M}$ *and* $I \subseteq [1, n]$: $\Pr[(SP) \leftarrow \mathsf{SGen}(1^k);\ (sk, pk) \leftarrow \mathsf{Kg}(SP);\ \sigma \leftarrow \mathsf{Sign}(sk, \vec{m});\ \sigma_I \leftarrow \mathsf{Derive}(pk, I, \vec{m}, \sigma) :\ \mathsf{CheckPK}(pk) = 0 \vee \mathsf{Verify}(pk, \sigma, \vec{m}) = 0 \vee \mathsf{Verify}(pk, \sigma_I, \vec{m}_I) = 0] < \nu(\kappa)$.

The additional $\mathsf{CheckKeys}$ *algorithm is correct if for any message* $\vec{m} \in \mathcal{M}$ *and* $I \subseteq [1, n]$ *and all* $pk \in \{0, 1\}^*$, $sk \in \{0, 1\}^*$ *there exists a negligible function* $\nu(\cdot)$ *such that* $\Pr[\sigma \leftarrow \mathsf{Sign}(sk, \vec{m});\ \sigma_I \leftarrow \mathsf{Derive}(pk, I, \vec{m}, \sigma) :\ \mathsf{CheckKeys}(pk, sk) = 1 \wedge (\mathsf{Verify}(pk, \sigma, \vec{m}) = 0 \vee \mathsf{Verify}(pk, \sigma_I, \vec{m}_I) = 0)] < \nu(\kappa)$.

Parameter Indistinguishability. Informally, parameter indistinguishability ensures that the SP produced by SGenT and SGen are computationally indistinguishable. It is formally defined as follows:

Definition 5 (Parameter Indistinguishability)**.** *A redactable signature scheme* $\mathsf{URS}.\{\mathsf{SGen}, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Derive}, \mathsf{Verify}\}$ *with alternative parameter generation* SGenT *is parameter indistinguishable if for any polynomial time adversary* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that* $\Pr[(SP_0, td) \leftarrow \mathsf{SGenT}(1^k);\ SP_1 \leftarrow \mathsf{SGen}(1^k);\ b \leftarrow \{0, 1\};\ b' \leftarrow \mathcal{A}(SP_b) : b = b'] < \frac{1+\nu(\kappa)}{2}$.

Key Extractability. Informally, the key extractability ensures that if SGenT was run and if $\mathsf{CheckPK}$ outputs 1, then the extraction algorithm $\mathsf{ExtractKey}(pk, td)$ will output a valid secret key sk, i.e. $\mathsf{CheckKeys}(pk, sk) = 1$.

Definition 6 (Key Extractability)**.** *A redactable signature scheme* $\mathsf{URS}.\{\mathsf{SGen}, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Derive}, \mathsf{Verify}\}$ *with additional algorithms* $(\mathsf{CheckPK},\ \mathsf{SGenT},\ \mathsf{CheckKeys}, \mathsf{ExtractKey})$ *is key extractable if* $\mathsf{CheckKeys}$ *is correct according to Definition 4 and for any polynomial time adversary* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that* $\Pr[(SP, td) \leftarrow \mathsf{SGenT}(1^k);\ pk \leftarrow \mathcal{A}(SP, td);\ sk \leftarrow \mathsf{ExtractKey}(pk, td) : (\mathsf{CheckPK}(pk) = 1 \wedge \mathsf{CheckKeys}(pk, sk) = 0))] < \nu(\kappa)$.

Composable Unlinkability holds even when parameters in the unlinkability game are generated using $(SP, td) \leftarrow \mathsf{SGenT}(1^k)$ and $\mathcal{A}$ is handed td. This allows for the use of the game in a hybrid argument when proving the security of the simulator. We note that in such an adapted unlinkability game the trapdoor td must only enable key-extraction, and crucially does not allow the adversary to extract a Tier 1 signature from a Tier 2 signature (this would break unlinkability). In our instantiation this is achieved by splitting SP into several parts. The trapdoor is only generated for the part used for key extraction.

UC realization. We prove that if an unlinkable redactable signature URS is correct, parameter indistinguishable, key extractable, unforgeable, and unlinkable, then $\mathcal{R}_{\mathsf{URS}}$ securely realizes $\mathcal{F}_{\mathsf{URS}}$. More formally, we have the following theorem (which is proven in §A).

Theorem 1. *Let* URS *be an unlinkable redactable signature scheme. If* URS *is correct, parameter indistinguishable, key-extractable, unforgeable, and composable unlinkable according to Definitions 4, 5, 6, 2, and 3, respectively, then* $\mathcal{R}_{\mathsf{URS}}$ *securely realizes* $\mathcal{F}_{\mathsf{URS}}$ *in the* $\mathcal{F}_{\mathsf{CRS}}$*-hybrid model.*

3 The Construction of Our Redactable Signature Scheme

As a first step toward our full solution, we will construct an unforgeable and unlinkable URS scheme without key extraction. The scheme should be of independent interest, in case universal composability is not a design requirement. This isolation of key extraction, seemingly only needed for universal composition, is a nice feature of our definitions.

Let $\mathcal{G}$ be a bilinear group generator that takes as an input security parameter 1^κ and outputs the descriptions of multiplicative groups $grp = (p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_t, e, G, \tilde{G})$ where $\mathbb{G}$, $\tilde{\mathbb{G}}$, and $\mathbb{G}_t$ are groups of prime order p, e is an efficient, non-degenerating bilinear map $e : \mathbb{G} \times \tilde{\mathbb{G}} \to \mathbb{G}_t$, and G and $\tilde{G}$ are generators of the groups $\mathbb{G}$ and $\tilde{\mathbb{G}}$, respectively.

Our construction makes use of a structure preserving signature (SPS) scheme $\mathsf{SPS}.\{\mathsf{Kg}, \mathsf{Sign}, \mathsf{Verify}\}$ and a vector commitment scheme $\mathsf{VC}.\{\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Check}\}$. We recall that the structure-preserving property of the signature scheme requires that verification keys, messages, and signatures are group elements and the verification predicate is a conjunction of pairing product equations. The intuition behind our construction is susceptible simple: Use $\mathsf{SPS.Kg}$ to generate a signing key pair and $\mathsf{VC.Setup}$ to add commitment parameters to the public key. To sign a vector $\vec{m}$, first, commit to $\vec{m}$ and then sign the resulting commitment C. To derive a quote for a subset I of the messages, simply open the commitment to the messages in $\vec{m}_I$. We verify a signature on a quote by verifying both the structure-preserving signature ($\mathsf{SPS.Verify}$) and checking the opening of the commitment ($\mathsf{VC.Check}$).

Such a scheme has, however, several shortcomings. First, it is linkable, as the same commitment is reused across multiple quotes of the same message. Even if both the underlying SPS scheme and the commitment scheme are individually re-randomizable, this seems hard to avoid as the unforgeability of the SPS scheme prevents randomization of the message. Second, such a construction is only heuristically secure. Existing vector commitments do not guarantee that multiple openings cannot be combined and mauled into an opening for a different sub-vector. We call vector commitment schemes that prevent this *opening non-malleable*. (Recently, [62] constructed an SPS scheme allowing for randomization within an equivalence class. However, their commitments cannot be opened to arbitrary vectors of $\mathbb{Z}_p$ and are not provably opening non-malleable.)

Our main design goal is to address both of these weaknesses while avoiding a large performance overhead. Our main tool for this is an efficient non-interactive proof-of-knowledge. Intuitively, we hide the commitments and their openings, as well as a small part of the signature to achieve unlinkability. Hiding the commitment opening also helps solve the malleability problems for commitments. To achieve real-world efficiency we show how to exploit the re-randomizability of the SPS (and optionally the commitment scheme as described in §C.2).

Before describing our redactable signature scheme in more detail, we present a vector commitment scheme that uses a variant of polynomial commitments from [55]. While our changes are partly cosmetic, they simplify the assumption needed for opening non-malleability.

3.1 Vector Commitments Simplified

A vector of messages $\vec{m} \in \mathbb{Z}_p^n$ is committed using a polynomial $f(x)$ that has a value $f(i) = m_i$ at the position i. In Lagrange form such a polynomial is a linear combination $f(x) = \sum_{i=1}^n m_i f_i(x)$ of Lagrange basis polynomials $f_i(x) = \prod_{j=0, j\neq i}^n \frac{x-j}{i-j}$. To batch-open a vector commitment for a position set $I \subseteq \{1, \ldots, n\}$, one uses a polynomial $f_I(x) = \sum_{i \in I} m_i f_i(x)$. For such a polynomial, it holds that $f_I(i) = m_i$ for $i \in I$; and $f_I(0) = 0$. (The additional root at 0 is necessary to achieve opening non-malleability). The reuse of the same Lagrange basis polynomials, which yields polynomials of not the lowest possible degree, reduces the number of variable bases in the equation of Check below and increases efficiency when used for the construction of bigger protocols such as anonymous credential.) Also, note that $f(x) - f_I(x)$ is divisible by the polynomial $p_I(x) = x \cdot \prod_{i \in I}(x - i)$. We use the polynomial $p(x) = x \cdot \prod_{i=1}^{n}(x - i)$ which is divisible by $p_I(x)$ for any $I \subseteq \{1, \ldots, n\}$ to randomize commitments to make them perfectly hiding.

Construction. We reuse the notation of §2 and use Tier 1 vectors $\vec{m}$ for the vectors being committed and Tier 2 vectors $\vec{m}_I$ for batch openings at positions I. We also let $grp = (p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_t, e, G, \tilde{G})$ be bilinear map parameters generated by a bilinear group generator $\mathcal{G}(1^\kappa)$.

$\mathsf{VC.Setup}(grp)$. Pick $\alpha \leftarrow \mathbb{Z}_p$ and compute $(G_1, \tilde{G}_1, \ldots, G_{n+1}, \tilde{G}_{n+1})$, where $G_i = G^{(\alpha^i)}$ and $\tilde{G}_i = \tilde{G}^{(\alpha^i)}$. Output parameters $pp = (grp, G_1, \tilde{G}_1, \ldots, G_{n+1}, \tilde{G}_{n+1})$. Values $G_1, \ldots, G_{n+1}$ suffice to compute $G^{\phi(\alpha)}$ for any polynomial $\phi(x)$ of maximum degree $n + 1$ (and similarly for $\tilde{G}^{\phi(\alpha)}$).

Furthermore, for the above defined $f_i(x)$, $p(x)$, and $p_I(x)$, we implicitly define $F_i = G^{f_i(\alpha)}$, $P = G^{p(\alpha)}$, $P_I = G^{p_I(\alpha)}$, and $\tilde{P}_I = \tilde{G}^{p_I(\alpha)}$. These group elements can be computed from the parameters pp.

$\mathsf{VC.Commit}(pp, \vec{m}, r)$. Output $C = \prod_{i=1}^{n} F_i^{m_i} P^r$.

$\mathsf{VC.Open}(pp, I, \vec{m}, r)$. Let $w(x) = \frac{f(x) - f_I(x) + r \cdot p(x)}{p_I(x)}$ and compute the witness $W = G^{w(\alpha)}$ using parameters pp.

$\mathsf{VC.Check}(pp, C, \vec{m}_I, W)$. Accept if $e(C, \tilde{G}) = e(W, \tilde{P}_I) e(\prod_{i \in I} F_i^{m_i}, \tilde{G})$.

Note that $p_I(x)$ always has the factor x. This is essential for achieving opening non-malleability. If $p_I(x)$ would be 1 for $I = \emptyset$, as in the original polynomial commitment scheme of [55], then C would be a valid batch opening witness for the empty set of messages.

Security analysis. We require the commitment scheme to be *complete*, *batch binding*, and *opening non-malleable*. Completeness is standard for a commitment scheme follows easily from the following equation: $e(C, \tilde{G}) = e(G, \tilde{G})^{f(\alpha) + r \cdot p(\alpha)} = e(G, \tilde{P}_I)^{\frac{f(\alpha) - f_I(\alpha) + r \cdot p(\alpha)}{p_I(\alpha)}} e(G, \tilde{G})^{f_I(\alpha)} = e(W, \tilde{P}_I) e(\prod_{i \in I} F_i^{m_i}, \tilde{G})$.

Next, we define the batch binding and opening non-malleability properties:

Definition 7 (Batch binding)**.** *For a vector commitment scheme* $\mathsf{VC.\{Setup, Commit, Open, Check\}}$ *and an adversary* $\mathcal{A}$ *consider the following game:*

- *Step 1.* $grp \leftarrow \mathcal{G}(1^\kappa)$ *and* $pp \leftarrow \mathsf{VC.Setup}(grp)$
- *Step 2.* $C, \vec{m}_I, W, \vec{m}'_{I'}, W' \leftarrow \mathcal{A}(pp)$

Then, the commitment scheme satisfies batch binding *if for all such PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that the probability (over the choices of* $\mathcal{G}$, Setup, *and* $\mathcal{A}$*) that* $1 = \mathsf{VC.Check}(pp, C, \vec{m}_I, W) = \mathsf{VC.Check}(pp, C, \vec{m}'_{I'}, W')$ *and that there exist* $i \in I \cap I'$ *such that* $m_i \neq m'_i$ *is at most* $\nu(\kappa)$*. (Note that* $\vec{m}_I$ *and* $\vec{m}'_{I'}$ *are Tier 2 vectors, and thus encode the sets* I *and* I' *respectively.)*

Definition 8 (Opening non-malleability)**.** *For a vector commitment scheme* $\mathsf{VC.\{Setup, Commit, Open, Check\}}$ *and an adversary* $\mathcal{A}$ *consider the following game:*

- *Step 1.* $grp \leftarrow \mathcal{G}(1^\kappa)$ *and* $pp \leftarrow \mathsf{VC.Setup}(grp)$
- *Step 2.* $\vec{m}, I \leftarrow \mathcal{A}(pp)$
- *Step 3. Pick random* r*, compute* $C \leftarrow \mathsf{VC.Commit}(pp, \vec{m}, r)$*, and* $W \leftarrow \mathsf{VC.Open}(pp, I, \vec{m}, r)$.
- *Step 4.* $W', I' \leftarrow \mathcal{A}(C, W)$

Then the commitment scheme satisfies opening non-malleability *if for all such PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that the probability (over the choices of* $\mathcal{G}$, Setup, Commit, *and* $\mathcal{A}$ *that* $1 = \mathsf{VC.Check}(pp, C, \vec{m}_{I'}, W')$*, and* $I \neq I'$ *is at most* $\nu(\kappa)$.

In the following theorems we make use of the n-BSDH assumption [49] and the J-RootDH assumption which is defined next. See §F for its generic group model proof. (We note that this assumption is only required for opening non-malleability, which is ignored by most existing constructions of anonymous credentials from vector commitments.)

Definition 9 (J-RootDH Assumption)**.** *Let J be a subset of $[1..n]$, let $\alpha, r \in \mathbb{Z}_p^*$, and $X = (G^{\alpha \cdot \prod_{i=1}^{n}(\alpha-i)})^r$, $Y = (G^{\prod_{i \in J}(\alpha-i)})^r$. For all PPT algorithms $\mathcal{A}$, the probability $\Pr[\mathcal{A}(G, \tilde{G}, \{G^{\alpha^i}, \tilde{G}^{\alpha^i}\}_{i=1}^{n+1}, X, Y) = J', Z]$ such that $Z = (G^{\prod_{i \in J'}(\alpha-i)})^r$ and $J' \subseteq [1..n]$ and $J' \neq J$, is at most a negligible function $\nu(\kappa)$.*

Theorem 2. *The commitment scheme* VC*defined above is batch binding under the $(n+1)$-BSDH assumption.* The proof is similar to that of [55] and is found in §B.3.

Theorem 3. *The commitment scheme* VC *defined above is opening non-malleable under the J-RootDH assumption.* The proofs can be found in §B.3.

Randomizable vector commitments. In §B.4 we consider an algorithm $\mathsf{VC.Rand}(pp, C, W, I, \rho)$ for randomizing commitments and openings. It randomizes the commitment as $C' = C \cdot P_I^{\rho}$ and the witness as $W' = W \cdot G^{\rho}$. These elements then look random, conditioned on them satisfying the Check equations for a particular message $\vec{m}_I$. We show that the relaxed versions of the above properties that we call *randomized batch binding* and *randomized opening non-malleability* hold under the $(n+1)$-BSDH and J-RootDH assumptions respectively.

3.2 Non-interactive Zero-Knowledge and Witness Indistinguishable Proof Systems

Let R be an efficiently computable binary relation. For pairs $(W, \mathit{Stmt}) \in R$ we call Stmt the statement and W the witness. Let $\mathcal{L}$ be the language consisting of statements in R. A non-interactive zero-knowledge (NIZK) proof-of-knowledge system for a language $\mathcal{L}$ consists of the following algorithms and protocols:

$\Pi.\mathsf{Setup}(\mathit{grp}) \to \mathit{CRS}$. On input $\mathit{grp} \leftarrow (1^\kappa)$, it outputs common parameters (a common reference string) CRS for the proof system.

$\Pi.\mathsf{Prove}(\mathit{CRS}, W, \mathit{Stmt}) \to \pi$. On input a statement Stmt and a witness W, it generates a zero-knowledge proof π that the witness satisfies the statement.

$\Pi.\mathsf{Verify}(\mathit{CRS}, \pi, \mathit{Stmt}) \to 0/1$. On input Stmt and π, it outputs 1 if π is valid, and 0 otherwise.

We explain the notation for the statements Stmt. We call extractable (non-extractable) witnesses that can (cannot) be extracted from the corresponding proof, respectively. To express the "extractability" property of the witnesses we use notation introduced by Camenisch et al. [29]. For the extractable witnesses we use the "knowledge" notation ($\text{\reflectbox{K}}$), and for the non-extractable witnesses we use "existence" ($\exists$) notation. We define $\mathfrak{K}$ as a set of extractable witnesses and $\mathfrak{E}$ as a set of non-extractable witnesses that we can only prove existence about. We only consider proofs for multi-exponentiations (for existence) and pairing products (for existence and knowledge) equations:

$$\mathit{Stmt} = \text{\reflectbox{K}} \left\{ Y_i, \tilde{Y}_i \in \mathfrak{K} \right\}_{i=1}^{n} ; \exists \{x_j \in \mathfrak{E}\}_{j=1}^{m} : \bigwedge_i (z_i = \prod_j G^{x_j}) \\ \wedge\, e(G, \tilde{G}) = \prod_{i,j} \left(e(Y_i, \tilde{B}_i) \cdot e(A_i, \tilde{Y}_i) \right).$$

For simplicity of presentation, we do not explicitly specify public values of a statement as additional input to the algorithms, since they are clear from the description of the statement and the list of witnesses.

We employ different proof systems that are either witness indistinguishable or zero-knowledge in terms of privacy, and either extractable or simulation-extractable in term of soundness. For the security proofs we introduce the following algorithms:

$\Pi.\mathsf{ExtSetup}(grp) \to (CRS, td_{ext})$. On input grp, it outputs a common reference string CRS and a trapdoor td_{ext} for extraction of valid witnesses from valid proofs. This is for witness-indistinguishable extractable proofs.

$\Pi.\mathsf{SimSetup}(grp) \to (CRS^{sim}, td_{ext}, td_{sim})$ It outputs a CRS and the extraction and simulation trapdoors. This is for simulation-extraction.

$\Pi.\mathsf{SimProve}(CRS^{sim}; td_{\mathsf{sim}}; Stmt) \to \pi$. On input CRS^{sim} and a trapdoor td_{sim}, it outputs a simulated proof π such that $\Pi.\mathsf{Verify}(CRS^{sim}; \pi; Stmt) = 1$.

$\Pi.\mathsf{Extract}(CRS; td_{\mathsf{ext}}; \pi; Stmt) \to W$. On input a proof π and a trapdoor td_{ext}, it extracts a witness W that satisfies the statement $Stmt$ of the proof π.

For simulation-extractable NIZK proofs (that are non-malleable) we also allow an additional public input to the Prove and Verify algorithms $\Pi.\mathsf{Prove}(CRS, W, Stmt, L)$, $\Pi.\mathsf{Verify}(CRS, \pi, Stmt, L)$ – a message (label) L, which is non-malleably attached to the proof (i.e. the signature of knowledge is computed on this message).

3.3 Our Redactable Signature Scheme

We construct our redactable signature scheme URS from a structure-preserving signature scheme SPS, a vector commitment scheme VC, and an extractable and witness-indistinguishable non-interactive proof-of-knowledge system Π described in the previous section. Some SPS and vector commitment schemes might also support randomization; we already discussed such a property for vector commitments in the last sub-section; for signatures we refer the reader to [3, 4]. We denote the randomization algorithms of commitments and signatures by VC.Rand and SPS.Rand, respectively. We denote the randomizable elements of a SPS signature Σ by $\psi_{\texttt{rnd}}(\Sigma)$ and the other elements by $\psi_{\texttt{wit}}(\Sigma)$. (For a non-randomizable SPS signature $\psi_{\texttt{wit}}(\Sigma) = \Sigma$.)

Construction. We first give a simplified construction that does not utilize the randomization algorithm of the vector commitment scheme and, therefore, is slightly less efficient.

$\mathsf{URS.SGen}(1^\kappa)$. Compute $grp \leftarrow \mathcal{G}(1^\kappa)$, $pp \leftarrow \mathsf{VC.Setup}(grp)$, $CRS \leftarrow \Pi.\mathsf{Setup}(grp)$, output $SP = (grp, pp, CRS)$.

$\mathsf{URS.Kg}(SP)$. Obtain grp from SP, generate $(pk_{sps}, sk_{sps}) \leftarrow \mathsf{SPS.Kg}(grp)$, output $pk = (pk_{sps}, SP)$ and $sk = (sk_{sps}, pk)$.

$\mathsf{URS.Sign}(sk, \vec{m})$. Pick $r \leftarrow \mathbb{Z}_p$, compute $C = \mathsf{VC.Commit}(pp, \vec{m}, r)$ and $\Sigma \leftarrow \mathsf{SPS.Sign}(sk_{sps}, C)$, and return $\sigma = (\Sigma, C, r)$.

$\mathsf{URS.Derive}(pk, I, \vec{m}, \sigma)$. First, compute $W = \mathsf{VC.Open}(pp, I, \vec{m}, r)$. Then, if a SPS.Rand algorithm is present, randomize the signature as $\Sigma' \leftarrow \mathsf{SPS.Rand}(pk_{sps}, \Sigma)$; otherwise, set $\Sigma' \leftarrow \Sigma$. And compute the proof $\pi \leftarrow \Pi.\mathsf{Prove}(CRS; C, W, \psi_{\texttt{wit}}(\Sigma'); \mathsf{N}\, C, W, \psi_{\texttt{wit}}(\Sigma') : \mathsf{SPS.Verify}(pk_{sps}, \Sigma', C) \wedge \mathsf{VC.Check}(pp, C, \vec{m}_I, W)\ )$. Return $\sigma = (\psi_{\texttt{rnd}}(\Sigma'), \pi)$ as the signature for $\vec{m}_I$.

$\mathsf{URS.Verify}(pk, \sigma, \vec{m}_I)$. Check that $\Pi.\mathsf{Verify}\big(CRS; \pi; \mathsf{N}\, C, W, \psi_{\texttt{wit}}(\Sigma') : \mathsf{SPS.Verify}(pk_{sps}, \Sigma', C)\big) = \mathsf{VC.Check}(pp, C, \vec{m}_I, W) = 1$.

Theorem 4. URS *is an unforgeable redactable signature scheme, if the SPS scheme is unforgeable, the vector commitment scheme satisfies the batch binding and opening non-malleability property, and the proof-of-knowledge system is extractable and witness indistinguishable.* The proofs of Theorems 4 is in §C.1.

Theorem 5. URS *is an unlinkable redactable signature scheme if the proof-of-knowledge system is witness indistinguishable.* The proofs is given in §C.1.

Alternative construction. If the vector commitment scheme has a randomization algorithm, we can use a slightly more efficient scheme as described in §C.2.

Strengthened scheme for an universally composable construction. To be able to satisfy the UC functionality, we require an additional key-extraction property. We thus build an augmented redactable signature scheme URS from a redactable signature scheme URS*(without key extraction) and a zero-knowledge non-interactive proof-of-knowledge system Π^*.

URS.SGen(1^κ). Run $SP^* \leftarrow$ URS*.SGen(1^κ), get grp from SP^*, run $CRS_{sk} \leftarrow \Pi^*.\mathsf{Setup}(grp)$, and output $SP = (SP^*, CRS_{sk})$.

URS.Kg(SP). Obtain SP^* and CRS_{sk} from SP, $(pk^*, sk^*) \leftarrow$ URS*.Kg(SP^*). Compute the proof
$\pi_{sk} \leftarrow \Pi^*.\mathsf{Prove}\left(CRS_{sk}; (sk^*, r); \text{⅄}\, sk^* \;\exists\, r : (pk^*, sk^*) = \mathsf{URS}^*.\mathsf{Kg}(SP^*; r)\right)$. Output $pk = (SP, pk^*, \pi_{sk})$ and $sk = (sk^*, pk)$. We note that URS*.Kg$(SP^*; r)$ denotes a key generation algorithm with fixed randomness r.

URS.CheckPK(SP, pk).
Check that $\Pi^*.\mathsf{Verify}(CRS_{sk}; \pi_{sk}; \text{⅄}\, sk \;\exists\, r \;:\; (pk, sk) = \mathsf{URS}^*.\mathsf{Kg}(SP^*; r)) = 1$.

Sign, Derive, Verify are almost unchanged and use pk^* internally. SGenT and ExtractKey use the extraction setup and extractor of the proof system respectively, while CheckKeys checks that the relation R holds for pk and sk.

Note that Groth-Sahai proofs can be used to implement key-extraction by proving a binary, or n-ary decomposition of the secret key [60]. But this comes at a huge cost of more than 61,000 group elements at 128-bit security, even if this cost is only incurred once by every user per public key. We propose instead to use *fully* structure-preserving signatures (FSPS) [1] such that sk consists of group elements and can be easily extracted. FSPS for signing single group elements can be as cheap as 15 elements per signature and proofs of key possession consist of just 18 elements.

Theorem 6. *The strengthened scheme* URS *is an unforgeable, unlinkable, and key extractable redactable signature scheme, if the underlying redactable signature scheme* URS* *is unforgeable and unlinkable, and the proof-of-knowledge system* Π^**is zero-knowledge and extractable.*

Unforgeability and unlinkability are simple corollaries of Theorem 4 and Theorem 5. Key-extractability follows directly from the extractability of the proof system.

Signing group elements as additional parts of the message. While the presented redactable signature scheme can sign and quote a large number of values in $\mathbb{Z}_p$ very efficiently, in certain applications, like the one presented in the next section, one might also need to sign a small number of additional group elements. In the Derive algorithm these elements will either be part of the derived message, and given in the clear after derivation, or be treated as part of the witness, i.e., hidden from the verifier. The detailed construction and the security proofs are given in §D.

4 From Unlinkable Redactable Signatures to Anonymous Credentials

As we designed our UC-secure URS scheme as a building block for privacy-preserving protocols, anonymous credentials are a natural application. Indeed, an (unlinkable) redactable signature scheme is already a simple selective-disclosure credential system where the attributes issued to users are the messages signed in Tier 1 signatures and a user can later reveal a subset of her attributes by deriving a Tier 2 signature. However, in an anonymous credential system, users also require secret keys and pseudonyms (pseudonymous public keys), on which credentials can be issued and with respect to which credentials can be presented. This allows users to prove that they possess several credentials issued from different parties on the same secret key [20, 32].

In this section, we extend the functionality of URS in two ways: (1) we bind Tier 1 signatures to user secret keys in a way that prevents the derivation of signatures without knowledge of the secret and (2) we bind Tier 2 derived signatures to the unique context, cxt (nonce), to prevent replay attacks in which an attacker shows the same signature derived twice.

We first recall the algorithms of a multi-issuer anonymous credential system and then provide an instantiation using URS. To be modular and to simplify the analysis, we then provide an ideal functionality for a single issuer. The functionality is carefully designed to self-compose naturally into a full-fledged credential system with multiple issuers. Finally, we provide a concrete instantiation of our generic construction and analyze its efficiency.

4.1 Algorithms of Our Anonymous Credential System

Let us first introduce the parties and the algorithms of a multi-issuer anonymous credential system supporting user attributes (cf. [20, 32]). Its protagonists are *users* ($\mathcal{U}$), *issuers* ($\mathcal{I}$), and *verifiers* ($\mathcal{V}$). Each user has a secret key X, from which she can derive (cryptographic) pseudonyms P. To get a credential issued, a user sends to the issuer a pseudonym P together with a (non-interactive) proof $\pi_{X,P}$ that she is privy to the underlying secret key. The issuer will then issue her a credential $Cred$ on P containing the attributes $\vec{a}$ the issuer vouches for. The user can then present the credential to a verifier under a potentially different pseudonym P' by sending, together with P', a (non-interactive) proof $\pi_{X,Cred}$ that she possesses a credential on the attributes $\vec{a}_I$. Recall that I defines which attributes shall be revealed.

A credential system Cred defines a set of algorithms: a system parameters generation algorithm SGen; an issuer setup algorithm Kg; a user secret generation algorithm SecGen; algorithms for pseudonym generation and verification NymGen and NymVerify, respectively; an algorithm to request a credential RequestCred; an algorithm for issuing a credential IssueCred; an algorithm to check a newly issued credential for correctness CheckCred; an algorithm to show a credential with respect to a pseudonym (to create a credential proof) Prove; and an algorithm to verify a credential proof Verify.

Definition 10. *A credential system* Cred *is a set of the following basic algorithms:*

Cred.SGen$(1^\kappa) \to SP$ *is a system setup algorithm that generates the common system-wide parameters SP.*

Cred.Kg$(SP) \to (pk, sk)$ *is an issuer setup algorithm, with which the issuer generates a key pair for issuing credentials and publishes the public key pk.*

Cred.SecGen$(SP) \to X$ *is a secret generating algorithm that takes the common parameters as an input and outputs a fresh user secret X.*

Cred.NymGen$(SP, X) \to (P, aux(P))$ *is a pseudonym generating algorithm that takes system parameters, a user secret as an input and creates a pseudonym P and some auxiliary information $aux(P)$ for P.*

Cred.NymVerify$(SP, P, X, aux(P)) \to 1/0$ *verifies that the pseudonym was generated correctly w.r.t. to the user's secret key.*

Cred.RequestCred$(SP, pk, X, P, aux(P)) \to (\pi_{X,P}, aux(Cred))$ *takes the common parameters SP, the public key of the issuer pk, the secret X, and a pseudonym P together with the corresponding auxiliary information $aux(P)$, and produces a request for issuing a credential (a proof of knowledge of the user secret) $\pi_{X,P}$ and some optional auxiliary information*

$aux(Cred)$ for the requested credential.

$\mathsf{Cred.IssueCred}(SP, sk, P, \vec{a}, \pi_{X,P}) \rightarrow Cred/\bot$ *takes the common parameters SP, a request for issuing a credential received from the user $\pi_{X,P}$, the issuer key pair, and the vector of attributes $\vec{a}$ assigned to the user as an input, and, if the issuance token verifies, produces a credential, otherwise it outputs $\bot$.*

$\mathsf{Cred.CheckCred}(SP, pk, X, P, aux(P), Cred, aux(Cred), \vec{a}) \rightarrow 0/1$ *verifies if a credential received from the issuer is valid.*

$\mathsf{Cred.Prove}(SP, pk, X, P, aux(P), Cred, aux(Cred), \vec{a}, I, cxt) \rightarrow \pi_{X,Cred}$ *is an algorithm executed by the user, that takes the user secret X, a credential $Cred$, a pseudonym P, and a context cxt as an input and outputs a (presentation) proof $\pi_{X,Cred}$ that reveals only a required subset I of the attributes $\vec{a}$ from the credential.*

$\mathsf{Cred.Verify}(SP, pk, P, \vec{a}_I, \pi_{X,Cred}, cxt) \rightarrow 1/0$ *is an algorithm executed by the verifier, that verifies a presentation proof provided by the user.*

We instantiate these algorithms by adding support for user secrets, pseudonyms and contexts to our redactable signature scheme. Besides the URS algorithms, we use pseudonym generation and verification algorithms based on a structure preserving commitment scheme SPC and a gap problem to generate credential specific secrets. A gap problem has a generator $\mathsf{K}_{\mathsf{Gap}}$ that generates (X_{Cred}, Y_{Cred}) and a verification algorithm $\mathsf{V}_{\mathsf{Gap}}$, such that it is easy to verify (X_{Cred}, Y_{Cred}) but hard to compute X_{Cred} from Y_{Cred}. This hardness is used to prevent the mauling of the simulation-extractable presentation proofs.

Table 1 gives the construction of our credential scheme. We group the core credential algorithms into those used for setup, issuing and presentation. In our security definition and the proof we will make use of additional algorithms for simulation and extraction.

4.2 Ideal Functionality for Credentials

Rather than defining a credential system with a number of different issuers, we chose to first give a protocol for a single issuer only, to reduce complexity. Later, we show how to build a full-fledged credential system with multiple issuers by combining multiple instances of the protocol, one for each issuer. To this end, we support sharing pseudonyms across instances. Moreover, users should be able to use different pseudonyms for the issuance of a credential and for each proof of a credential. Thus, the functionality is parameterized by a pseudonym generation and verification scheme ($\mathsf{SecGen}, \mathsf{NymGen}, \mathsf{NymVerify}$) which in our realization will be a commitment scheme. For compositionality we do not model unlinkability, but that presentation proofs leak nothing except their pseudonyms and the revealed attributes (modeled using simulation). We therefore, somewhat counter intuitively, do not immediately rely on pseudonyms being hiding. The hiding property, however, matters for the whole credential system. Indeed, without pseudonyms being hiding, users can be linked when they interact with the issuer or prove possession of a credential w.r.t. a pseudonym, both in the real and the ideal world.

We start with specifying an ideal functionality $\mathcal{F}_{\mathsf{Cred}}$ that describes the security properties with respect to a single issuer and for pseudonyms that are controlled by the environment. We then describe a protocol $\mathcal{R}_{\mathsf{Cred}}$ for a single issuer based on $\mathcal{F}_{\mathsf{CA}}$ and $\mathcal{F}_{\mathsf{CRS}}$. Finally, we show that $\mathcal{R}_{\mathsf{Cred}}$ indeed securely realizes $\mathcal{F}_{\mathsf{Cred}}$ in the $(\mathcal{F}_{\mathsf{CA}}, \mathcal{F}_{\mathsf{CRS}})$-hybrid model under static corruptions.[2] As the efficient integration of pseudonyms requires zero-knowledge and thus whitebox techniques, $\mathcal{R}_{\mathsf{Cred}}$ does not use $\mathcal{R}_{\mathsf{URS}}$ as a (blackbox) subroutine. We will, however, carefully align the internals of $\mathcal{F}_{\mathsf{Cred}}$ and $\mathcal{R}_{\mathsf{Cred}}$ with those of $\mathcal{F}_{\mathsf{URS}}$ and $\mathcal{R}_{\mathsf{URS}}$ respectively, such that we can use the ~~UC emulation theorem in one of the hy~~brid steps of our security proof.

[2]As users are anonymous and the network is insecure we require the presence of at least one dishonest user for modeling purposes.

Setup algorithms

$\mathsf{Cred.SGen}(1^\kappa)$: Compute $SP_{\mathsf{URS}} \leftarrow \mathsf{URS.SGen}(1^\kappa)$; $CRS_X \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$; $pp_{\mathsf{SPC}} \leftarrow \mathsf{SPC.Setup}(SP_{\mathsf{URS}})$; and output $SP = (SP_{\mathsf{URS}}, CRS_X, pp_{\mathsf{SPC}})$.

$\mathsf{Cred.Kg}(SP)$: Compute $(pk_{\mathsf{URS}}, sk_{\mathsf{URS}}) \leftarrow \mathsf{URS.KeyGen}(SP)$, and output $(sk, pk) = (sk_{\mathsf{URS}}, pk_{\mathsf{URS}})$.

$\mathsf{Cred.SecGen}(SP)$: Take G from SP, pick random $x \leftarrow \mathbb{Z}_p, X = G^x$. Output X.

$\mathsf{Cred.NymGen}(SP, X)$: $(P, O) \leftarrow \mathsf{SPC}(pp_{\mathsf{SPC}}, X)$. Output $(P, aux(P) = O)$.

$\mathsf{Cred.NymVerify}(SP, X, P, aux(P))$: Parse $aux(P)$ as O. Output the result of $\mathsf{SPC.Check}(pp_{\mathsf{SPC}}, P, O)$.

Issuing algorithms

$\mathsf{Cred.RequestCred}(SP, pk, X, P, aux(P))$:

$(X_{Cred}, Y_{Cred}) \leftarrow \mathsf{K}_{\mathsf{Gap}}$; $\pi_{X,P} \leftarrow \Pi.\mathsf{Prove}(CRS_X; (X, X_{Cred}, aux(P)); Stmt_P)$, where $Stmt_P = \big(⋊\, X, X_{Cred}, aux(P) : \mathsf{NymVerify}(SP, X, P, aux(P)) = 1 \wedge \mathsf{V}_{\mathsf{Gap}}(X_{Cred}, Y_{Cred}) = 1\big)$.

Add $X_{Cred}, Y_{Cred}, P, aux(P)$ to $aux(Cred)$ and Y_{Cred} to $\pi_{X,P}$.

$\mathsf{Cred.IssueCred}(SP, sk, P, \vec{a}, \pi_{X,P})$:

1. Verify the request for issuance $\pi_{X,P}$:
   If $\Pi.\mathsf{Verify}(CRS_X; \pi_{X,P}; Stmt_P) = 0$, return $\perp$.
2. Else, generate a credential by creating a Tier 1 signature on the vector of messages, providing the pseudonym and a gap problem challenge, and calling $\sigma \leftarrow \mathsf{URS.Sign}(sk, (P, Y_{Cred}, \vec{a}))$ and output $Cred = \sigma$.

$\mathsf{Cred.CheckCred}(SP, pk, X, P, aux(P), Cred, aux(Cred), \vec{a})$: Output the result of $\mathsf{URS.Verify}(pk, Cred, (P, Y_{Cred}, \vec{a}))$.

Presentation algorithms

$\mathsf{Cred.Prove}(SP, pk, X, P', aux(P)', Cred, aux(Cred), \vec{a}, I, cxt) \rightarrow \pi_{X,Cred}$:

1. Obtain $X_{Cred}, Y_{Cred}, P, aux(P)$ from $aux(Cred)$.
2. Run $\sigma_I \leftarrow \mathsf{URS.Derive}(pk, I, (P, Y_{Cred}, \vec{a}), \sigma))$.
3. Compute a proof of knowledge of the secret, pseudonym, and the correctness of the signature on a context: $\pi_{X,Cred} = \Pi.\mathsf{Prove}(CRS_X; (X, P, aux(P), aux(P)', Y_{Cred}, X_{Cred}); Stmt, cxt)$; $Stmt = \big(⋊\, X, P, aux(P), aux(P)', Y_{Cred}, X_{Cred} : \mathsf{NymVerify}(SP, X, P', aux(P)') = 1 \wedge \mathsf{NymVerify}(SP, X, P, aux(P)) = 1 \wedge \mathsf{URS.Verify}(pk, \sigma_I, (P, Y_{Cred}, \vec{a})_I)) = 1 \wedge \mathsf{V}_{\mathsf{Gap}}(X_{Cred}, Y_{Cred}) = 1\big)$.
   Add σ_I to $\pi_{X,Cred}$ as a part of the public input.

$\mathsf{Cred.Verify}(SP, pk, P', \pi_{X,Cred}, \vec{a}_I, cxt)$: Output the result of $\Pi.\mathsf{Verify}(CRS_X; \pi_{X,\mathsf{Cred}}; Stmt(SP, P', \sigma_I, \vec{a}_I), cxt)$.

Simulation and extraction algorithms

$\mathsf{Cred.SGenT}(1^\kappa)$: $(SP_{\mathsf{URS}}, td) \leftarrow \mathsf{URS.SGenT}(1^\kappa)$;
$(CRS_X, td_{ext}, td_{sim}) \leftarrow \Pi.\mathsf{SimSetup}(1^\kappa)$; $pp_{\mathsf{SPC}} \leftarrow \mathsf{SPC.Setup}(SP_{\mathsf{URS}})$.
Output $\big(SP = (SP_{\mathsf{URS}}, CRS_X, pp_{\mathsf{SPC}}), td_{ext} = (td, td_{ext}), td_{sim}\big)$.

$\mathsf{Cred.Extract}(SP, td_{ext}, pk, P', \pi_{X,Cred}, \vec{a}_I, cxt)$:
Take $(X, aux(P))$ from $\Pi.\mathsf{Extract}(SP; td_{ext}; \pi_{X,Cred};\ Stmt))$.

$\mathsf{Cred.SimProve}(SP, sk, td_{sim}, P', \vec{a}_I, cxt) \rightarrow \pi_{X,Cred}$:

1. $X \leftarrow \mathsf{SecGen}(SP)$; $(P, aux(P)) \leftarrow \mathsf{NymGen}(SP, X)$.
2. Let $\vec{a}_0$ be a Tier 1 message restored from $\vec{a}_I$ by replacing $\perp$-s with 0-s as if it was derived from the original message $\vec{a}$ by applying $\mathsf{Zero}(\vec{a}, I)$.
3. $\sigma \leftarrow \mathsf{URS.Sign}(sk, (P, Y_{Cred}, \vec{a}_0))$
4. $\sigma_I \leftarrow \mathsf{URS.Derive}(pk, I, (P, Y_{Cred}, \vec{a}_0), \sigma))$.
5. Compute a proof of knowledge of the secret, pseudonym, and the correctness of the signature on a context:
   $\pi_{X,Cred} \leftarrow \Pi.\mathsf{SimProve}(CRS_X; td_{sim}; P'; Stmt, cxt)$.

Table 1: Algorithms of our credential system

Single issuer ideal functionality. The starting point for our credential functionality is the ideal functionality of unlinkable redactable signatures, extended in a number of ways.

Similar to $\mathcal{F}_{\mathsf{URS}}$ (and in line with other UC-functionalities such as $\mathcal{F}_{\mathsf{sig}}$ that need to output cryptographic values), $\mathcal{F}_{\mathsf{Cred}}$ is handed a number of cryptographic algorithms by the simulator, so that $\mathcal{F}_{\mathsf{Cred}}$ can produce artifacts for proofs of credential ownership and attribute disclosure, and to verify such proofs. Indeed, the functionality cannot ask the simulator for such artifact's as sometimes done, but needs to run these algorithm itself to guarantee privacy (cf. $\mathcal{F}_{\mathsf{URS}}$ and the UC-functionalities for blind signatures [6, 46]).

For this to work, the functionality needs to assure verifiers that all pseudonyms are related to some legitimate user. Consequently, our simulator uses a special parameter generation algorithm that also outputs extraction and simulation trapdoors: $\mathsf{Cred.SGenT}(1^\kappa) \rightarrow (SP, td_{ext}, td_{sim})$. The extraction trapdoor is an input to the user secret extraction algorithm $\mathsf{Cred.Extract}(SP, td_{ext}, pk, P, \pi_{X,Cred}, \vec{a}_I, cxt) \rightarrow X, aux(P)$. This algorithm is used to ensure that a proof of possession is accepted only if correct X and $aux(P)$ (such that $\mathsf{NymVerify}$ accepts) are extracted. Similarly, the functionality needs to give assurance to users that nothing except their pseudonyms and their attributes is leaked in a proof of possession. We model this using a simulation algorithm $\mathsf{Cred.SimProve}(SP, sk, td_{sim}, P', \vec{a}_I, cxt) \rightarrow \pi_{X,Cred}$ that the simulator provides to the functionality. This corresponds to the simulator providing the Sign algorithm in $\mathcal{F}_{\mathsf{URS}}$.

We now describe the ideal functionality and highlight the security properties it ensures. $\mathcal{F}_{\mathsf{Cred}}$ maintains some bookkeeping information (tables): $\mathcal{M}_{ISS}$ - for storing information about issued credentials, and $\mathcal{M}_{PRES}$ - for storing information about credentials that produced presentation proofs.

First, upon receiving the $(\mathtt{keygen}, sid)$ message, $\mathcal{F}_{\mathsf{Cred}}$ performs a setup by asking the simulator for the system parameters, keys, trapdoors, algorithms and a list of corrupted parties. It generates all cryptographic artifacts using the algorithms provided. Message $(\mathtt{leakSK}, sid)$ is handled in exactly the same way as for redactable signatures. Message $(\mathtt{checkPK}, \ldots)$ used in $\mathcal{F}_{\mathsf{URS}}$ is no longer needed, as we assume that in the real world all issuer keys are automatically checked upon registration with $\mathcal{F}_{\mathsf{CA}}$.

Second, upon receiving the $(\mathtt{issueCred}, sid, qid, \mathcal{U}, X, P, aux(P))$ message from a user, $\mathcal{F}_{\mathsf{Cred}}$ initiates credential issuing. It will, however, only proceed if the issuer specified in $sid = (\mathcal{I}, sid')$ is willing to issue some attributes $\vec{a}$ to the user, and if X, P, and $aux(P)$ form a valid pseudonym.

Third, upon receiving a proof request in the form of a $(\mathtt{proveCred}, \ldots)$ message, instead of creating a proof using the user secret and other credential information used during the issuance request, the functionality uses the $\mathsf{Cred.SimProve}$ algorithm that only takes the pseudonym, with respect to which the presentation is done, and disclosed attributes as an input, but no private or linkable information, i.e. non-disclosed attributes are already "redacted" . This guarantees that after seeing a presentation token, the issuer cannot tell to which credential the token is related (absent weaknesses in the pseudonym algorithms as discussed above).

Finally, upon receiving the $(\mathtt{verifyCredProof}, \ldots)$ message, $\mathcal{F}_{\mathsf{Cred}}$—besides verifying the proof itself—uses the $\mathsf{Cred.Extract}$ algorithm and bookkeeping information to make checks that guarantee that: (1) the adversary should not be able to produce a presentation proof that verifies and that reveals attributes that were never issued to the user secret of the pseudonym; (2) for honest users that keep their user secret private the adversary should not be able to repeat a proof with a fresh context that was not seen before. Here, we consider an honest user's secret private as long as no dishonest user received a credential for the same secret.

We describe the ideal functionality $\mathcal{F}_{\mathsf{Cred}}$ for credentials in Figure 1. Note that because we consider static corruption, we assume that $\mathcal{F}_{\mathsf{Cred}}$ and $\mathcal{SIM}$ are aware of corrupted parties. Namely, whether $\mathcal{I}$ is corrupted or not and which users $\mathcal{U}$ are honest and which are corrupted.

Functionality $\mathcal{F}_{\mathsf{Cred}}(\mathsf{SecGen}, \mathsf{NymGen}, \mathsf{NymVerify})$

The functionality maintains tables $\mathcal{M}_{ISS}$ and $\mathcal{M}_{PRES}$ initialized to $\emptyset$ and flags kg and $keyleak$ which are initially unset.

- On input $(\mathtt{keygen}, sid)$ from $\mathcal{I}$, verify that $sid = (\mathcal{I}, sid')$ for some sid' and that flag kg is unset. If not, then return $\perp$. Else, do the following:
  1. Send $(\mathtt{initF}, sid)$ to $\mathcal{SIM}$ and wait for a message $(\mathtt{initF}, sid, SP, sk, pk, td_{sim}, td_{ext}, \mathsf{SimProve}, \mathsf{Verify}, \mathsf{Extract})$ from $\mathcal{SIM}$, where SP are the system parameters, td_{sim} and td_{ext} are the simulation and extraction trapdoors respectively, and the rest are polynomial-time algorithms. Store all of these values and set flag kg.
  2. Return $(\mathtt{verificationKey}, sid, pk)$ to $\mathcal{I}$.
- On input $(\mathtt{leakSK}, sid)$ from $\mathcal{I}$ verify that $sid = (\mathcal{I}, sid')$ for some sid'. If not, return $\perp$. Else, if flag kg is set, set flag $keyleak$ and return $(\mathtt{leakSK}, sid, sk)$.
- On input $(\mathtt{issueCred}, sid, qid, X, P, aux(P))$ from $\mathcal{U}$, check $sid = (\mathcal{I}, sid')$ for some sid', and that flag kg is set. If not, return $\perp$. Else send a public delayed output $(\mathtt{issueCred}, sid, qid, P)$ to $\mathcal{I}$.
- On input $(\mathtt{issueCred}, sid, qid, \vec{a})$ from $\mathcal{I}$, check for $(\mathtt{issueCred}, sid, qid, X, P, aux(P))$ from $\mathcal{U}$, and verify that $sid = (\mathcal{I}, sid')$ for some sid' and that the flag kg is set. If not, return $\perp$. Else, do the following:
  1. Run $b \leftarrow \mathsf{NymVerify}(SP, P, X, aux(P))$. If $b = 0$, return $\perp$.
  2. Add $(ISS, \perp, X, \vec{a})$ to $\mathcal{M}_{ISS}$.
  3. Send a public delayed output $(\mathtt{credIssued}, sid, qid, \vec{a})$ to $\mathcal{U}$.
  4. When $(\mathtt{credIssued}, sid, qid, \vec{a})$ is delivered to $\mathcal{U}$, update the issuance record by adding the user to $(ISS, \mathcal{U}, X, \vec{a})$ of $\mathcal{M}_{ISS}$.
- On input $(\mathtt{proveCred}, sid, X, P', aux(P)', I, \vec{a}, cxt)$ from $\mathcal{U}$, do the following:
  1. Check if kg is set. If not, return $\perp$.
  2. Check if $\mathsf{NymVerify}(SP, P', X, aux(P)') = 1$. If not, return $\perp$.
  3. Check if $(ISS, \mathcal{U}, X, \vec{a})$ exists. If not, return $\perp$.
  4. $\pi_{X,Cred} \leftarrow \mathsf{Cred.SimProve}(SP, sk, td_{sim}, P', \vec{a}_I, cxt)$.
  5. Check if $\mathsf{Cred.Verify}(SP, pk, P', \pi_{X,Cred}, \vec{a}_I, cxt) = 0$, then output $\perp$.
  6. Add $(PRES, \mathcal{U}, cxt, X, P', aux(P)', \vec{a}_I, \pi_{X,Cred})$ to $\mathcal{M}_{PRES}$.
  7. Return $(\mathtt{credProved}, sid, \vec{a}_I, \pi_{X,Cred})$ to $\mathcal{U}$.
- On input $(\mathtt{verifyCredProof}, sid, pk', P', \pi'_{X,Cred}, \vec{a}'_I, cxt')$ from some party $\mathcal{P}$, do the following:
  1. Verify the proof $result = \mathsf{Cred.Verify}(SP, pk', P', \pi'_{X,Cred}, \vec{a}'_I, cxt')$.
  2. If $pk \neq pk'$, or $keyleak$ is set, or $\mathcal{I}$ is dishonest, or $result = 0$, send $(\mathtt{verified}, sid, \vec{a}'_I, result)$ to $\mathcal{P}$.
  3. Else, if there is a record $(PRES, *, cxt', *, P', *, \vec{a}'_I, \pi'_{X,Cred})$ return $(\mathtt{verified}, sid, \vec{a}'_I, 1)$ to $\mathcal{P}$.
  4. Otherwise, extract $X', aux(P)' \leftarrow \mathsf{Cred.Extract}(SP, td_{ext}, pk, P', \pi'_{X,Cred}, \vec{a}'_I, cxt')$.
  5. If $\mathsf{NymVerify}(SP, P', X', aux(P)') = 0$, return $(\mathtt{verified}, sid, \vec{a}'_I, 0)$ to $\mathcal{P}$.
  6. Else, if there is a record $(ISS, \mathcal{U}, X', \vec{a})$ in $\mathcal{M}_{ISS}$ for a corrupted user $\mathcal{U}$ such that $\vec{a}_I = \vec{a}'_I$, return $(\mathtt{verified}, sid, \vec{a}'_I, 1)$ to $\mathcal{P}$.
  7. Else return $(\mathtt{verified}, sid, \vec{a}'_I, 0)$ to $\mathcal{P}$.

Figure 1: The ideal functionality for single issuer anonymous credentials

We consider a protocol $\mathcal{R}_{\mathsf{URS}}$ that realizes $\mathcal{F}_{\mathsf{URS}}$ using the algorithms in §4.1 in the $(\mathcal{F}_{\mathsf{CRS}}, \mathcal{F}_{\mathsf{CA}})$-hybrid model where SP is the reference string and each call to $\mathcal{F}_{\mathsf{Cred}}$ is essentially replaced by running one of the algorithms of Cred. The detailed description of $\mathcal{R}_{\mathsf{Cred}}$ is given below in §E.1.

Theorem 7. *Let* URS *be the unlinkable redactable signature scheme according to Definition 1,* SPC *is a structure-preserving commitment scheme,* Gap *be a gap problem,* Π *be a non-interactive proof of knowledge system. Then* $\mathcal{R}_{\mathsf{Cred}}$ *securely realizes* $\mathcal{F}_{\mathsf{Cred}}$ *in the* $(\mathcal{F}_{\mathsf{CRS}}, \mathcal{F}_{\mathsf{CA}})$*-hybrid model if* URS *is correct, unlinkable, unforgeable, and key extractable,* SPC *is binding, the non-interactive proof-of-knowledge system is zero-knowledge and simulation extractable, and the* Gap *problem is hard.* The proof is provided in §E.2.

Building a full-fledged credential system with multiple issuers. We now explain how to use our credential functionality to support multiple issuers using multiple sessions of $\mathcal{F}_{\mathsf{Cred}}$, one for each issuer, and a pseudonym functionality that supports creation of user secrets and generation and verification of pseudonyms. The pseudonyms are required to be both hiding and binding to hide a user secret and prevent users from sharing credentials without also sharing their secret. Users now can generate a user secret and different pseudonyms on it and then use multiple calls to $\mathcal{F}_{\mathsf{Cred}}$ instances for different issuers to get credentials (from different issuers but bound to the same secret). To compose a presentation proof that reveals attributes from different credentials, the user creates a pseudonym P' and uses the corresponding $\mathcal{F}_{\mathsf{Cred}}$ instances to generate the required proofs with respect to this pseudonym. Since the pseudonym is the same in different proofs and each proof guarantees the same underlying secret in the credential and the pseudonym, pulling these proofs together results in a single proof for multiple credentials. Each proof block guarantees unlinkability and unforgeability, and since the pseudonym is both binding and hiding this composed proof is also unforgeable and unlinkable with respect to the other composed proofs. The verification is done by querying the corresponding $\mathcal{F}_{\mathsf{Cred}}$ instances for verification of each particular proof part.

4.3 Instantiation and Efficiency Analysis

To analyze the efficiency of our scheme we consider a concrete instantiation scenario. We instantiate our non-interactive construction with Groth-Sahai proofs [50], the structure-preserving commitment scheme of [5], and our unlinkable redactable signature scheme presented in §3.3. As a gap problem we pick the Computational Diffie-Hellman problem. The URS scheme is instantiate with the fully structure-preserving signature scheme by Abe et al. [1], Groth-Sahai proofs, and the vector commitment scheme from §3.1. The proof of Theorem 8 follows from Theorems 6-7.

Theorem 8. *The credential system described above securely realizes* $\mathcal{F}_{\mathsf{Cred}}$ *defined in §4.2 if the SXDH, J-RootDH,* n*-BSDH,* q*-SDH, XDLIN, co-CDH, and DBP assumptions hold.* Consult building blocks for definitions of assumptions.

We refer to §5 for the comparison with prior work. We stress that the complexity of the Prove and Verify algorithms is independent of the number of all attributes contained in a credential.

The size of the credential proof is roughly 130 group elements (100 when using the SPS of [3] instead of FSPS). This means that the communication efficiency for showing a credential with respect to a pseudonym is around 8 KB (6 KB for SPS) at 128-bit security level, which is close to Idemix credentials [32] as the size of pairing groups is much smaller than the size of RSA groups and because the size of Idemix credential proofs is linear in the number of attributes. Besides, Idemix credentials do not provide such strong formal security guarantees, i.e. they require random oracles for non-interactive proofs and are not universally composable. Our non-UC scheme is more efficient than the credential system of Izabachène et al. [54] that has credential proofs of around 8 KB, while our UC scheme has comparable proof sizes. Our scheme is much less efficient than the scheme of [62] but their scheme relies on hash functions in their construction and thus does not enable efficient protocol design.

Open questions. We leave the construction of a scheme that achieves the same functionality as ours with the efficiency of [62]—perhaps using fully structure preserving signatures of equivalence classes—as an interesting open problem. Other interesting questions are exploiting the lack of opening non-malleability for attacks on existing constructions and efficiently basing the opening non-malleability property of vector commitments on a more standard cryptographic assumption than the J-RootDH assumption of Definition 9.

5 Detailed Comparison with Related Work

In this section we compare our URS and Cred schemes with related work and highlight the advantages of our scheme. A variety of signature schemes with flexible signing capabilities and strong privacy properties have been proposed [41, 18, 10, 14]. Such schemes generalize existing privacy-preserving schemes such as group signatures, and anonymous credentials. Our work on URS is most closely related to the work of Chase et al. [41]. They define simulatable malleable signatures and show how to instantiate (a base scheme without attributes for) delegatable credentials. Ahn et al. [7] and Attrapadung et al. [8] consider redactable signatures but for a different redact operation that allows to quote substrings. While these works provide a fresh definitional approach (indeed our definitions of unlinkability and unforgeability are inspired by [41]), they do not aim at being directly applicable in the real world and are rather inefficient, especially when redacting a message vector with a large number of attributes. Canard and Lescuyer [35] proposed an anonymous credential system built from sanitizable signatures—a special case of malleable signatures. Apart from being less efficient than ours, their scheme is only proven secure in the random oracle model.

Our starting point for achieving practical efficiency is vector commitments [40, 59, 38], a cryptographic building block introduced by Catalano, Fiore and Messina [40]. Kate et al. [55] mention one-time-show credentials as one of the applications of their polynomial commitments scheme. Without providing any construction and security proofs, the authors suggest to build credentials by committing to the attributes, which the issuer then certifies by signing the resulting commitment. Recently, Kohlweiss and Rial [58] built a private access control protocol with efficient selective attribute disclosure from vector commitments. Their solution, however, is still one-time-show and requires re-issuing and thus interaction. Instead, we construct a multi-show credential system and put their intuition on a formal basis.

The first efficient multi-show anonymous credential scheme is [30]. Its success led to implementations [34] and related protocols are even contained in standards [21]. Many cryptographic papers on anonymous credentials often focus on a base protocol, with pseudonyms, but without attributes. A notable exception is [26], which studies an extension to the identity mixer system for efficient attribute disclosure. Their result is in the random oracle model, however. The first non-interactive anonymous credential scheme that does not rely on random oracles is [12]. P-signatures, the primitive underlying their construction is a signature scheme falling short of being structure-preserving and is engineered towards the anonymous credentials application. Both P-signatures, and also the work on delegatable credentials [47] consider a base protocol without attributes. Belenkiy et al. [13] define multi-block P-signatures with a proof size linear in the number of messages.

In Table 2 we compare the communication efficiency of our schemes with P-signatures [12, 13]. Our scheme is more efficient and provides UC-security only at the cost of the public key being larger by a constant factor.

	UC	PK size	Tier 1 Signature	Tier 2 signature/Proof
URS* (SPS,[3])	–	$O(n)$ $\|\|\mathbb{G}\|\|$	8 $\|\|\mathbb{G}\|\|$, $(n+1)$ $\|\|\mathbb{Z}_p\|\|$	19 $\|\|\mathbb{G}\|\|$, k $\|\|\mathbb{Z}_p\|\|$
URS (SPS,[3])	+	$O(n + \|\|sk\|\|)$ $\|\|\mathbb{G}\|\|$	8 $\|\|\mathbb{G}\|\|$, $(n+1)$ $\|\|\mathbb{Z}_p\|\|$	19 $\|\|\mathbb{G}\|\|$, k $\|\|\mathbb{Z}_p\|\|$
URS (FSPS,[1])	+	$O(n)$ $\|\|\mathbb{G}\|\|$	16 $\|\|\mathbb{G}\|\|$, $(n+1)$ $\|\|\mathbb{Z}_p\|\|$	38 $\|\|\mathbb{G}\|\|$, k $\|\|\mathbb{Z}_p\|\|$
P-sig [12]	–	4$\|\|\mathbb{G}\|\|$	3$\|\|\mathbb{G}\|\|, n = 0$	34 $\|\|\mathbb{G}\|\|, k = 0$
Block P-sig [13]	–	$O(n)$ $\|\|\mathbb{G}\|\|$	3$\|\|\mathbb{G}\|\|$, n $\|\|\mathbb{Z}_p\|\|$	$(16n + 34)$ $\|\|\mathbb{G}\|\|$ + k $\|\|\mathbb{Z}_p\|\|$

Table 2: Efficiency comparison of URS with P-signatures. (n, k are the size of the original and quoted message, $||sk||$, $||\mathbb{G}||$ and $||\mathbb{Z}_p||$ are the bit-lengths of the signing key, group elements and exponents, respectively. URS* is our basic scheme without key extraction. The proof of key possession for FSPS-based URS consists of 18 elements.

Extending the work of [12] in combination with vector commitments, Izabachène et al. [54] construct block-wise P-signatures and use them to built efficient anonymous credentials (see §4.3 for an efficiency comparison with our credential scheme). Although their techniques are superficially similar to ours, we list some crucial differences:

Their credential scheme targets a restricted class of predicates, individual attribute release and dot-products, however, it is less efficient than the instantiation of our scheme from §4.3, and requires more communication rounds. Instead, we advocate the use of general-purpose NIZK to prove arbitrary predicates as natural extensions to our core scheme. Moreover, they do not provide replay-attack countermeasures for presentation proofs. A reuse of another user's proof is not considered a forgery. Indeed, their definitions do not guarantee that credentials are non-malleable, implying that multiple credential proofs of true predicates may be combinable into a proof of an yet unproven, but true, predicate. This is crucial for a non-interactive credential scheme and is taken care of in this work.

Independently, Hanser and Slamanig [62] presented a credential system with efficient (independent of the number of attributes) attribute disclosure. Their scheme had security weaknesses that were subsequently fixed. However, it still does not capture opening non-malleability, and is only proven in the generic group model.

Contrary to a lot of existing work, we aim for modularity both in our constructions and proofs by formalizing the security properties of all building blocks, including vector commitments, from which we build our URS and our anonymous credential scheme. Thus, all the building blocks and UC-functionalities we define can be used for the modular design of other protocols (e-cash, group signatures, e-voting, etc.).

References

[1] M. Abe, M. Kohlweiss, M. Ohkubo, and M. Tibouchi. Fully structure-preserving signatures and shrinking commitments. In *EUROCRYPT 2015, Part II*, pages 35–65, 2015.

[2] M.Abe, M.Chase, B.David, M.Kohlweiss, R.Nishimaki, and M.Ohkubo. Constant-size structure-preserving signatures: Generic constructions and simple assumptions. ASIACRYPT 2012.

[3] M.Abe, G.Fuchsbauer, J.Groth, K.Haralambiev, and M.Ohkubo. Structure-preserving signatures and commitments to group elements. *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, 2010.

[4] M. Abe, J.Groth, K.Haralambiev, and M.Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. *CRYPTO 2011*, *LNCS*, pages 649–666.

[5] M.Abe, K.Haralambiev, and M.Ohkubo. Group to group commitments do not shrink. *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 301–317. Springer, 2012.

[6] M.Abe and M.Ohkubo. A framework for universally composable non-committing blind signatures. *ASIACRYPT 2009,LNCS*, pages 435–450. Springer, 2009.

[7] J. H.Ahn, D.Boneh, J.Camenisch, S.Hohenberger, A.Shelat, and B.Waters. Computing on authenticated data. *TCC 2012*, *LNCS*, pages 1–20.

[8] N.Attrapadung, B.Libert, and T.Peters. Computing on Authenticated Data: New Privacy Definitions and Constructions. *Asiacrypt 2012*, volume 7658 of *Lecture Notes on Computer Science*. Springer, 2012.

[9] M. H.Au, W.Susilo, and Yi Mu. Constant-size dynamic k-TAA. *SCN 06*, volume 4116 of *LNCS*, pages 111–125. Springer, 2006.

[10] M.Backes, S.Meiser, and D.Schröder. Delegatable functional signatures. Cryptology ePrint Archive, Report 2013/408, 2013. `http://eprint.iacr.org/`.

[11] E.Bangerter, S.Krenn, A.-R.Sadeghi, T.Schneider, and J.-K.Tsay. On the design and implementation of efficient zero-knowledge proofs of knowledge.SPEED-CC'09.

[12] M.Belenkiy, M.Chase, M.Kohlweiss, and A.Lysyanskaya. P-signatures and noninteractive anonymous credentials. *TCC 2008*, *LNCS*, pages 356–374. Springer,2008.

[13] M.Belenkiy, M.Chase, M.Kohlweiss, and A.Lysyanskaya. Compact e-cash and simulatable VRFs revisited. *PAIRING 2009*, *LNCS*, pages 114–131. Springer, 2009.

[14] M.Bellare and G.Fuchsbauer. Policy-based signatures. Cryptology ePrint Archive, Report 2013/413, 2013. `http://eprint.iacr.org/`.

[15] M.Bellare, D.Micciancio, and B.Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. *EUROCRYPT 2003*, *LNCS*, pages 614–629. Springer, 2003.

[16] M.Bellare, H.Shi, and C.Zhang. Foundations of group signatures: The case of dynamic groups. *CT-RSA 2005*, *LNCS*, pages 136–153. Springer, 2005.

[17] D.Boneh and X.Boyen. Short signatures without random oracles. *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.

[18] E.Boyle, S.Goldwasser, and I.Ivan. Functional signatures and pseudorandom functions. Cryptology ePrint Archive, Report 2013/401. `http://eprint.iacr.org/`.

[19] S.Brands. Untraceable off-line cash in wallets with observers (extended abstract). *CRYPTO'93*, volume 773 of *LNCS*, pages 302–318. Springer, 1994.

[20] S.Brands. Restrictive blinding of secret-key certificates. *EUROCRYPT'95*, volume 921 of *LNCS*, pages 231–247. Springer, 1995.

[21] E. F. Brickell, J.Camenisch, and L.Chen. Direct anonymous attestation. *ACM CCS 04*, pages 132–145. ACM Press, 2004.

[22] C.Brzuska, H.Busch, Ö.Dagdelen, M.Fischlin, M.Franz, S.Katzenbeisser, M.Manulis, C.Onete, A.Peter, B.Poettering, and D.Schröder. Redactable signatures for tree-structured data: Definitions and constructions. *in ACNS'10.*

[23] C. Brzuska, M.Fischlin, A.Lehmann, and D.Schröder. Unlinkability of sanitizable signatures. *PKC*, volume 6056 of *LNCS*, pages 444–461. Springer, 2010.

[24] J.Camenisch, M.Dubovitskaya, A.Lehmann, G.Neven, C.Paquin, and F.-S. Preiss. Concepts and languages for privacy-preserving attribute-based authentication. IFIP IDMAN 2013, Springer, IFIP AICT, Vol. 396.

[25] J.Camenisch, M.Dubovitskaya, G.Neven, and G. M. Zaverucha. Oblivious transfer with hidden access control policies. *PKC 2011,LNCS*, pages 192–209.

[26] J.Camenisch and T.Groß. Efficient attributes for anonymous credentials. *ACM CCS 08*, pages 345–356. ACM Press, 2008.

[27] J.Camenisch, S.Hohenberger, and A.Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *EUROCRYPT*, *LNCS*, pages 302–321. Springer, 2005.

[28] J.Camenisch, A.Kiayias, and M.Yung. On the portability of generalized schnorr proofs. *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 425–442. Springer, 2009.

[29] J.Camenisch, S.Krenn, and V.Shoup. A framework for practical universally composable zero-knowledge protocols. *ASIACRYPT 2011*, *LNCS*, pages 449–467.

[30] J.Camenisch and A.Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, 2001.

[31] J.Camenisch and A.Lysyanskaya. A signature scheme with efficient protocols. *SCN 02*, volume 2576 of *LNCS*, pages 268–289. Springer, 2002.

[32] J.Camenisch and A.Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. *CRYPTO 2004*, *LNCS*, pages 56–72. Springer, 2004.

[33] J.Camenisch, G.Neven, and A.Shelat. Simulatable adaptive oblivious transfer. *EUROCRYPT 2007*, *LNCS*, pages 573–590. Springer, 2007.

[34] J.Camenisch and E.Van Herreweghen. Design and implementation of the idemix anonymous credential system. *ACM CCS 02*, pages 21–30. ACM Press, 2002.

[35] S.Canard and R.Lescuyer. Protecting privacy by sanitizing personal data: a new approach to anonymous credentials. ASIA CCS '13, ACM, pages 381–392.

[36] R.Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, 2001.

[37] R.Canetti, O.Goldreich, and S.Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998.

[38] D.Catalano and D.Fiore. Vector commitments and their applications. *PKC 2013*, volume 7778 of *LNCS*, pages 55–72. Springer, 2013.

[39] D.Catalano and D.Fiore. Vector commitments and their applications. *PKC 2013*.

[40] D.Catalano, D.Fiore, and M.Messina. Zero-knowledge sets with short proofs. *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 433–450. Springer, 2008.

[41] M.Chase, M.Kohlweiss, A.Lysyanskaya, and S.Meiklejohn. Malleable signatures: New definitions and delegatable anonymous credentials. In *IEEE 27th CSFS*, 2014.

[42] M.Chase and A.Lysyanskaya. On signatures of knowledge. *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer,2006.

[43] D.Chaum. Showing credentials without identification transferring signatures between unconditionally unlinkable pseudonyms. *AUSCRYPT'90*, pp. 246–264.

[44] D.Chaum, A.Fiat, and M.Naor. Untraceable electronic cash. *CRYPTO'88*, volume 403 of *LNCS*, pages 319–327. Springer, 1990.

[45] I.Damgård. Payment systems and credential mechanisms with provable security against abuse by individuals. *CRYPTO'88*, *LNCS*, pages 328–335. Springer, 1990.

[46] M.Fischlin. Round-optimal composable blind signatures in the common reference string model. *CRYPTO 2006*, volume 4117 of *LNCS*, pages 60–77. Springer, 2006.

[47] G.Fuchsbauer. Commuting signatures and verifiable encryption. *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 224–245. Springer, 2011.

[48] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. EUF-CMA-secure structure-preserving signatures on equivalence classes. Cryptology ePrint Archive, Report 2014/944, 2014. `http://eprint.iacr.org/`.

[49] V.Goyal. Reducing trust in the PKG in identity based cryptosystems. *CRYPTO 2007*, *LNCS*, pages 430–447. Springer, 2007.

[50] J.Groth and A.Sahai. Efficient non-interactive proof systems for bilinear groups. *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, 2008.

[51] S.Haber, Y.Hatano, Y.Honda, W.Horne, K.Miyazaki, T.Sander, S.Tezoku, and D.Yao. Efficient signature schemes supporting redaction, pseudonymization, and data deidentification. *ASIACCS 08*, pages 353–362. ACM Press, 2008.

[52] M.Hirt and K.Sako. Efficient receipt-free voting based on homomorphic encryption. *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 539–556. Springer, 2000.

[53] D.Hofheinz and V.Shoup. Gnuc: A new universal composability framework. *IACR Cryptology ePrint Archive*, 303, 2011.

[54] M.Izabachène, B.Libert, and D.Vergnaud. Block-wise p-signatures and non-interactive anonymous credentials with efficient attributes. In *IMA Int. Conf.*, pp. 431–450. Springer, 2011.

[55] A.Kate, G. M. Zaverucha, and I.Goldberg. Constant-size commitments to polynomials and their applications. *ASIACRYPT 2010*, *LNCS*, pages 177–194.

[56] A.Kiayias and M.Yung. Group signatures with efficient concurrent join. *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 198–214. Springer, 2005.

[57] A.Kiayias and H.-S.Zhou. Equivocal blind signatures and adaptive uc-security. In *TCC*, *Lecture Notes in Computer Science*, pages 340–355. Springer, 2008.

[58] M.Kohlweiss and A.Rial. Optimally private access control. In WPES '13, pages 37–48, ACM, 2013.

[59] B.Libert and M.Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. *TCC 2010*, *LNCS*, pages 499–517.

[60] S.Meiklejohn. An extension of the groth-sahai proof system. *Masters thesis*, 2009.

[61] R.Nojima, J.Tamura, Y.Kadobayashi, and H.Kikuchi. A storage efficient redactable signature in the standard model. *ISC 2009*, volume 5735 of *LNCS*, pages 326–337. Springer, 2009.

[62] C.Hanser and D.Slamanig. Structure-Preserving Signatures on Equivalence Classes and their Application to Anonymous Credentials. ASIACRYPT 2014.

The rest of this page intentionally left blank.

A Proof of Theorem 1

First, we show how the ideal functionality is implemented with the real algorithms of URS from Definition 1.

We define protocol $\mathcal{R}_{\mathsf{URS}}$ as follows:

Realization $\mathcal{R}_{\mathsf{URS}}$:

- On input $(\mathtt{keygen}, sid)$ from S, verify that $sid = (S, sid')$ for some sid' and that flag *kg* is unset. If not, then ignore the request. Else, ask $\mathcal{F}_{CRS}$ for SP, run $(pk, sk) \leftarrow \mathsf{Kg}(SP)$, set flag *kg*, store (pk, sk) and return $(\mathtt{verificationKey}, sid, pk)$ to S.
- On input $(\mathtt{checkPK}, sid, pk')$ from some party P, compute $b = \mathsf{CheckPK}(pk')$ and return $(\mathtt{checkedPK}, sid, b)$ to P.
- On input $(\mathtt{leakSK}, sid)$ from S verify that $sid = (S, sid')$ for some sid'. If not, ignore the request. Else, if flag *kg* is set, set flag *corrupt* and return $(\mathtt{leakSK}, sid, sk)$.
- On input $(\mathtt{sign}, sid, \vec{m})$ from S, verify that $sid = (S, sid')$ for some sid' and that flag *kg* is set. If not, ignore the request. Else, return the output of $\mathsf{Sign}(sk, \vec{m})$ to S.
- On input $(\mathtt{derive}, sid, pk', I, \vec{m}, \sigma)$ from some party P return the output of $\mathsf{Derive}(pk', I, \vec{m}, \sigma)$.
- On input $(\mathtt{verify}, sid, pk', \sigma, \vec{m}_I)$ from some party P, return the result of $\mathsf{Verify}(pk', \sigma, I, \vec{m}_I)$.

Theorem 1. *Let* URS *be the unlinkable redactable signature scheme according to Definition 1. Then* $\mathcal{R}_{\mathsf{URS}}$ *securely realizes* $\mathcal{F}_{\mathsf{URS}}$ *in the* $\mathcal{F}_{CRS}$*-hybrid model if* URS *is correct, parameter indistinguishable, key extractable, unforgeable, and composable unlinkable.*

Proof. We prove the theorem by proving the indistinguishability of the ideal world and the real world. In the ideal world, the environment $\mathcal{Z}$ interacts with the simulator $\mathcal{SIM}$ and the ideal functionality $\mathcal{F}_{\mathsf{URS}}$. In the real world, the environment $\mathcal{Z}$ interacts with the real adversary $\mathcal{A}$ and the protocol $\mathcal{R}_{\mathsf{URS}}$. For any real world adversary we construct an ideal world adversary such that if the environment can distinguish weather it interacts in the real or in the ideal world, one can construct a reduction algorithm that uses $\mathcal{Z}$ to break correctness, key extractability, parameter indistinguishability, unforgeability, and unlinkability of the redactable signature scheme.

First, we provide a description of the simulator in the ideal world. When receiving $(\mathtt{initF}, sid)$ message from $\mathcal{F}_{\mathsf{URS}}$, $\mathcal{SIM}$ generates (SP, td) by running $\mathsf{SGenT}(1^\kappa)$ algorithm, stores td and returns $(\mathtt{initF}, sid, SP, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Derive}, \mathsf{Verify})$. On receiving $(\mathtt{checkPK}, sid, pk)$ message, $\mathcal{SIM}$ uses the algorithm $\mathsf{ExtractKey}(pk, td)$ to extract the secret key sk and returns sk to $\mathcal{F}_{\mathsf{URS}}$. If extraction fails, it returns $\perp$.

The proof is done through a sequence of games. In Game 0, $\mathcal{Z}$ interacts with $\mathcal{F}_{\mathsf{URS}}$ and $\mathcal{SIM}$ as in the ideal world, and, in Game 6, $\mathcal{Z}$ interacts with $\mathcal{A}$ and $\mathcal{R}_{\mathsf{URS}}$ as in the real world. In each game we make a step towards the real world and show that if the environment can distinguish between the current and the previous game, then one can build an adversary that breaks a security property of URS. We define the success probability of $\mathcal{Z}$ to distinguish between Game i and Game j as $|p_i - p_j|$. We show that $|p_6 - p_0| \leq p_{corr} + p_{crs-ind} + q \cdot p_{unlink} + p_{unf} + p_{key-extr}$, where $p_{corr}, p_{crs-ind}, p_{unlink}, p_{unf}$, and $p_{key-extr}$ are the probabilities of breaking the correctness, parameter indistinguishability, unlinkability, unforgeability, and key extraction properties of the redactable signature scheme, respectively.

Game 0: *implements the ideal world, where the environment $\mathcal{Z}$ interacts with the simulator $\mathcal{SIM}$ and the functionality $\mathcal{F}_{\mathsf{URS}}$ as described above.*

Game 0′: *is the same as Game 0 except the simulator $\mathcal{SIM}$ and the functionality $\mathcal{F}_{\mathsf{URS}}$ are subsumed in a single entity that we denote $\mathcal{C}$, i.e., $\mathcal{C}$ runs $\mathcal{SIM}$ and $\mathcal{F}_{\mathsf{URS}}$ internally and interacts with the environment on behalf of $\mathcal{SIM}$ and $\mathcal{F}_{\mathsf{URS}}$.*

The change from Game 0 to Game 0′ is purely conceptual and hence $|p_{0'} - p_0| = 0$.

Game 1: *Here $\mathcal{C}$ behaves as in Game 0′ except when receiving $(\texttt{derive}, sid, pk', I, \vec{m}, \sigma)$, $\mathcal{C}$ just returns the output of $\mathsf{Derive}(pk', I, \vec{m}, \sigma)$, in particular $\mathcal{C}$ will never compute a fresh signature $\sigma' \xleftarrow{\$} \mathsf{Sign}(sk_{tmp}, \mathsf{Zero}(\vec{m}, I))$ and return $\mathsf{Derive}(pk', I, \mathsf{Zero}(\vec{m}, I), \sigma')$.*

Using an hybrid argument, we prove that if $\mathcal{Z}$ can distinguish between Game 0′ and Game 1 then one can break the unlinkability of URS. We define Hybrid k as follows:

Hybrid k: *Upon input the i^{th} $(\texttt{derive}, sid, pk', I, \vec{m}, \sigma)$ query, the derived signature is computed as follows:*

- *for all $i < k$, $\mathcal{C}$ returns the output of $\mathsf{Derive}(pk', I, \vec{m}, \sigma)$ (as in Game 1);*
- *for $i \geq k$, $\mathcal{C}$ first runs $\mathsf{Derive}(pk', I, \vec{m}, \sigma)$ and if it fails, outputs $\perp$. Otherwise, if $pk = pk'$ then let $sk_{tmp} = sk$. If there is an entry $(pk', sk') \in \mathcal{K}$ recorded, let $sk_{tmp} = sk'$. Then $\mathcal{C}$ runs $\sigma' \xleftarrow{\$} \mathsf{Sign}(sk_{tmp}, \mathsf{Zero}(\vec{m}, I))$ and returns $\mathsf{Derive}(pk', I, \mathsf{Zero}(\vec{m}, I), \sigma')$. Otherwise, $\mathcal{C}$ returns the output of $\mathsf{Derive}(pk', I, \vec{m}, \sigma)$ (as in Game 0′).*

One can see that Hybrid 0 is the same as Game 0′ and Hybrid q is the same as Game 1. Now, we show that if the environment can distinguish between Hybrid k and Hybrid k-1, then we can construct the adversary $\mathcal{A}_{unlink}$ that wins the unlinkability game for URS. The reduction is as follows. Upon the k^{th} input $(\texttt{derive}, sid, pk', I, \vec{m}, \sigma)$, $\mathcal{A}_{unlink}$ runs the verification protocol, sets messages $\vec{m}_0 = \vec{m}$ and $\vec{m}_1 = \mathsf{Zero}(\vec{m}, I)$, computes two signatures $\sigma_0 = \mathsf{Sign}(sk, \vec{m}_0)$ and $\sigma_1 = \mathsf{Sign}(sk, \vec{m}_1)$, outputs the tuple $(pk, I, \vec{m}_0, \vec{m}_1, \sigma_0, \sigma_1)$ to the unlinkability challenger, and waits for the challenge signature. When it receives the challenge derived signature $\sigma_I^{(b)}$ it forwards it to the environment. It is easy to see that if $b = 1$ then it is Hybrid k-1, and if $b = 0$ then it is Hybrid k. Thus, if the environment can distinguish between the two hybrids, $\mathcal{A}_{unlink}$ can distinguish between $\sigma_I^{(0)}$ and $\sigma_I^{(1)}$. Thus, summing up over all hybrids, one can see that $|p_1 - p_{0'}| \leq q \cdot p_{unlink}$.

Game 2: *Here $\mathcal{C}$ behaves as in Game 1 except when the environment sends the message $(\texttt{checkPK}, sid, pk')$, $\mathcal{C}$ computes $b = \mathsf{CheckPK}(pk')$ and returns $(\texttt{checkedPK}, sid, b)$. In particular, $\mathcal{C}$ does not attempt to extract the secret key from pk'.*

If the environment can distinguish between Games 1 and 2, then we can build an adversary $\mathcal{A}_{key-extr}$ that breaks the key extractability of URS. Let E be the event when $\mathsf{ExtractKey}()$ algorithm fails to output a valid sk even though $\mathsf{CheckPK}(pk)$ outputs 1. As long as this event does not occur, $\mathcal{Z}$'s view in Game 2 and Game 1 is identical. Now, the reduction works as follows. When receiving an $(\texttt{keygen}, sid)$ message, $\mathcal{A}_{key-extr}$ uses SP received from the challenger. On input $(\texttt{checkPK}, sid, pk')$ $\mathcal{A}_{key-extr}$ runs $\mathsf{ExtractKey}(pk)$ and, if it fails, outputs pk to the challenger. One can see that when event E happens then $\mathcal{A}_{key-extr}$ breaks the key extractability property. Therefore, $|p_2 - p_1| \leq p_{key-extr}$.

Game 3: *Here $\mathcal{C}$ behaves as in Game 2 except when the environment sends the message $(\texttt{keygen}, sid)$, $\mathcal{C}$ computes SP using $\mathsf{SGen}()$ algorithm instead of SGenT.*

If the environment can distinguish between Games 2 and 3, then we can build an algorithm $\mathcal{A}_{crs-ind}$ that breaks the parameter indistinguishability of URS. Let $b = 0$ if it is Game 2 (since in Game 2 SP is generated using SGenT) and $b = 1$ if it is Game 3 (since in Game 3 SP is generated using SGen). Then $\mathcal{A}_{crs-ind}$ just forwards the output of $\mathcal{Z}$ to the parameter indistinguishability challenger. Hence $|p_3 - p_2| \leq p_{crs-ind}$.

Game 4: *Here $\mathcal{C}$ behaves as in Game 3 except when the environment sends* $(\mathtt{sign}, sid, \vec{m})$*, $\mathcal{C}$ does not run the verification algorithm before outputting the signature.*

Let E_1 be the event when $\mathsf{Verify}(pk, \sigma, \vec{m})$ algorithm outputs 0 even though σ is output of the signing algorithm run with the corresponding secret key and the message vector. As long as this event does not occur, $\mathcal{Z}$'s view in Game 3 and Game 4 is identical. One can see that when event E happens then the correctness property does not hold. Hence we can write that $|p_4 - p_3| \leq p_{corr}$.

Game 5: *Here $\mathcal{C}$ behaves as in Game 4 except when receiving verification request input* $(\mathtt{verify}, sid, pk', \sigma, \vec{m}_I)$*, $\mathcal{C}$ returns the result of* $\mathsf{Verify}(pk', \sigma, \vec{m}_I)$.

Now, we show that if the environment can distinguish between Game 5 and Game 4 then we can construct the adversary $\mathcal{A}_{unf}$ that breaks the unforgeability of the redactable signature scheme. The reduction algorithm $\mathcal{A}_{unf}$ works as follows:

- On input $(\mathtt{keygen}, sid)$ from S, verify that $sid = (S, sid')$ for some sid' and that flag kg is unset. If not, then ignore the request. Get pk from the unforgeability challenger, set flag kg, store $(pk$ and return $(\mathtt{verificationKey}, sid, pk)$ to S.
- On input $(\mathtt{sign}, sid, \vec{m})$ from S, verify that $sid = (S, sid')$ for some sid' and that flag kg is set. If not, ignore the request. Else, ask the signing oracle $\mathsf{Sign}(\cdot)$ for the signature σ on $\vec{m}$ and output $(\mathtt{signature}, sid, \vec{m}, \sigma)$ to S.
- On input $(\mathtt{verify}, sid, pk', \sigma, I, \vec{m}_I)$ from some party P, compute $result \leftarrow \mathsf{Verify}(pk, \sigma, I, \vec{m}_I)$ and do the following: If $result = 1$ and $\nexists\ \vec{m}'$ such that $\vec{m} = \mathsf{Zero}(m')$ and m' was not queried before, then output (m, σ) as a forgery to the unforgeability challenger and $(\mathtt{verified}, sid, \vec{m}, result)$ to $\mathcal{Z}$. Otherwise, continue the simulation.

Let us analyze the success probability of $\mathcal{A}_{unf}$. Let E' denote an event when $\mathcal{Z}$ sends a verification request $(\mathtt{verify}, sid, \vec{m}_I, pk)$ on a message that was never signed before (or that cannot be derived from any message $\vec{m}'$ that was signed before: $\vec{m} \neq I(\vec{m})$ and the signer S is uncorrupted. As long as the event E' does not occur, $\mathcal{Z}$'s view between Game 4 and Game 5 is the same. Therefore, the probability that event E' occurs is greater equal than the probability that $\mathcal{Z}$ distinguish between Game 4 and Game 5. Notice, that event E' can occur only before the signer S is corrupted (or the secret key gets leaked). This means that whenever event E' occurs, $\mathcal{A}_{unf}$ outputs a successful forgery. So the probability of the event E' is actually the probability of the adversary $\mathcal{A}_{unf}$ to win the unforgeability game. Thus, we have that $|p_5 - p_4| \leq p_{unf}$.

Game 6: *is the realization of the real world where $\mathcal{C}$ is replaced with real $\mathcal{A}$ and π_{URS}.*

One can see that Game 5 is the same as Game 6, since already in Game 5 $\mathcal{C}$ runs $\mathcal{A}$ and $\mathcal{R}_{\mathsf{URS}}$ internally and interacts with the environment on behalf of $\mathcal{A}$ and $\mathcal{R}_{\mathsf{URS}}$. So the change from Game 5 to Game 6 is purely conceptual and hence $|p_6 - p_5| = 0$.

By summing up over all games, the probability that the environment can distinguish between interacting with the ideal functionality $\mathcal{F}_{\mathsf{URS}}$ and $\mathcal{SIM}$ and with the real protocol $\mathcal{R}_{\mathsf{URS}}$ and $\mathcal{A}$ is $|p_6 - p_0| \leq p_{corr} + p_{crs-ind} + q \cdot p_{unlink} + p_{unf} + p_{key-extr}$. $\square$

B Vector Commitments and Re-randomization

B.1 Definitions of Commitment Schemes

Definition 11 (Commitment Scheme)**.** *A commitment scheme* Com *is a triple of PT algorithms* $\mathsf{Com}.\{\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify}\}$*, where*

$\mathsf{Com}.\mathsf{Setup}(1^\kappa) \xrightarrow{\$} pp$ *is a common parameter generator that takes a security parameter 1^κ and outputs a set of public parameters pp that determine the message space $\mathcal{M}_{\mathsf{Com}}$.*

$\mathsf{Com}.\mathsf{Commit}(pp, m) \xrightarrow{\$} (C, O)$ *is a commitment function that takes common parameters pp and a message m and outputs a commitment C and opening information O.*

$\mathsf{Com.Check}(pp, C, m, O)$ *is a verification function that takes parameters, the commitment, the message and the opening as input, and outputs* 1 *or* 0 *representing acceptance or rejection, respectively.*

We note that in some cases the opening can be an input to the Commit algorithm.

We say that a commitment scheme Com is secure if it satisfies the following definitions of perfect hiding and computational binding.

Definition 12 (Perfect Hiding). *A commitment scheme* Com *is perfectly hiding if for any adversary* $\mathcal{A}$ *and security parameter* κ *the following holds:*

$$\Pr[pp \overset{\$}{\leftarrow} \mathsf{Com.Setup}(1^\kappa); (m_0, m_1, state) \leftarrow \mathcal{A}(pp); b \overset{\$}{\leftarrow} \{0,1\}; \\ (C, O) \overset{\$}{\leftarrow} \mathsf{Com.Commit}(pp, m_{(b)}); \mathcal{A}(state, C) \rightarrow b' : b = b'] = \frac{1}{2}.$$

Definition 13 (Computational Binding). *A commitment scheme* Com *is computationally binding if for any PPT adversary* $\mathcal{A}$ *and security parameter* κ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr[pp \overset{\$}{\leftarrow} \mathsf{Com.Setup}(1^\kappa); \mathcal{A}(pp) \rightarrow (C, m_0, m_1, O_0, O_1) : (m_0 \neq m_1) \wedge \\ \mathsf{Com.Check}(pp, C, m_0, O_0) = \mathsf{Com.Check}(pp, C, m_1, O_1) = 1] \leq \mathsf{negl}(\kappa).$$

We also provide the definition of a Structure-Preserving Commitment Scheme from Abe et al. [5].

Definition 14 (Strictly Structure-Preserving Commitment Scheme [5]). *A commitment scheme* Com *is strictly structure-preserving* SPC *with respect to a bilinear group generator* $\mathcal{G}$ *if*

- SPC.Setup *outputs pp that consist of the group description* $grp = (p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_t, e, G, \tilde{G})$ *and group elements in* $\mathbb{G}$ *and* $\tilde{\mathbb{G}}$*;*
- *the message space* $\mathcal{M}_{\mathsf{SPC}}$ *consists of group elements in* $\mathbb{G}$ *and* $\tilde{\mathbb{G}}$*;*
- SPC.Commit *outputs the commitment and the opening that consist of group elements in* $\mathbb{G}$ *and* $\tilde{\mathbb{G}}$*;and*
- SPC.Check *checks only membership in* $\mathbb{G}$ *and* $\tilde{\mathbb{G}}$ *and pairing product equations over grp.*

The authors of [5] call such scheme "strictly" structure-preserving, since parameters, messages, commitments, and openings are allowed to consist only of the group elements from $\mathbb{G}$ and $\tilde{\mathbb{G}}$ (and the group description for parameters). Some works, however, for example [3], relax this definition and also allow the target group elements (from $\mathbb{G}_t$) in the output spaces of the algorithms.

B.2 Vector Commitments

Vector commitments (VC) [39, 59] allow to commit to a vector of values in such a way that it is possible to open the commitment only w.r.t. specific positions.

Definition 15 (Vector Commitment Scheme). *A Vector Commitment scheme is a non-interactive primitive that consists of the following algorithms:*

$\mathsf{VC.Setup}(1^\kappa, \ell) \overset{\$}{\rightarrow} pp$. *Given the security parameter* 1^κ *and the size* ℓ *of the committed vector (with* $\ell = poly(\kappa)$*), the key generation outputs some public parameters pp, which implicitly define the message space* $\mathcal{M}$.

$\mathsf{VC.Commit}(pp, \vec{m}, r) \xrightarrow{\$} C$. *On input a sequence (vector) of ℓ messages $\vec{m} = (m_1, \ldots, m_\ell)$, randomeness r and the public parameters pp, the committing algorithm outputs a commitment string C.*

$\mathsf{VC.Open}(pp, i, \vec{m}, r) \xrightarrow{\$} W$. *This algorithm is run by the committer to produce a witness W that m is the i^{th} committed message. the auxiliary information aux can include the update information produced by these updates.*

$\mathsf{VC.Check}(pp, C, x, i, W) \to 0/1$. *The verification algorithm accepts (i.e., it outputs 1) only if W is a valid witness that C was created to a vector $m_1, \ldots m_\ell$ such that $x = m_i$.*

B.3 Proofs of the Theorems from §3.1

Theorem 2. *The commitment scheme $\mathsf{VC}.\{\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Check}\}$ defined in §3.1 is batch binding under the n-BSDH assumption.*

The proof of the theorem is analogous to that of [55] and could be omitted. Nevertheless, we give a proof of the theorem as a simple corollary of Theorem 9 presented in the next section.

Theorem 3. *The commitment scheme $\mathsf{VC}.\{\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Check}\}$ defined in §3.1 is opening non-malleable under the J-RootDH assumption.*

Proof. The implicit quantification over all $J \subseteq [1, n]$ gives us some flexibility as to which instance to attack. Alternatively, one can interpret this assumption as an interactive assumption in which the adversary is allowed a one-time query for J.

We follow the steps of the game to build a reduction:

1. We create pp as usual using the $G, \tilde{G}, \{G^{\alpha^i}, \tilde{G}^{\alpha^i}\}_{i=1}^{n+1}$ that form part of every assumption instance and provide it to $\mathcal{A}$.
2. When $\mathcal{A}$ outputs $(\vec{m}, I)$ we will attack a J-RootDH instance with $J = [1, n] \setminus I$.
3. We compute $C = \prod_{i \in I} F_i^{m_i} \cdot X$. Similarly we represent $W = G^{w(\alpha)}$ as $G^{\frac{f(\alpha) - f_I(\alpha)}{p_I(\alpha)}} \cdot Y$. The reduction hands C and W to $\mathcal{A}$.
4. When $\mathcal{A}$ returns I', W' we know from the verification equations that

$$e(C, \tilde{G}) = e(W', \tilde{P}_{I'}) e(F_{\vec{m}_{I'}}, \tilde{G})$$

Let $w' = \log_G W'$. We have

$$\begin{aligned} f(\alpha) + r p(\alpha) &= w' p_{I'}(\alpha) + f_{I'}(\alpha) \\ r p(\alpha) / p_{I'}(\alpha) &= \left(\frac{f_{I'}(\alpha) - f(\alpha)}{p_{I'}(\alpha)} + w' \right) . \end{aligned}$$

Thus the reduction can break the J-RootDH assumption below by outputing $[1, n] \setminus I'$, $\left(G^{\frac{f_{I'}(\alpha) - f(\alpha)}{p_{I'}(\alpha)}} \cdot W' \right)$.

□

B.4 Re-randomizable Vector Commitments

We consider the vector commitment scheme $\mathsf{VC}.\{\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Check}\}$ from §3.1 together with a randomization algorithm:

$\mathsf{VC.Rand}(pp, C, W, I, \rho) \xrightarrow{\$} (C', W')$. Compute the re-randomized commitment $C' = C \cdot P_I^{\rho}$ and the witness $W' = W \cdot G^{\rho}$, and return a couple (C', W').

We revisit our batch-binding and opening non-malleability definitions in the presence of such randomization algorithm.

Definition 16 (Randomizable batch binding)**.** *For a vector commitment scheme* $\mathsf{VC}.\{\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Check}, \mathsf{Rand}\}$ *and an adversary* $\mathcal{A}$ *consider the following game:*

- *Step 1.* $grp \leftarrow \mathcal{G}(1^k)$ *and* $pp \leftarrow \mathsf{Setup}(grp)$
- *Step 2.* $C, I, \vec{m}, W, \Delta, C', I', \vec{m}', W', \Delta' \leftarrow \mathcal{A}(pp)$

Then the commitment scheme satisfies randomizable batch binding *if for all such PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that the probability (over the choices of* grp, Setup, *and* $\mathcal{A}$*) that* $\mathsf{Rand}(C, W, I, -\log_G \Delta) = \mathsf{Rand}(C', W', I', -\log_G \Delta')$, $1 = \mathsf{Check}(pp, C, I, \vec{m}, W) = \mathsf{Check}(pp, C', I', \vec{m}, W')$, *and that there exist* $i = i_j = i'_{j'}$ *such that* $m_i \neq m'_i$ *is at most* $\nu(k)$.

Note that this definition corresponds to the standard batch binding if Δ and Δ' are fixed to 1.

Definition 17 (Randomizable opening non-malleability)**.** *For a vector commitment commitment scheme* $\mathsf{VC}.\{\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Check}, \mathsf{Rand}\}$ *and an adversary* $\mathcal{A}$ *consider the following game:*

- *Step 1.* $grp \leftarrow \mathcal{G}(1^k)$ *and* $pp \leftarrow \mathsf{Setup}(grp)$
- *Step 2.* $\vec{m}, I \leftarrow \mathcal{A}(pp)$
- *Step 3. Pick random* r*, compute* $C \leftarrow \mathsf{Commit}(pp, \vec{m}), r)$*, and* $W \leftarrow \mathsf{Open}(pp, \vec{\imath}, \vec{m}, r)$.
- *Step 4.* $C', W', \vec{i'}, \Delta' \leftarrow \mathcal{A}(C, W)$

Then the commitment scheme satisfies opening non-malleability *if for all such PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that the probability (over the choices of grp,* Setup, Commit, *and* $\mathcal{A}$*) that* $\mathsf{Rand}(C, W, I', -\log_G \Delta') = (C', W')$, $1 = \mathsf{Check}(pp, C', I', \vec{m}_{\vec{i'}}, W')$, *and* $I \neq \vec{i'}$ *is at most* $\nu(k)$.

Theorem 9. *The vector commitment scheme* VC *is randomizable batch binding under the BSDH assumption.*

Proof. From the randomization and verification equations we know that

$$\begin{aligned} e(C, \tilde{G})e(\Delta, \tilde{P}_{\vec{i}}) &= e(W, \tilde{P}_{\vec{i}})e(F_{\vec{i}}, \tilde{G})e(\Delta, \tilde{P}_{\vec{i}}) = \\ &= e(W', \tilde{P}_{\vec{i'}})e(F'_{\vec{i'}}, \tilde{G})e(\Delta', \tilde{P}_{\vec{i'}}) = \\ &= e(C', \tilde{G})e(\Delta', \tilde{P}_{\vec{i'}}). \end{aligned}$$

If $\mathcal{A}$ is successful, there exists $i = i_j = i'_{j'}$ such that $m_{i_j} \neq m'_{i'_{j'}}$

The polynomials $f_{\vec{i}}, f'_{\vec{i'}}, p_{\vec{i}}, p_{\vec{i'}}$ can be factored as follows using polynomials ϕ, ϕ', π, π':

$$\begin{aligned} f_{\vec{i}}(x) &= \phi(x)(x - i) + m_i & f'_{\vec{i'}}(x) &= \phi'(x)(x - i) + m'_i \\ p_{\vec{i}}(x) &= \pi(x)(x - i) & p_{\vec{i'}}(x) &= \pi'(x)(x - i) \end{aligned}$$

We denote $w = \log_G W$, $w' = \log_G W'$, and $\rho = \log_{\tilde{G}} \Delta$, and rewrite the above equations expressed using exponents

$$\begin{aligned} p_{\vec{i}}(\alpha)w + f_{\vec{i}}(\alpha) + p_{\vec{i}}(\alpha)\rho &= p_{\vec{i'}}(\alpha)w' + \\ & \quad f'_{\vec{i'}}(\alpha) + p_{\vec{i'}}(\alpha)\rho' \\ \pi(\alpha)(\alpha - i)(w + \rho) + \phi'(\alpha)(\alpha - i) + m_i &= \pi'(\alpha)(\alpha - i)(w' + \rho') + \\ & \quad \phi'(\alpha)(\alpha - i) + m'_i \\ \left(\pi(\alpha)(w + \rho) - \pi'(\alpha)(w' + \rho') + \phi'(\alpha) - \phi'(\alpha)\right)(\alpha - i) &= m'_i - m_i \\ \frac{\pi(\alpha)(w + \rho) - \pi'(\alpha)(w' + \rho') + \phi'(\alpha) - \phi'(\alpha)}{m'_i - m_i} &= \frac{1}{\alpha - i} \end{aligned}$$

Now moving back to computing over group elements and using pairings to do multiplication, we get that

$$\Big(e(W,\tilde{G}^{\pi(\alpha)})e(G^{\pi(\alpha)},\Delta)e(W',\tilde{G}^{\pi'(\alpha)})^{-1}e(G^{\pi'(\alpha)},\Delta')^{-1}\cdot$$
$$e(G^{\phi'(\alpha)},\tilde{G})e(G^{\phi'(\alpha)},\tilde{G})^{-1}\Big)^{\frac{1}{m_i-m_i'}} = e(G,\tilde{G})^{\frac{1}{\alpha-i}}\ .$$

Which contradicts the BSDH assumption. □

Theorem 10. *The batch openings of* VC.{Setup, Commit, Open, Check, Rand} *are non-malleable under the J-RootDH assumption.*

Proof. The implicit all quantification over all $J \subseteq [1,n]$ gives us some flexibility as to which instance to attack. Alternatively we could make the assumption interactive.

We follow the steps of the game to build a reduction:

1. We create pp as usual using the $G, G^{\alpha}, \dots, G^{\alpha^{n+1}}$ part of every assumption instance and provide it to $\mathcal{A}$.
2. When $\mathcal{A}$ outputs $\vec{m}, I$ we will attack an J-RootDH assumption instance with $J = [1,n] \setminus I$.
3. We compute $C = \prod_{j=1}^{I} F_{i_j}^{m_j} X$. Similarly we represent $W = G^{w(\alpha)}Y$. The reduction hands C and W to $\mathcal{A}$.
4. When $\mathcal{A}$ returns I', W' we know from the verification equations that

$$e(C',\tilde{G}) = e(W',\tilde{P}_{I'})e(F_{\vec{m}_{I'}},\tilde{G})$$

Let $w' = \log_G W'$, $\rho' = \log_G \Delta$. We have

$$f(\alpha) + rp(\alpha) + \rho' p_{I'}(\alpha) = w' p_{I'}(\alpha) + f_{I'}(\alpha)$$
$$rp(\alpha)/p_{I'}(\alpha) = \left(\frac{f_{I'}(\alpha) - f(\alpha)}{p_{I'}(\alpha)} + w' - \rho'\right)\ .$$

Thus, the reduction can break the J-RootDH assumption by outputting $[1,n] \setminus I'$, $\left(G^{\frac{f_{I'}(\alpha)-f(\alpha)}{p_{I'}(\alpha)}} \cdot W/\Delta'\right)$.

□

C Security and Instantiation of Our Redactable Signature Scheme

C.1 Unforgeability and Unlinkability Proofs

Theorem 4. URS.{SGen, Kg, Sign, Derive, Verify} *is an unforgeable redactable signature scheme, if the SPS scheme is unforgeable, the vector commitment scheme satisfies the batch binding and opening non-malleability property, and the proof-of-knowledge system is witness indistinguishable.*

Proof. The proof is done through the sequence of games.

We proceed with the proof through a sequence of games. Game 0 is the original unforgeability game, while Game 5 is such that the adversary has zero probability of winning it. We denote the success probability of $\mathcal{A}$ in Game i as p_i.

Game 0: *is the original malleable signatures unforgeability game;* $p_0 = p_{qbsig-unf}$.

Game 1: *is the same as the previous game except that* Derive *computes a fresh commitment and signature, i.e.,* $r \leftarrow \mathbb{Z}_p$, $C \leftarrow \mathsf{VC.Commit}(pp, \vec{m}, r)$, $W \leftarrow \mathsf{VC.Open}(pp, I, \vec{m}, r)$, $\Sigma \leftarrow \mathsf{SPS.Sign}(sk_{sps}, C)$, *and compute* $\pi \leftarrow \Pi.\mathsf{Prove}(CRS; (C, W, \psi_{\mathtt{wit}}(\Sigma)); \exists C, W, \psi_{\mathtt{wit}}(\Sigma) : \mathsf{SPS.Verify}(pk_{sps}, \Sigma, C) \wedge \mathsf{VC.Check}(pp, C, I, \vec{m}_I, W)\ )$ *to return* $(\psi_{rnd}(\Sigma), \pi)$.
This change is perfectly indistinguishable because of the witness indistinguishability of the proof system and the perfect randomization of the $\psi_{\mathtt{rnd}}(\Sigma)$, hence $|p_1 - p_0| \leq p_{nip-wi}$.

Game 2: *proceeds like Game 1, but the proof-of-knowledge system parameters are switched to extraction parameters and using the extraction trapdoor, the SPS signature* σ^* *and the commitment* C^* *of the forgery are extracted;* $|p_2 - p_1| \leq p_{nip-ext}$.

Game 3: *simulates the view of adversary in the same way as Game 2; except that if* (σ^*, C^*) *are not a signature/message pair produced by any of the signing oracles, the forgery is rejected.*
The success probability of the adversary decreases at most by the success probability of breaking unforgeability for the SPS scheme, i.e. $|p_3 - p_2| \leq p_{sps-unf}$.

Game 4: *is the same as Game 3, except for rejecting any forgery which contains* $C, I^*, \vec{m}^*$ *such that the* Sign *or* Derive *query corresponding to* C *was made for a message* $\vec{m}$, *but* $\vec{m}^* \neq I^*(\vec{m})$.
The success probability of the adversary decreases at most by the success probability of breaking the binding property of the commitment scheme (since the adversary would have to open a commitment to a different value). Thus, $|p_3 - p_2| \leq p_{com-bind}$.

Game 5: *proceeds like Game 4, but any forgery for which the extracted commitment* C *corresponds to a* Derive *query with a set* I, *but* $I^* \neq I$;
The success probability of the adversary decreases at most by the success probability of breaking the non-malleability property of the vector commitment scheme. (Since the output of Derive contains only (hiding) ZK proofs of the witness, but not the witness itself, one-time non-malleability is sufficient for here.) Thus, $|p_5 - p_4| \leq p_{com-nmal}$.
Note that as both I^* and $I^*(\vec{m}) = \vec{m}'$ are bound to queried values, the success probability of $\mathcal{A}$ in Game 5 is 0.

Overall, the adversary's probability of winning Game 0 is $p_{qbsig-unf} \leq p_{nip-wi} + p_{nip-ext} + p_{sps-unf} + p_{com-bind} + p_{com-nmal}$.

□

Theorem 5. $\mathsf{URS}.\{\mathsf{SGen}, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Derive}, \mathsf{Verify}\}$ *is an unlinkable redactable signature scheme if the proof-of-knowledge system is witness indistinguishable.*

Proof. We define two experiment $\mathcal{E}_0$ and $\mathcal{E}_1$ which correspond to the experiment in the unlinkability game defintion (cf. Definition 3) with, respectively, $b = 0$ and $b = 1$ chosen in the step 3.

$\mathcal{E}_0$ and $\mathcal{E}_1$ are indistinguishable due to the perfect witness indistinguishability of the proof system and the perfect randomization of the $\psi_{\mathtt{rnd}}(\Sigma')$. Hence, $p_{qbsig-unlk} = p_{nip-wi}$. □

C.2 Alternative Construction

This construction utilizes the $\mathsf{VC.Rand}$ algorithm of the vector commitment scheme.

$\mathsf{URS}.\{\mathsf{SGen}, \mathsf{Kg}, \mathsf{Sign}\}$ algorithms are unchanged.

$\mathsf{URS.Derive}(pk, \vec{m}, I, \sigma)$. If the SPS scheme has a $\mathsf{SPS.Rand}$ algorithm, randomize the signature as
$\Sigma' \leftarrow \mathsf{SPS.Rand}(pk_{sps}, \Sigma)$; otherwise, set $\Sigma' \leftarrow \Sigma$. Compute $W = \mathsf{VC.Open}(pp, I, \vec{m}, r)$ and $(C', W') \leftarrow \mathsf{VC.Rand}(pp, C, W, I, \rho)$ for a randomly chosen $\rho \leftarrow \mathbb{Z}_p$. Finally, let
$\pi \leftarrow \Pi.\mathsf{Prove}\left(CRS; (\psi_{\mathtt{wit}}(\Sigma'), \rho); \exists\, \psi_{\mathtt{wit}}(\Sigma'), \rho : \mathsf{SPS.Verify}(pk_{sps}, \Sigma', C'/P_I^{\rho})\right)$.
Return $\sigma = (\psi_{\mathtt{rnd}}(\Sigma'), C', W', \pi)$ as the signature for $I, \vec{m}_I$.

$\mathsf{URS.Verify}(pk, \sigma, I, \vec{m}_I)$. Check that
$\Pi.\mathsf{Verify}\big(CRS; \pi; \reflectbox{K}\, \psi_{\mathtt{wit}}(\Sigma'), \rho \,:\, \mathsf{SPS.Verify}(pk_{sps}, \Sigma', C'/P_I^{\rho})\big) \;=\; \mathsf{VC.Check}(pp, C', \vec{m}_I, W') = 1$.

Theorem 11. $\mathsf{URS} = (\mathsf{SGen}, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Derive}, \mathsf{Verify})$ *of the alternative construction is an* unforgeable *and* unlinkable *redactable signature scheme, if the SPS scheme is unforgeable, the vector commitment scheme satisfies the batch binding and opening non-malleability property, and the proof-of-knowledge system is witness indistinguishable.*

The proof of the theorem follows similarly to the proofs of Theorem 4 and Theorem 5 using the randomizable properties of the commitment described in §B.4. We defer the proof to the full version of the paper.

C.3 Instantiation

To instantiate our scheme presented in §3.3 we use the randomizable vector commitment scheme from §3.1, the SPS scheme(s) of Abe et al. [3], and the Groth-Sahai proof system already described in §3.2.

Theorem 12. *The redactable signature scheme presented in §3.3 is unforgeable, unlinkable, and key extractable if the SXDH, J-RootDH, and SFP [3] assumptions hold. It has tier 2 signatures of size* 19 *group elements.*

The security of the theorem is a specific case of Theorem 6. The signature size is computed as $13 + 2n$ (described below) for $n = 1$ when doing a GS proof for the signature and the signed message C plus a commitment of W and the proof for the Check equation, each of which requires two group elements.

Structure Preserving Signatures [3] Recall that we use a bilinear group $grp = (p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_t, e, G, \tilde{G})$ generated by $\mathcal{G}(1^k)$. For a message space $\mathbb{G}^n$, the signature algorithms are defined as follows:

$\mathsf{SPS.Kg}(grp)$. Pick at random $g_r, h_u \leftarrow \mathbb{Z}_p^*$ to compute $G_r = G^{g_r}, H_u = G^{h_u}, \tilde{G}_r^{g_r}, \tilde{H}_u^{h_u}$. For $i \in [1..n]$ pick at random $x_i, y_i \leftarrow \mathbb{Z}_p^*$ and compute $\tilde{G}_i = \tilde{G}_r^{x_i}, \tilde{H}_i = \tilde{H}_u^{y_i}$. Choose $x_z, y_z \leftarrow \mathbb{Z}_p^*$ and compute $\tilde{G}_z = \tilde{G}_r^{x_z}, \tilde{H}_z = \tilde{H}_u^{y_z}$. Also choose $a, b \leftarrow \mathbb{Z}_p^*$ and compute $A = e(G_r, \tilde{G}^a)$ and $B \leftarrow e(H_u, \tilde{G}^b)$. Return the verification key $pk = (grp, G_r, H_u, \tilde{G}_r, \tilde{H}_u, \tilde{G}_z, \tilde{H}_z, \{\tilde{G}_i, \tilde{H}_i\}_{i=1}^n, A, B))$ and the signing key $sk = (pk, a, b, x_z, y_z, \{x_i, y_i\}_{i=1}^n)$.

$\mathsf{SPS.Sign}(sk_{sps}, (M_1, \ldots, M_n))$. Pick random $z, r, t, u, w \leftarrow \mathbb{Z}_p^*$, and set:

$$Z = G^z,\; R = G^{a-rt-x_z z}\prod_{i=1}^{n} M_i^{-x_i},\; S = G_r^r,\; \tilde{T} = \tilde{G}^t,$$

$$U = G^{b-uw-y_z z}\prod_{i=1}^{n} M_i^{-x_i},\; V = H_u^u,\; \tilde{W} = \tilde{G}^w.$$

Output $\Sigma = (Z, R, S, \tilde{T}, U, \tilde{W})$ as a signature.

$\mathsf{SPS.Verify}(pk_{sps}, \Sigma, (M_1, \ldots, M_n))$. Parse σ as $(Z, R, S, \tilde{T}, U, V, \tilde{W})$ and check that

$$e(Z, \tilde{G}_z)e(R, \tilde{G}_r)e(S, \tilde{T})\prod_{i=1}^{n} e(M_i, \tilde{G}_i) = A \quad \wedge$$

$$e(Z, \tilde{H}_z)e(U, \tilde{H}_u)e(V, \tilde{W})\prod_{i=1}^{n} e(M_i, \tilde{H}_i) = B.$$

$\mathsf{SPS.Rand}(pk_{sps}, \Sigma)$. Parse σ as $(Z, R, S, \tilde{T}, U, V, \tilde{W})$ and return $(Z, R', S', \tilde{T}', U', V', \tilde{W}')$, where

$$R' = RS^{-\rho},\ S' = S^{\frac{1}{\tau}},\ \tilde{T}' = (\tilde{T}G_r^{\rho})^{\tau},\ U' = UV^{-\mu},\ V' = V^{\frac{1}{\varphi}},\ \tilde{W}' = (\tilde{W}H_u^{\mu})^{\varphi},$$

for randomly chosen $\rho, \tau, \mu, \varphi \leftarrow \mathbb{Z}_p$.

The scheme is existentially unforgeable under the Strong Flexible-Pairing assumption [3]. Its signatures Σ are of size 7, and Rand algorithm yields $\psi_{\mathtt{rnd}}(\Sigma) = (R, \tilde{T}, U, \tilde{W})$ and $\psi_{\mathtt{wit}}(\Sigma) = (Z, S, V)$. Therefore, when doing a GS proof of knowledge of a signature and a message, one can give 4 of the signature elements in the clear as part of the proof after a re-randomization, commit to $3 + n$ group elements, and produce proof elements for two one-sided equations; this yields a proof of size $3 + 2 * (3 + n) + 2 * 2 = 13 + 2n$.

We also note that one could use a different SPS scheme with smaller signature size. However, it yields more efficient redactable signatures only under stronger assumptions which are not non-interactive q-type. Hence, we prefer to use the above SPS scheme for our main instantiation.

Structure Preserving Signatures [4]

$\mathsf{SPS.Kg}(grp)$. Select $\alpha, \beta, \gamma, \gamma_1, \ldots, \gamma_n \leftarrow \mathbb{Z}_p^*$ and compute

$$H \leftarrow \tilde{G}^{\alpha},\ V \leftarrow \tilde{G}^{\beta},\ U \leftarrow \tilde{G}^{\gamma},\ \text{ and } U_i \leftarrow \tilde{G}^{\gamma_i} \text{ for } i \in [1, \ldots, n].$$

Output $pk_{sps} = (H, V, U, \{U_i\}_{i=1}^n)$ and $sk_{sps} = (\alpha, \beta, \gamma, \{\gamma_i\}_{i=1}^n)$.

$\mathsf{SPS.Sign}(sk_{sps}, (M_1, \ldots, M_n))$. Choose $\rho, \tau \leftarrow \mathbb{Z}_p^*$ and return $\Sigma = (R, \tilde{R}, S, T)$, where

$$R = G^{\rho}, \quad \tilde{R} = \tilde{G}^{\frac{1}{\rho}}, \quad S = G^{\alpha - \beta\rho - \gamma\tau} \prod_i M_i^{-\gamma_i}, \text{ and } \quad T = \tilde{G}^{\tau}.$$

$\mathsf{SPS.Verify}(pk_{sps}, \Sigma, (M_1, \ldots, M_n))$. Parse $\Sigma = (R, \tilde{R}, S, T)$ and accept if the following equations hold:

$$\begin{aligned} &e(R, V)e(S, \tilde{G})e(T, U) \prod_{i=1}^{n} e(M_i, U_i) = e(G, H) \\ &e(R, \tilde{R}) = e(G, \tilde{G}). \end{aligned}$$

The scheme is strongly existentially unforgeable against adaptive chosen message attack under a "q-type" non-interactive assumption. A Groth-Sahai NIWI proof of knowledge of a signature and a message for this scheme will consist of $18 + 2n$ group elements as one needs to commit to $4 + n$ group elements, each commitment consists of two group elements, and produce proof elements for one pairing-product equation and one one-sided pairing product equation, which require 8 and 2 group elements, respectively.

Note that the previous presented scheme [3] has larger signature size than the [4] scheme, when combined with GS proofs for proving knowledge of a signature and a message, the former yields slightly more efficient proofs. However, if one is willing to tolerate stronger assumptions, a more efficient variant of the latter scheme presented in the same paper can be used. Its verification equations are:

$$e(V, R) = e(G, S) \quad \wedge \quad e(T, R) \prod_i (M_i, U_i) = e(G, \tilde{G}),$$

where (R, S, T) is the signature. Moreover, the scheme supports full re-randomization of the R element of the signature. Hence, the cost of a NIWI proof of knowledge for that scheme is $9 + 2n$ if R is given in the clear as part of the proof after a re-randomization as one has to commit to $2 + n$ group elements and produce the proof elements for two one-sided pairing product equations.

D Extending URS to Sign Additional Group Elements

In this section we extend our redactable signature scheme to sign additional group elements besides the commitment. Namely, we extend Sign to take as additional input a vector $\vec{M}$ of group elements. For that we use a SPS signature that supports signing multiple messages. In the Derive algorithm these elements will either be part of the derived message, and given in the clear after derivation, or be treated as part of the witness, i.e., hidden from the verifier. To do so, we extend our input vector of exponents by appending a vector of group elements to it $(\vec{M}, \vec{m}) = (M_1, \ldots, M_N, m_{N+1}, \ldots, m_{N+|\vec{m}|})$. A set I applied to this extended vector $(\vec{M}, \vec{m})_I$, will now indicate not only which exponents, but also which group elements are revealed.

Construction. We denote the extended algorithms as $\mathsf{URS} = (\mathsf{SGen}, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Derive})$. We define the extended signature, derivation, and verification algorithms as follows.

$\mathsf{URS.SGen}(sk, \vec{M}, \vec{m})$. Compute $grp \leftarrow \mathcal{G}(1^k)$, $pp_{\mathsf{VC}} \leftarrow \mathsf{VC.Setup}(grp)$, $CRS \leftarrow \Pi.\mathsf{Setup}(grp)$, $(pp_{\mathsf{SPC}}, td_{com}) \leftarrow \mathsf{SPC.Setup}(grp)$.
Output $SP = (grp, pp_{\mathsf{VC}}, pp_{\mathsf{SPC}}, CRS)$.

$\mathsf{URS.Kg}(SP)$. Obtain grp from SP, generate $(pk_{sps}, sk_{sps}) \leftarrow \mathsf{SPS.Kg}(grp)$, output $pk = (pk_{sps}, SP)$ and $sk = (sk_{sps}, pk)$.

$\mathsf{URS.Sign}(sk, (\vec{M}, \vec{m}))$. Pick $r \leftarrow \mathbb{Z}_p$, compute $C = \mathsf{VC.Commit}(pp_{\mathsf{VC}}, \vec{m}, r)$ and $\Sigma \leftarrow \mathsf{SPS.Sign}(sk_{sps}, (C, \vec{M}))$, and return $\sigma = (\Sigma, C, r)$.

$\mathsf{URS.Derive}(pk, (\vec{M}, \vec{m}), I, \sigma)$. First, compute $W = \mathsf{VC.Open}(pp_{\mathsf{VC}}, I, \vec{m}, r)$. Then, if an $\mathsf{SPS.Rand}$ algorithm is present, randomize the signature as $\Sigma' \leftarrow \mathsf{SPS.Rand}(pk_{sps}, \Sigma)$; otherwise, set $\Sigma' \leftarrow \Sigma$.
Compute the proof $\pi \leftarrow \Pi.\mathsf{Prove}\big(CRS; (C, W, \psi_{\mathtt{wit}}(\Sigma')); S\big)$, where

$$S = \text{⅄}\, C, W, \psi_{\mathtt{wit}}(\Sigma') \; : \;$$
$$\mathsf{SPS.Verify}(pk_{sps}, \Sigma', (C, \vec{M})) = \mathsf{VC.Check}(pp_{\mathsf{VC}}, C, I, \vec{m}_I, W) = 1.$$

Output $(\sigma_I = \psi_{\mathtt{rnd}}(\Sigma'), \pi)$.

$\mathsf{URS.Verify}(pk, \sigma, (\vec{M}, \vec{m}))$. Check that $\Pi.\mathsf{Verify}\big(CRS; \pi; S\big) = 1$.

Theorem 13. *The extended signature scheme* $\mathsf{URS}.\{\mathsf{SGen}, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Derive}\}$ *is an unforgeable, unlinkable, and key extractable redactable signature scheme, if the underlying SPS scheme is unforgeable and randomizable, the vector commitment scheme is batch-binding and opening non-malleable, and the proof system is zero-knowledge and extractable.*

Proof. The proof is very similar to the proofs of Theorems 4-5 and is omitted here. □

E Details on Anonymous Credentials

E.1 Full Single Issuer Protocol.

Before proceeding to the details of the realization, let us also describe the functionality $\mathcal{F}_{\mathsf{CA}}$, which is a slightly modified version of the one provided by Canetti in [36]. More precisely, our $\mathcal{F}_{\mathsf{CA}}$ functionality additionally checks the correctness of the key. In the realization, this can be done by executing an interactive proof of knowledge of the secret key between the issuer and the certification authority. As shown by Camenisch et al. [29], such interactive proofs are efficient and still online extractable (a crucial property for the UC model). When implementing such scheme, the ideal functionalities $\mathcal{F}_{\mathsf{CA}}$ and $\mathcal{F}_{\mathsf{Cred}}$ will be replaced by their secure implementations.

Functionality $\mathcal{F}_{\mathsf{CA}}(\mathsf{CheckKeys})$

- Upon receiving the first message $(\texttt{registerIssuerKey}, sid, pk)$ from party $\mathcal{P}$, send $(\texttt{registeredIssuerKey}, sid, pk)$ to $\mathcal{SIM}$; upon receiving $(\texttt{registeredIssuerKey}, sid, sk)$ from $\mathcal{SIM}$, verify that $sid = (\mathcal{P}, sid')$ for some sid', that $\mathsf{CheckKeys}(sk, pk) = 1$, and this is the first request from $\mathcal{P}$, then record the pair (sid, pk).
- Upon receiving the first message $(\texttt{requestIssuerPK}, sid)$ from party $\mathcal{P}'$, send $(\texttt{requestIssuerPK}, sid, \mathcal{P}')$ to $\mathcal{SIM}$ and wait for an ok from $\mathcal{SIM}$. Then, if there is a recorded pair (sid, pk), output $(\texttt{requestIssuerPK}, sid, pk)$ to $\mathcal{P}'$. Otherwise output $(\texttt{requestIssuerPK}, sid, \perp)$ to $\mathcal{P}'$.

We now provide a realization $\mathcal{R}_{\mathsf{Cred}}$ of the anonymous credentials functionality, described above. The functionality makes use of $\mathcal{F}_{\mathsf{CRS}}$ that provide public parameters, and of the certification authority functionality $\mathcal{F}_{\mathsf{CA}}$ described above. $\mathcal{R}_{\mathsf{Cred}}$ also maintains flags kg and $keyleak$ that are initially unset. The network tapes $NET_{\mathcal{I}}$ and $NET_{\mathcal{U}}$ indicate messages being sent to the issuer and user protocol respectively. As is the default in UC, messages are always sent via the adversary.

Realization $\mathcal{R}_{\mathsf{Cred}}(\mathsf{SecGen}, \mathsf{NymGen}, \mathsf{NymVerify})$

- On input $(\texttt{keygen}, sid)$ from $\mathcal{I}$, verify that $sid = (\mathcal{I}, sid')$ for some sid' and that flag kg is unset. If not, then return $\perp$. Else, ask $\mathcal{F}_{\mathsf{CRS}}$ for SP, compute $(pk, sk) \leftarrow \mathsf{Cred.Kg}(SP)$. Send a message $(\texttt{registerIssuerKey}, pk$ to $\mathcal{F}_{\mathsf{CA}}$. Wait for the confirmation from $\mathcal{F}_{\mathsf{CA}}$, set flag kg, store sk, pk, and forward $(\texttt{verificationKey}, sid, pk)$ to $\mathcal{I}$.
- On input $(\texttt{leakSK}, sid)$ from $\mathcal{I}$ verify that $sid = (\mathcal{I}, sid')$ for some sid'. If not, return $\perp$. Else, send $(\texttt{leakSK}, sid, sk)$ to $\mathcal{I}$.
- On input $(\texttt{issueCred}, sid, qid, X, P, aux(P))$ from $\mathcal{U}$, do the following:
  1. Run $b \leftarrow \mathsf{NymVerify}(SP, P, X, aux(P))$. If $b = 0$ return $\perp$.
  2. Ask $\mathcal{F}_{\mathsf{CA}}$ for the issuer's public key by sending a message $(\texttt{requestIssuerPK}, sid)$.
  3. Compute the issuance request $(\pi_{X,P}, aux(Cred)) \leftarrow \mathsf{Cred.RequestCred}(SP, pk, P, X, aux(P))$ and store $aux(Cred)$.
  4. Send $(\texttt{issueCred}, sid, qid, P, \pi_{X,P})$ to $NET_{\mathcal{I}}$. Upon receiving $(\texttt{issueCred}, sid, qid, P, \pi_{X,P})$, issuer $\mathcal{I}$ outputs $(\texttt{issueCred}, sid, qid, P)$
  5. Upon receiving $(\texttt{credIssued}, sid, qid, P, \vec{a}, Cred)$ from $NET_{\mathcal{I}}$, verify the credential by calling $\mathsf{CheckCred}(SP, pk, X, P, aux(P), Cred, aux(Cred), \vec{a})$. If it returns 1, then $\mathcal{U}$ stores the credential and outputs $(\texttt{credIssued}, sid, qid, \vec{a})$.
- On input $(\texttt{issueCred}, sid, qid, \vec{a})$ from $\mathcal{I}$, verify that $sid = (\mathcal{I}, sid')$ for some sid' and that the flag kg is set. If not, return $\perp$. Else, do the following:
  1. Receive the request for issuance $\pi_{X,P}$ from $NET_{\mathcal{U}}$.
  2. Run the issuance algorithm: If $\mathsf{Cred.IssueCred}(SP, sk, P, \vec{a}, \pi_{X,P}) \to \perp$, then return $\perp$, else send $(\texttt{credIssued}, sid, qid, P, \vec{a}, Cred)$ to $NET_{\mathcal{U}}$.
- On input $(\texttt{proveCred}, sid, X', P', aux(P)', I, \vec{a}', cxt)$ from $\mathcal{U}$, run $\pi_{X,Cred} \leftarrow \mathsf{Cred.Prove}(SP, pk, X, P, aux(P), Cred, aux(Cred), \vec{a}, I, cxt)$. Return $(\texttt{credProved}, sid, \vec{a}_I, \pi_{X,Cred})$ to $\mathcal{U}$.
- On input $(\texttt{verifyCredProof}, sid, pk', \pi_{X,Cred}, \vec{a}'_I, cxt)$ from a party $\mathcal{P}$. Run the proof verification algorithm $result = \mathsf{Cred.Verify}(SP, pk, P', \pi_{X,Cred}, \vec{a}_I, cxt)$ forward its output as $(\texttt{verified}, sid, \vec{a}'_I, result)$ to $\mathcal{P}$.

Interactive proofs of ownership of credentials. We would like to point out that another construction can be built from the alternative construction of URS presented in §C.2 and interactive zero-knowledge proofs as in existing schemes [32, 9]. Briefly, the interactive construction works as follows. The user secret is an exponent. A credential is a Tier 1 signature on a vector of attributes, where the user secret is the fixed first attribute. During the presentation the user derives a Tier 2 signature on the required set of messages (except the secret, of course) and also provides a ZK proof about her secret key.

This construction is more efficient than the non-interactive one because of the efficiency of sigma-protocols, but it requires more rounds of communication and does not provide straight-line extractability, i.e., the extraction protocols require rewinding.

E.2 Proof of Theorem 7

Theorem 6. *Let* URS *be the unlinkable redactable signature scheme according to Definition 1,* SPC *is a structure-preserving commitment scheme,* Gap *be a gap problem,* Π *be a non-interactive proof of knowledge system. Then* $\mathcal{R}_{\mathsf{Cred}}$ *securely realizes* $\mathcal{F}_{\mathsf{Cred}}$ *in the* $(\mathcal{F}_{\mathsf{CRS}}, \mathcal{F}_{\mathsf{CA}})$*-hybrid model if* URS *is correct, unlinkable, unforgeable, and key extractable,* SPC *is binding, the non-interactive proof-of-knowledge system is zero-knowledge and simulation extractable, and the* Gap *problem is hard.*

Proof. First, we provide a description of the simulator in the ideal world. $\mathcal{SIM}$ generates $(SP, td, td_{sim}, td_{ext})$ by running the $\mathsf{Cred.SGenT}(1^\kappa)$ algorithm, stores these values and simulates $\mathcal{F}_{\mathsf{CRS}}$ and $\mathcal{F}_{\mathsf{CA}}$.

If the **issuer is honest**, when receiving $(\texttt{initF}, sid)$ message from $\mathcal{F}_{\mathsf{Cred}}$, $\mathcal{SIM}$ generates issuer keys honestly and returns $(\texttt{initF}, sid, SP, sk, pk, td_{sim}, td_{ext}, \mathsf{Cred.SimProve}, \mathsf{Cred.Verify}, \mathsf{Cred.Extract})$ where it uses URS algorithms inside. It also simulates issuer key registration with $\mathcal{F}_{\mathsf{CA}}$.

If the ***user is also honest***, only the network is controlled by the adversary, so the simulator generates a simulated transcript for the issuance as follows. On seeing the delayed output signal for $(\texttt{issueCred}, sid, qid, P)$, simulate an honest user credential request. $\mathcal{SIM}$ records which X_{Cred}, Y_{Cred} it used in the simulation. If the user's message is delivered (in the real world), let the message through to $\mathcal{I}$. Simulate the honest issuer response. On seeing the delayed output signal for $(\texttt{credIssued}, sid, qid, \vec{a})$ finish the simulation of the issuing protocol, and if the protocol runs to completion, let the messages through to $\mathcal{U}$.

If the ***user is corrupted***, the issuance simulation is done as follows. When seeing on the network the adversary's issuance request to the issuer, $\mathcal{SIM}$ extracts the secret $X, aux(P)$ from the proof $\pi_{X,P}$. If the extraction fails pick any invalid X and $aux(P)$. Send message $(\texttt{issueCred}, sid, qid, X, P, aux(P))$ to $\mathcal{F}_{\mathsf{Cred}}$ on behalf of a dishonest user and short-circuit the delay. Upon receiving $(\texttt{credIssued}, sid, qid, \vec{a})$ as a dishonest user from $\mathcal{F}_{\mathsf{Cred}}$, $\mathcal{SIM}$ computes a credential using the $\mathsf{URS.Sign}$ algorithm and sends the signature to the real dishonest user. Note that $\mathcal{F}_{\mathsf{Cred}}$ does not send the $(\texttt{credIssued}, \dots)$ message if extraction failed because of the $\mathsf{NymVerify}$ check.

Upon a verification request, $\mathsf{Cred.Extract}$ obtains the witness $(X, P, aux(P), aux(P)', X_{Cred}, Y_{Cred})$ from the proof $\pi_{X,\mathsf{Cred}}$ for bookkeeping using the extraction trapdoors and aborts, if the extraction fails.

If the **issuer is corrupted**, on message $(\texttt{registeredIssuerKey}, sid, pk)$ with $sid = (\mathcal{I}, sid')$, $\mathcal{SIM}$ runs $\mathsf{ExtractKey}(pk, td)$ to extract the secret key sk. If extraction fails, it returns $\perp$. Otherwise, $\mathcal{SIM}$ sends $(\texttt{initF}, sid)$ to $\mathcal{F}_{\mathsf{Cred}}$ on behalf of $\mathcal{I}$ and replies with $(\texttt{initF}, sid, SP, sk, pk, td_{sim}, td_{ext}, \mathsf{Cred.SimProve}, \mathsf{Cred.Verify}, \mathsf{Cred.Extract})$ to $\mathcal{F}_{\mathsf{Cred}}$.

If the ***user is honest***, upon receiving the message $(\texttt{issueCred}, sid, qid, P)$ from $\mathcal{F}_{\mathsf{Cred}}$, simulate an honest user credential request with the real adversary controlling the issuer. $\mathcal{SIM}$ records which X_{Cred}, Y_{Cred} it used in the simulation. Wait for the reply from the real dishonest issuer.

If the credential is successfully received, then send $(\texttt{issueCred}, sid, qid, \vec{a})$ to $\mathcal{F}_{\mathsf{Cred}}$ and again short-circuit delays.

The case when the ***user is also corrupted*** is handled internally by the adversary and the simulator does not need to do anything.

We will make steps towards using the ideal functionality of URS. We start with viewing $\mathcal{SIM}$ as using the simulator $\mathcal{SIM}_{\mathsf{URS}}$ internally as a blackbox. Then $\mathcal{SIM}$ generates $(SP, td, td_{sim}, td_{ext})$ by sending $(\texttt{initF}, sid)$ to $\mathcal{SIM}_{\mathsf{URS}}$ to learn $(\texttt{initF}, sid, SP_{\mathsf{URS}}, \mathsf{URS.Kg}, \mathsf{URS.Sign}, \mathsf{URS.Derive}, \mathsf{URS.Verify})$ and generating the remaining values as prescribed by algorithm $\mathsf{Cred.SGenT}(1^{\kappa})$. Similarly, it extracts keys by sending $(\texttt{checkPK}, sid, pk)$ to $\mathcal{SIM}_{\mathsf{URS}}$ to receive the secret key sk in return.

One can also see $\mathsf{Cred.SimProve}$ as running $\mathcal{F}_{\mathsf{URS}}$ and a copy of $\mathcal{SIM}_{\mathsf{URS}}$ with its current state internally. Namely, $\mathsf{Cred.SimProve}$ generates fresh X, P, computes a valid signature σ, and obtains σ_I by first sending $(\texttt{checkPK}, sid, pk_{\mathsf{URS}})$ and then $(\texttt{derive}, sid, pk_{\mathsf{URS}}, I, \vec{a}, \sigma)$ to $\mathcal{F}_{\mathsf{URS}}$.

The proof is done through a sequence of games. In Game 0, $\mathcal{Z}$ interacts with $\mathcal{F}_{\mathsf{Cred}}$ and $\mathcal{SIM}$ as in the ideal world. In Game 15, $\mathcal{Z}$ interacts with $\mathcal{R}_{\mathsf{Cred}}$ (and $\mathcal{A}$, though we will use dummy adversaries) as in the real world. In each game we make a step toward the real world and show that if the environment can distinguish between the current and the previous game, then one can build an adversary that breaks one of the security properties of URS, Π, or solves the gap problem Gap.

We define the success probability of $\mathcal{Z}$ to distinguish between Game i and Game j as $|p_i - p_j|$. We show that $|p_{15} - p_0| \leq p_{urs-corr} + 2p_{spc-bind} + 2p_{urs-unf} + p_{urs-unl} + p_{urs-kext} + p_{nizk-zk} + p_{nizk-se} + p_{gap}$, where $p_{urs-corr}$, $p_{urs-unf}$, $p_{urs-unl}$, and $p_{urs-kext}$ are the probabilities of breaking the correctness, unforgeability, unlinkability, and the key extractability of the URS scheme respectively, $p_{nizk-zk}$ and $p_{nizk-se}$ is the probability of breaking zero-knowledge and simulation extractability of the NIZK scheme respectively, $p_{spc-bind}$ is the probability of breaking the binding property of the commitment scheme, and p_{gap} is the probability of solving a gap problem.

Game 0: *implements the ideal world, where the environment $\mathcal{Z}$ interacts with $\mathcal{F}_{\mathsf{Cred}}$ and $\mathcal{SIM}$ as described above.*

Game 1: *is the same as Game 0 except the simulator $\mathcal{SIM}$ and the functionality $\mathcal{F}_{\mathsf{Cred}}$ are subsumed in a single entity that we denote $\mathcal{C}$, i.e., $\mathcal{C}$ runs $\mathcal{SIM}$ and $\mathcal{F}_{\mathsf{Cred}}$ internally and interacts with the environment on behalf of $\mathcal{SIM}$ and $\mathcal{F}_{\mathsf{Cred}}$. It also maintains the lists $\mathcal{M}_{ISS}$ and $\mathcal{M}_{PRES}$ initialized to $\emptyset$ and flags kg and $keyleak$ that are initially unset.*

The change from Game 0 to Game 1 is purely conceptual and hence $|p_1 - p_0| = 0$.

Game 2: *Here $\mathcal{C}$ behaves as in Game 1 except that we now conceptually split $\mathcal{C}$ into the parts that simulate $\mathcal{F}_{\mathsf{CRS}}$ and $\mathcal{F}_{\mathsf{CA}}$, and the rest. We split the rest into $\mathcal{SIM}_{\mathsf{Cred}}$ that uses $\mathcal{SIM}_{\mathsf{URS}}$ and $\mathcal{F}_{\mathsf{Cred}}$ that uses algorithms, like* $\mathsf{SimProve}$ *that can be seen as using $\mathcal{F}_{\mathsf{URS}}$. We remove the redundant copy of $\mathcal{SIM}_{\mathsf{URS}}$ and connect parts directly.*

State changes to the state of the $\mathcal{SIM}_{\mathsf{Cred}}$ copy of $\mathcal{SIM}_{\mathsf{URS}}$ did not affect the $\mathsf{SimProve}$ copy and vice versa: the $\mathcal{SIM}_{\mathsf{Cred}}$ copy was only used for extraction, which is stateless, and the $\mathsf{SimProve}$ copy only required that $(\texttt{checkPK}, sid, pk_{\mathsf{URS}})$ was sent at least once before deriving signatures. The change from Game 1 to Game 2 are thus purely conceptual and hence $|p_2 - p_1| = 0$.

Game 3: *Modify $\mathcal{F}_{\mathsf{Cred}}$ to pass messages $(\texttt{keygen}, sid)$ and $(\texttt{leakSK}, sid)$ through to $\mathcal{F}_{\mathsf{URS}}$, which in turn interacts with $\mathcal{SIM}_{\mathsf{URS}}$. $\mathcal{SIM}_{\mathsf{URS}}$ observes the key pk_{URS} returned as part of the $(\texttt{verificationKey}, sid, pk_{\mathsf{URS}})$ message and embeds this key into the issuer public key pk.*

This is merely a preparation for later steps to install the right key with the correct $keyleak$ flag in

$\mathcal{F}_{\mathsf{URS}}$ The modification here does not change anything, therefore $|p_3 - p_2| = 0$

Game 4: *When simulating credential issuing, $\mathcal{SIM}_{\mathsf{Cred}}$ also sends $(\mathtt{sign}, sid, (P, Y_{\mathsf{Cred}}, \vec{a}))$ to $\mathcal{F}_{\mathsf{URS}}$ instead of generating σ himself.*

The only difference is that $\mathcal{F}_{\mathsf{URS}}$ verifies signatures after generation. By correctness of the URS scheme, $|p_4 - p_3| \leq p_{urs-corr}$. Again, this is in preparation for later steps to inform $\mathcal{F}_{\mathsf{URS}}$ about honestly signed attributes.

Game 5: *Here $\mathcal{C}$ behaves as in Game 4 except when answering a verification request message $(\mathtt{verifyCredProof}, sid, pk', P', \pi'_{X,Cred}, \vec{a}'_I, cxt')$, we now also send a message $(\mathtt{verify}, sid, pk_{\mathsf{URS}}, \sigma_{\mathsf{URS}}, (P^*, Y^*_{\mathsf{Cred}}, \vec{a})_I)$ to $\mathcal{F}_{\mathsf{URS}}$ with the $\sigma_{\mathsf{URS}}, P^*, Y^*_{Cred}$ extracted by $\Pi.\mathsf{Extract}$ in $\mathsf{Cred}.\mathsf{Extract}$.*

This can only lead to additional signature rejections because of the additional checks in $\mathcal{F}_{\mathsf{URS}}$. Let E be the event in Game 5 of $(P, Y_{\mathsf{Cred}}, \vec{a})_I$ being rejected by $\mathcal{F}_{\mathsf{URS}}$ because no $(P, Y_{\mathsf{Cred}}, \vec{a}')$ such that $(P, Y_{\mathsf{Cred}}, \vec{a}')_I = (P, Y_{\mathsf{Cred}}, \vec{a})_I$ was sent to $\mathcal{F}_{\mathsf{URS}}$ for signing. We show a reduction that uses an environment that triggers E, to break the unforgeability of URS. We simply simulate the honest issuer using signing queries, and return $\sigma_{\mathsf{URS}}, (P, Y_{\mathsf{Cred}}, \vec{a})_I)$ as a forgery. Unless E happens, all checks are already made by $\mathcal{F}_{\mathsf{Cred}}$ in $(\mathtt{verifyCredProof}, \ldots)$, hence $|p_5 - p_4| = p_{urs-unf}$.

Game 6: *Here $\mathcal{C}$ behaves as in Game 5 except that if the issuer is honest, it corrects verification to false if X^* extracted from the proof in the verification request $(\mathtt{verifyCredProof}, sid, pk', P', \pi'_{X,Cred}, \vec{a}'_I, cxt')$ is not contained in a record $(ISS, \mathcal{U}, X^*, *)$ for a dishonest user.*

All corrections are implied by the check in step 6. of $\mathcal{F}_{\mathsf{Cred}}$. Hence, $|p_6 - p_5| = 0$.

Note that in this case, either the extracted signature σ_{URS} is on a fresh message, it opens an existing pseudonym of a dishonest user to a different X^*, or it signs some P^* and the Y^*_{Cred} of an honest user, for which we also extract a corresponding X^*_{Cred}.

Game 7: *This is the same as Game 6, except that we replace the checks of step 6 for message $(\mathtt{verifyCredProof}, \ldots)$ in F_{Cred} with a check that we do not extract different $X^*, aux(P)^*$ for the same P during verification than provided at issuing.*

Let E be the event that we extract different X^*,$aux(P)^*$ for the same P^* during verification and issuing. We show a reduction that uses an environment that triggers E, to break the binding property of $\mathsf{NymVerify}$. We simply simulate the game using the commitment parameters, and return $(X, aux(P))$, $(X^*, aux(P)^*)$ to break binding. Unless E happens, these checks are implied by the checks introduced in Game 6 and 5 and thus no longer needed. Hence, $|p_7 - p_6| \leq p_{spc-bind}$.

Game 8: *This is the same as Game 7, except that we remove the check introduced in Game 6.*

Note that checks corresponding to fresh messages is already enforced by the checks in $\mathcal{F}_{\mathsf{URS}}$ and checks corresponding different openings of pseudonyms are enforced by the new checks in Game 6.

We reduce an environment that triggers the case of an honest P^* and Y^*_{Cred} to an adversary against the gap problem for X^*_{Cred} and Y^*_{Cred}, thus $|p_8 - p_7| \leq p_{gap}$.

Game 9: *is the same as Game 8, except that the calls to $\mathcal{F}_{\mathsf{URS}}$ are replaced with calls to $\mathcal{R}_{\mathsf{URS}}$. This removes the checks introduced in Game 4*

Since $\mathcal{R}_{\mathsf{URS}}$ securely realizes $\mathcal{F}_{\mathsf{URS}}$ with simulator $\mathcal{SIM}_{\mathsf{URS}}$ in case URS is correct, unlinkable, unforgeable and key extractable, we have that $|p_9 - p_8| \leq p_{urs-unf} + p_{urs-unl} + p_{urs-kext}$.

Game 10: *Here $\mathcal{C}$ behaves as in Game 9 except that we no longer perform the check for duplicate pseudonym openings.*

We know that $|p_{10} - p_9| \leq p_{spc-bind}$.

Game 11: *Here $\mathcal{C}$ behaves as in Game 10 except that we no longer send a message $(\mathtt{verify}, \ldots)$ to $\mathcal{R}_{\mathsf{URS}}$ with the signature extracted by $\Pi.\mathsf{Extract}$ in $\mathsf{Cred}.\mathsf{Extract}$.*

As the extracted witness is correct we know that $|p_{11} - p_{10}| = 0$. Note that we now no longer make use of extracted values except for checking that the witnesses are correct.

Game 12: *is the same as Game 11, except that no values are extracted from the users' requests.*

Note that in $(\texttt{verifyCredProof}, sid, \dots)$ we only have to extract if there is no record $(PRES, \mathcal{U}, cxt', X^*, P', aux(P)', \vec{a}'_I, \pi'_{X,Cred})$, that is for proofs that are not simulated. The advantage of the adversary in Game 12 is bounded by the simulation extractability of the non-interactive proof system, therefore $|p_{11} - p_{10}| \leq p_{nizk-se}$.

Game 13: *is the same as Game 12, except that calls to the realization $\mathcal{R}_{\mathsf{URS}}$ are replaced with calls to URS^+ algorithms.*

Due to the fact that $\mathcal{R}_{\mathsf{URS}}$ is realized in such a way that on any incoming message the real algorithm are invoked, this change is purely conceptual, $|p_{13} - p_{12}| = 0$. Note that we always run concrete algorithms with their inputs readily available.

Game 14: *is the same as Game 13, except that we generate proofs using* Prove *instead of* $\mathsf{SimProve}$.

The advantage of the adversary in Game 14 is bounded by the zero-knowledge property of the non-interactive proof system, therefore $|p_{14} - p_{13}| \leq p_{nizk-zk}$.

Game 15: *is the realization of the real world where $\mathcal{C}$ is replaced by $\mathcal{A}$, $\mathcal{F}_{\mathsf{CRS}}$, $\mathcal{F}_{\mathsf{CA}}$, and $\mathcal{R}_{\mathsf{Cred}}$.*

One can see that Game 15 is the same as Game 14, since already in Game 14 $\mathcal{C}$ runs $\mathcal{A}$, $\mathcal{F}_{\mathsf{CRS}}$, $\mathcal{F}_{\mathsf{CA}}$, and $\mathcal{R}_{\mathsf{Cred}}$ internally and interacts with the environment on behalf of $\mathcal{A}$ and $\mathcal{R}_{\mathsf{Cred}}$. So the change from Game 15 to Game 14 is purely conceptual and hence $|p_{15} - p_{14}| = 0$.

Summing up the probabilities, we have that $|p_{15} - p_0| \leq p_{urs-corr} + 2p_{spc-bind} + 2p_{urs-unf} + p_{urs-unl} + p_{urs-kext} + p_{nizk-zk} + p_{nizk-se} + p_{gap}$. $\square$

F Generic Group Proof of J-RootDH Assumption

Definition 18 (J-RootDH Assumption)**.** *Let J be a subset of $[1..n]$, let $\alpha, r \in \mathbb{Z}_p^*$, and $X = (G^{\alpha \cdot \prod_{i=1}^{n}(\alpha - i)})^r$, $Y = (G^{\prod_{i \in J}(\alpha - i)})^r$. For all PPT algorithms $\mathcal{A}$, the probability $Pr[\mathcal{A}(G, \tilde{G}, \{G^{\alpha^i}, \tilde{G}^{\alpha^i}\}_{i=1}^{n+1}, X, Y) = J', Z]$ such that $Z = (G^{\prod_{i \in J'}(\alpha - i)})^r$ and $J' \subseteq [1..n]$ and $J' \neq J$, is at most a negligible function $\nu(\kappa)$.*

We show that the assumption holds in the generic group model. In the generic group model, elements of the bilinear groups $\mathbb{G}, \tilde{\mathbb{G}}$, and $\mathbb{G}_t$ are encoded as unique random strings. Thus, the adversary cannot directly test any property other than equality. Oracles are assumed to perform operations between group elements, such as performing the group operations in $\mathbb{G}, \tilde{\mathbb{G}}$, and $\mathbb{G}_t$. The opaque encoding of the elements of $\mathbb{G}$ is defined as the function $\xi_1 : \mathbb{Z}_p \to \{0,1\}^*$, which maps all $a \in \mathbb{Z}_p$ to the string representation $\xi_1(a)$ of $G^a \in \mathbb{G}$. Similarly, we have $\xi_2 : \mathbb{Z}_p \to \{0,1\}^*$ for $\tilde{\mathbb{G}}$ and $\xi_T : \mathbb{Z}_p \to \{0,1\}^*$ for $\mathbb{G}_t$. The adversary $\mathcal{A}$ communicates with the oracles using the ξ-representations of the group elements only.

Theorem 14. *Let $\mathcal{A}$ be an algorithm that solves the J-RootDH assumption in the generic group model. Let n_G be the number of queries $\mathcal{A}$ makes to the oracles computing the group action and pairing. If ξ_1, ξ_2, and ξ_T are chosen at random, then the probability ϵ that $\mathcal{A}(p, \xi_1(1), \xi_2(1), \xi_1(\alpha^i), \xi_2(\alpha^i), i = 1, \dots n+1, \xi_1(\alpha \cdot \prod_{i=1}^{n}(\alpha - i)r), \xi_1(\prod_{i \in J}(\alpha - i)r))$ outputs $\xi_1(\prod_{i \in J'}(\alpha - i)r)$ for some $J' \neq J$, is bounded by*

$$\epsilon \leq \frac{(n_G + 2n + 6)^2 n}{p}.$$

Proof. Consider an algorithm $\mathcal{B}$ that interacts with $\mathcal{A}$ in the following game. Let $F_{1,i}, F_{2,i}, F_{T,i}$ be multivariate polynomials in $\mathbb{Z}_p[\alpha, r]$. $\mathcal{B}$ maintains three lists of pairs $L_1 = \{(F_{1,i}, \xi_{1,i}) : i = 0, \dots, \tau_1 - 1\}, L_2 = \{(F_{2,i}, \xi_{2,i}) : i = 0, \dots, \tau_2 - 1\}, L_T = \{(F_{T,i}, \xi_{T,i}) : i = 0, \dots, \tau_T - 1\}$, such that, at step τ in the game $\tau_1 + \tau_2 + \tau_T \leq \tau + 2n + 6$.

Initially, $\tau = 0, \tau_1 = n + 4, \tau_2 = n + 2, \tau_T = 0$ and the polynomials are: $F_{1,0} = 1; F_{1,i} = \alpha^i, i = 1, \dots n+1; F_{1,n+2} = \alpha \cdot \prod_{i=1}^{n}(\alpha - i)r; F_{1,n+3} = \prod_{i \in J}(\alpha - i)r$. $F_{2,0} =$

1; $F_{2,i} = \alpha^i, i = 1, \ldots n+1$. $\mathcal{B}$ begins the game with $\mathcal{A}$ by providing it with the random strings $\xi_{1,0}, \ldots \xi_{1,n+3}, \xi_{2,0}, \ldots \xi_{2,n+1}$. We describe the oracles $\mathcal{A}$ may query:

Group action: $\mathcal{A}$ inputs two group elements $\xi_{1,i}$ and $\xi_{1,j}$, where $0 \leq i, j < \tau_1$, and a request to multiply/divide. $\mathcal{B}$ sets $F_{1,\tau_1} \leftarrow F_{1,i} \pm F_{1,j}$. If $F_{1,\tau_1} = F_{1,u}$ for some $u \in \{0, \ldots, \tau_1 - 1\}$, then $\mathcal{B}$ sets $\xi_{1,\tau_1} = \xi_{1,u}$; otherwise, it sets ξ_{1,τ_1} to a random string in $\{0,1\}^* \backslash \{\xi_{1,0}, \ldots, \xi_{1,\tau_1-1}\}$. Finally, $\mathcal{B}$ returns ξ_{1,τ_1} to $\mathcal{A}$, adds $(F_{1,\tau_1}, \xi_{1,\tau_1})$ to L_1, and increments τ_1. Group actions for $\mathbb{G}$ and $\mathbb{G}_t$ are handled the same way.

Pairing: $\mathcal{A}$ inputs two group elements $\xi_{1,i}$ and $\xi_{2,j}$, where $0 \leq i < \tau_1, 0 \leq j < \tau_2$. $\mathcal{B}$ sets $F_{T,\tau_T} \leftarrow F_{1,i} \cdot F_{2,j}$. If $F_{T,\tau_T} = F_{T,u}$ for some $u \in \{0, \ldots, \tau_T - 1\}$, then $\mathcal{B}$ sets $\xi_{T,\tau_T} = \xi_{T,u}$; otherwise, it sets ξ_{T,τ_T} to a random string in $\{0,1\}^* \backslash \{\xi_{T,0}, \ldots, \xi_{T,\tau_T-1}\}$. Finally, $\mathcal{B}$ returns ξ_{T,τ_T} to $\mathcal{A}$, adds $(F_{T,\tau_T}, \xi_{T,\tau_T})$ to L_T, and increments τ_T.

We assume SXDH holds in $(\mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_t)$ and therefore no isomorphism oracles exist. Eventually, $\mathcal{A}$ stops and outputs an element $\xi_{1,e}$, where $0 \leq e \leq \tau_1$.

We argue that it is impossible for $\mathcal{A}$'s output to always be correct. The output polynomial must be a linear combination of the polynomials corresponding to the elements available to $\mathcal{A}$ in the respective group. Consider the output polynomial $F_{1,e}$:

$$F_{1,e} = e_0 + \sum_{i=1}^{n+1} e_{1,i}\alpha^i + e_{1,n+2}\alpha \cdot \prod_{i=1}^{n}(\alpha - i)r + e_{1,n+3}\prod_{i\in J}(\alpha - i)r,$$

We denote as $\overline{J'}$ the set that is complimentary to J'. The conditions for the adversary's forgery being successful is

$$F_{1,e} \cdot \alpha \cdot \prod_{i\in\overline{J'}}(\alpha - i) = F_{1,0} \cdot F_{1,n+2},\ J \neq J'. \tag{1}$$

Substituting $F_{1,e}$, $F_{1,0}$, and $F_{1,n+2}$ in equation (1) by their polynomials we have

$$\begin{aligned}\big(e_0 + \sum_{i=1}^{n+1} e_{1,i}\alpha^i + e_{1,n+2}\alpha \cdot \prod_{i=1}^{n}(\alpha - i)r + e_{1,n+3}\prod_{i\in J}(\alpha - i)r\big)\cdot \\ \alpha \cdot \prod_{i\in\overline{J'}}(\alpha - i) = \alpha \cdot \prod_{i=1}^{n}(\alpha - i)r\end{aligned} \tag{2}$$

which can be further expanded and simplified to

$$\begin{aligned}e_0 \cdot \alpha \prod_{i\in\overline{J'}}(\alpha - i) + \sum_{i=1}^{n+1} e_{1,i} \cdot \alpha^{i+1}\prod_{i\in\overline{J'}}(\alpha - i) + e_{1,n+2} \cdot \alpha^2 \cdot \prod_{i=1}^{n}(\alpha - i)r \prod_{i\in\overline{J'}}(\alpha - i) + \\ + e_{1,n+3} \cdot \alpha \prod_{i\in J}(\alpha - i)r \prod_{i\in\overline{J'}}(\alpha - i) - \alpha \cdot \prod_{i=1}^{n}(\alpha - i)r = 0\ .\end{aligned}$$

Since polynomials $e_0 \cdot \alpha \prod_{i\in\overline{J'}}(\alpha - i)$ and $e_{1,i} \cdot \alpha^{i+1}\prod_{i\in\overline{J'}}(\alpha - i)$ do not contain variable r, and polynomial $e_{1,n+2} \cdot \alpha^2 \cdot \prod_{i=1}^{n}(\alpha - i)r \prod_{i\in\overline{J'}}(\alpha - i)$ has a degree that is different from the one of $\alpha \cdot \prod_{i=1}^{n}(\alpha - i)r$, equation (2) can always hold only in case of $e_0 = e_{1,i} = e_{1,n+2} = 0$.

But since $F_{1,e} \neq 0$ and $e_0 = e_{1,i} = e_{1,n+2} = 0$ we have that $e_{1,n+3} \neq 0$ and

$$e_{1,n+3} \cdot \alpha \prod_{i\in J}(\alpha - i)r \prod_{i\in\overline{J'}}(\alpha - i) - \alpha \cdot \prod_{i=1}^{n}(\alpha - i)r = 0,\ J \neq J'. \tag{3}$$

We now discuss when equation (3) always holds. We only need to consider the case when the degrees of the polynomials are equal (i.e. when $|J| = |J'|$, since in this case $|J| + |\bar{J}'| = n$).

$$e_{1,n+3} = \frac{\prod_{i=1}^{n}(\alpha - i)}{\prod_{i\in J}(\alpha - i)\prod_{i\in \overline{J'}}(\alpha - i)},\; J \neq J', |J| = |J'|. \tag{4}$$

$$e_{1,n+3} = \frac{\prod_{i\in \overline{J}}(\alpha - i)}{\prod_{i\in \overline{J'}}(\alpha - i)} = \frac{\prod_{i\in J'\setminus J}(\alpha - i)}{\prod_{i\in J\setminus J'}(\alpha - i)}. \tag{5}$$

$$e_{1,n+3} \cdot \prod_{i\in J\setminus J'}(\alpha - i) = \prod_{i\in J'\setminus J}(\alpha - i). \tag{6}$$

Since $J \neq J'$ the polynomials on the left and the right sides have different roots, equation (6) cannot always hold.

Simulation analysis: $\mathcal{B}$ chooses random values to instantiate the variables α and r. The chance of choosing a random assignment that hits the root of any given polynomial is bounded from above by the Schwartz-Zippel theorem by the degree of the polynomial divided by p. The success probability of $\mathcal{A}$ is bounded by the probability that any of the following holds:

1. $F_{1,i} - F_{1,j} = 0$ for some i, j such that $F_{1,i} \neq F_{1,j}$,
2. $F_{2,i} - F_{2,j} = 0$ for some i, j such that $F_{2,i} \neq F_{2,j}$,
3. $F_{T,i} - F_{T,j} = 0$ for some i, j such that $F_{T,i} \neq F_{T,j}$,
4. $F_{1,e} \cdot \alpha \cdot \prod_{i\in \overline{J'}}(\alpha - i) = F_{1,0} \cdot F_{1,n+2}$.

The maximum total degree of the polynomials are:

1. $n + 2$,
2. $n + 2$,
3. $2(n + 2)$,
4. $n + 2$.

Summing up over all pairs (i, j) in each case, we see that the probability that a collision causes $\mathcal{B}$'s simulation to fail $\epsilon \leq \binom{\tau_1}{2}\frac{n+2}{p} + \binom{\tau_2}{2}\frac{n+2}{p} + \binom{\tau_T}{2}\frac{2n+4}{p} + \frac{n+2}{p}$. Recalling that $\tau_1 + \tau_2 + \tau_T \leq n_G + 2n + 6$, results into the following overall probability: $\epsilon \leq (n_G + 2n + 6)^2(n/p)$.

□