Composable & Modular Anonymous Credentials: Definitions and Practical Constructions

Jan Camenisch[1], Maria Dubovitskaya[1], Kristiyan Haralambiev[2], and Markulf Kohlweiss[3]

[1] IBM Research – Zurich
[2] Google Inc.
[3] 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 definitional approach to be of independent interest.

Keywords: 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 [6, 45], group signatures [16, 57], direct anonymous attestation [21], electronic cash [43], voting schemes [52], adaptive oblivious transfer [33, 24], and anonymous credentials [13].

For protocols constructed like this to be efficient, special properties are demanded from 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 in a higher-level protocol it must typically be revealed so that a third party can be convinced of its validity. Thus, a signature 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 [26].

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, the q-SDH, or the LRSW assumption, respectively, and allow for an alternative approach when using a signature in a protocol: instead of revealing the signature to a party, the user employs zero-knowledge proofs to convince the party that she possesses a valid signature. While in theory this is possible for any signature scheme, CL-signatures were the first that enabled efficient proofs using generalized Schnorr proofs of knowledge. This is due to the algebraic properties of CL-signatures, i.e., no hash function is applied to the message and the signature and message values are either exponents or group elements.

Since the introduction 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), one needs to resort to the Fiat-Shamir heuristic, i.e., to the random oracle model, and thus looses all provable security guarantees when the oracle is instantiated by a hash function [37].
2. *Straight-line extraction.* When designing a protocol to be secure in the UC model [36], rewinding can not 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, this is rather expensive [27] and may render the overall protocol impractical.
3. *Linear size.* When proving ownership of a CL-signature on many messages, all of them are needed for the verification of the signature and therefore a 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 [49], 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 [2, 3, 1]. 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 (cf. drawback (3)), which is a typical requirement in applications such as anonymous credentials. Indeed, as for CL-signatures, the size of proofs with structure-preserving signatures grows linearly with the num-

ber 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 when a single message is signed and becomes superior when the number of messages signed 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. Comparing the two definitions we find the seemingly common phenomenon that the functionality-based definition requires a key-registration process (allowing for the extraction of keys in the proof) while the property based definition per se does not require that. 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 [56, 61, 38], structure-preserving signatures [2, 3], and (a minimal dose of) non-interactive proofs of knowledge (NIPoK), which in practice can be instantiated by *witness-indistinguishable* Groth-Sahai proofs [49]. As we are interested in practical efficiency, we instantiate our scheme with concrete building block that deliberately rely on stronger assumptions (see §4.3). However, if one is willing to accept a less efficient scheme, a CDH-based vector commitment scheme [38] secure under less strong assumptions. We show how to make use of algebraic properties in our building blocks to minimize the witness size of the NIPoK.

Third, in §4, to demonstrate the versatility of our URS scheme as a CL-signature scheme '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, a full comparison is deferred to the full paper [10]. As there are a multitude of papers on redactable, quotable, and sanitizable signatures [50, 22, 63, 7], we focus on the most influential definitional work and the most promising approaches in terms of efficiency.

A variety of signature schemes with flexible signing capabilities and strong privacy properties have been proposed [41, 18, 8, 7, 11, 15, 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 [56, 60] achieve better efficiency but only consider one-time-show credentials, and while the scheme in [56] is not defined formally, the scheme in [60] involves interaction.

The first efficient multi-show anonymous credential scheme is [30]. It was extended with efficient attribute disclosure [25] 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 [13, 14]. An instantiation of our URS scheme, however, is almost twice more efficient than [13] despite the fact that the latter does not support signing multiple messages. Belenkiy et al. [12] show how to use the randomizability of P-signatures for delegation and Chow et al. [44] extend the randomizable group signatures scheme underlying [12] with a flexible attribute mechanism. Izabachène et al. [55] extends the work of [13] with vector commitments; their scheme is, however, not secure under our definitions. In independent work, Hanser and Slamanig [51] 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 [47]. Furthermore, it uses hash function to encode attributes and thus does not enable efficient protocol design. None of these schemes is (universally) composable. Camenisch et al. [28] have recently proposed property-based definitions of anonymous credentials and of the necessary building blocks, given a construction and proved it secure. Their definitions turn out to be rather complex, indicating that for complex systems functionality-based definitions might be easier to handle. Indeed, for their definition of privacy, Camenisch et al. make use of what they call 'filter' which is very reminiscent of an ideal functionality. Finally, the construction they provide is based on CL-signatures and thus suffers from the drawbacks of that approach.

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 $\boldsymbol{m} = (1, m_1, \ldots, m_n)$ and a quote as $\boldsymbol{m}_I = (2, m'_1, \ldots, m'_n)$. We represent each valid quoting transformation by a set $I \subseteq [1, n]$ of message positions and denote quoting either by $I(\boldsymbol{m})$ or $\boldsymbol{m}_I$. We denote the i$^{\text{th}}$ message element either by $\boldsymbol{m}[i]$ or m_i. A quote $\boldsymbol{m}_I$ from $\boldsymbol{m}$ is of the form

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

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 $\boldsymbol{m}$ and $\boldsymbol{m}_I$ as Tier 1 and Tier 2 vectors respectively. The vector $\boldsymbol{m}_I$ can be sparse and can have a much shorter encoding than $\boldsymbol{m}$. Finally, we define $\mathsf{Zero}(\boldsymbol{m}, I) = (1, \tilde{m}_1, \ldots, \tilde{m}_n)$, 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.

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:*

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

$\mathsf{URS.Kg}(SP) \to (pk, sk)$. 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* $\mathcal{M}$*.*

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

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

$\mathsf{URS.Verify}(pk, \sigma, \boldsymbol{m}) \to 0/1$. Verify *takes the verification key* pk*, a signature* σ*, and a message* $\boldsymbol{m}$ *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 *output unique handles, for instance implemented using a counter. 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.* $(\boldsymbol{m}_I^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}(\cdot), \mathcal{O}_{\mathsf{Derive}}(\cdot,\cdot), \mathcal{O}_{\mathsf{Reveal}}(\cdot)}(pk)$*, where* $\mathcal{O}_{\mathsf{Sign}}$*,* $\mathcal{O}_{\mathsf{Derive}}$*, and* $\mathcal{O}_{\mathsf{Reveal}}$ *behave as follows:*

$\underline{\mathcal{O}_{\mathsf{Sign}}(\boldsymbol{m})}$	$\underline{\mathcal{O}_{\mathsf{Derive}}(h, I)}$	$\underline{\mathcal{O}_{\mathsf{Reveal}}(h)}$
$h \leftarrow \mathsf{H}$*;* $\sigma \leftarrow \mathsf{Sign}(sk, \boldsymbol{m})$	*if* $(h, \boldsymbol{m}, \sigma) \in Q_1$	*if* $(h, \boldsymbol{m}, \sigma) \in Q_1$
add $(h, \boldsymbol{m}, \sigma)$ *to* Q_1	$\sigma' \leftarrow \mathsf{Derive}(pk, I, \boldsymbol{m}, \sigma)$	*add* $\boldsymbol{m}$ *to* Q_3
return h	*add* $\boldsymbol{m}_I$ *to* Q_2*; return* σ'	*return* σ

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* $\mathsf{Verify}(pk, \sigma^*, \boldsymbol{m}_I^*) = 1$ *and* $\forall \boldsymbol{m} \in Q_3,\ \boldsymbol{m}_I^* \neq \boldsymbol{m}_I$*, and* $\boldsymbol{m}_I^* \notin Q_2$ *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 $\boldsymbol{m}$ without calling $\mathsf{Sign}(\boldsymbol{m})$ 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.* $(pk, I, \boldsymbol{m}^{(0)}, \boldsymbol{m}^{(1)}, \sigma^{(0)}, \sigma^{(1)}) \xleftarrow{\$} \mathcal{A}(SP)$*, where* $\boldsymbol{m}_I^{(0)} = \boldsymbol{m}_I^{(1)}$, $\mathsf{Verify}(pk, \sigma^{(0)}, \boldsymbol{m}^{(0)}) = 1$*, and* $\mathsf{Verify}(pk, \sigma^{(1)}, \boldsymbol{m}^{(1)}) = 1$.
- *Step 3. Pick* $b \leftarrow \{0, 1\}$ *and form* $\sigma_I^{(b)} \xleftarrow{\$} \mathsf{Derive}(pk, I, \boldsymbol{m}^{(b)}, \sigma^{(b)})$.
- *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 $(\texttt{keygen}, sid)$ from S, verify that $sid = (S, sid')$ for some sid' and that flag kg is unset. If not, then return $\perp$. Else, send $(\texttt{initF}, sid)$ to $\mathcal{SIM}$ and wait for a message $(\texttt{initF}, sid, SP, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Derive}, \mathsf{Verify})$ 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) in $\mathcal{K}$, and return $(\texttt{verificationKey}, sid, pk)$ to S.
- On input $(\texttt{checkPK}, sid, pk')$ 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 $(\texttt{checkedPK}, sid, true)$. Else, if $(pk', \perp)$ was stored in $\mathcal{K}$ return $(\texttt{checkedPK}, sid, false)$. Else, send $(\texttt{checkPK}, sid, pk')$ to $\mathcal{SIM}$, wait for $(\texttt{checkedPK}, sid, sk')$ from $\mathcal{SIM}$, add (pk', sk') to $\mathcal{K}$. If $sk' \neq \perp$, return $(\texttt{checkedPK}, sid, \mathsf{true})$ to P. Otherwise, return $(\texttt{checkedPK}, sid, \mathsf{false})$ to P.
- On input $(\texttt{leakSK}, sid)$ 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 $(\texttt{leakSK}, sid, sk)$, otherwise - abort.
- On input $(\texttt{sign}, sid, \boldsymbol{m})$ from S, verify that $sid = (S, sid')$ for some sid' and that the flag kg is set. If not, return $\perp$. Else, run $\sigma \leftarrow \mathsf{Sign}(sk, \boldsymbol{m})$ and $\mathsf{Verify}(pk, \sigma, \boldsymbol{m})$. If Verify is successful, return $(\texttt{signature}, sid, \boldsymbol{m}, \sigma)$ to S and add $\boldsymbol{m}$ to $\mathcal{Q}$, otherwise return $\perp$.
- On input $(\texttt{derive}, sid, pk', I, \boldsymbol{m}, \sigma)$ from some party P, run $\mathsf{Derive}(pk', I, \boldsymbol{m}, \sigma)$ 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 $\sigma' \leftarrow \mathsf{Sign}(sk_{tmp}, \mathsf{Zero}(\boldsymbol{m}, I))$ and return $\mathsf{Derive}(pk', I, \mathsf{Zero}(\boldsymbol{m}, I), \sigma')$. Otherwise, return the output of $\mathsf{Derive}(pk', I, \boldsymbol{m}, \sigma)$. (Continue on the next page.)

– On input $(\texttt{verify}, sid, pk', \sigma, \boldsymbol{m}_I)$ from some party P, compute $result \leftarrow \mathsf{Verify}(pk', \sigma, \boldsymbol{m}_I)$. If the flag kg is set, $pk' = pk$, flag $keyleak$ is not set, and $\nexists\, \boldsymbol{m} \in \mathcal{Q}$ such that $\boldsymbol{m}_I = I(\boldsymbol{m})$, then output $(\texttt{verified}, sid, \boldsymbol{m}_I, 0)$. Otherwise, output $(\texttt{verified}, sid, \boldsymbol{m}_I, result)$.

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, 45, 54, 58]. 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 $\mathcal{F}_{\mathsf{URS}}$ and 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 $\perp$.

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 the full version [10]), 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.

$\mathsf{URS.CheckPK}(pk) \to 0/1$. 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.*

$\mathsf{URS.SGenT}(1^\kappa) \to (SP, td)$. 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.*

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

$\mathsf{URS.CheckKeys}(pk, sk) \to 0/1$. 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 $\mathsf{CheckKeys}(pk, sk)$ holds can be used to create signatures that will verify. It moreover guarantees that $\mathsf{CheckPK}(pk)$ holds for honestly generated public keys.

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

Definition 4 (Parameter Indistinguishability). *A redactable signature scheme* $\mathsf{URS.\{SGen, Kg, Sign, Derive, Verify\}}$ *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 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 5 (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 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 the full version of this paper [10]).

Theorem 1. *Let* URS *be an unlinkable redactable signature scheme. If* URS *is correct, parameter indistinguishable, key-extractable, unforgeable, and composable unlinkable 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 susceptibly 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 $\boldsymbol{m}$, first, commit to $\boldsymbol{m}$ 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 $\boldsymbol{m}_I$. We verify a signature on a quote by verifying both the structure-preserving signature (SPS.Verify) and checking the opening of the commitment (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, [51] 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 the full version [10]).

Before describing our redactable signature scheme in more detail, we present a vector commitment scheme that uses a variant of polynomial commitments from [56]. While our changes are partly cosmetic, they simplify the assumption needed for opening non-malleability.

3.1 Vector Commitments Simplified

A vector of messages $\boldsymbol{m} \in \mathbb{Z}_p^n$ 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 $\boldsymbol{m}$ for the vectors being committed and Tier 2 vectors $\boldsymbol{m}_I$ 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.

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

$\mathsf{VC.Open}(pp, I, \boldsymbol{m}, r)$. 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.

$\mathsf{VC.Check}(pp, C, \boldsymbol{m}_I, W)$. 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 [56], 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 6 (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.* $C, \boldsymbol{m}_I, W, \boldsymbol{m}'_{I'}, W' \leftarrow \mathcal{A}(pp)$

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* $\mathcal{G}, \mathsf{Setup}$, *and* $\mathcal{A}$*) that* $1 = \mathsf{VC.Check}(pp, C, \boldsymbol{m}_I, W) = \mathsf{VC.Check}(pp, C, \boldsymbol{m}'_{I'}, W')$ *and that there exist* $i \in I \cap I'$ *such that* $m_i \neq m'_i$ *is at most* $\nu(\kappa)$*. (Note that* $\boldsymbol{m}_I$ *and* $\boldsymbol{m}'_{I'}$ *are Tier 2 vectors, and thus encode the sets* I *and* I' *respectively.)*

Definition 7 (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.* $\boldsymbol{m}, I \leftarrow \mathcal{A}(pp)$
- *Step 3. Pick random* r*, compute* $C \leftarrow \mathsf{VC.Commit}(pp, \boldsymbol{m}, r)$*, and* $W \leftarrow \mathsf{VC.Open}(pp, I, \boldsymbol{m}, r)$*.*
- *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 $1 = \mathsf{VC.Check}(pp, C, \boldsymbol{m}_{I'}, W')$, and $I \neq I'$ is at most $\nu(\kappa)$.*

In the following theorems we make use of the n-BSDH assumption [48] and the n-RootDH assumption that are defined next. See the full version of this paper [10] 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 8 (n-SDH Assumption). *The n-strong Diffie-Hellman (n-SDH) assumption [17] states that there exists a $\mathcal{G}$ that for all algorithms $\mathcal{A}$, the following advantage*

$$\mathbf{Adv}^{\mathrm{nSDH}}_{\mathcal{G}}(\lambda) = \mathsf{Pr}\big[(p, e, \mathbb{G}, G) \xleftarrow{\$} \mathcal{G}\ ;\ x, c \xleftarrow{\$} \mathcal{Z}_p\ ;\ \mathcal{A}(1^\lambda, p, \mathbb{G}, G, G^x, \ldots, G^{x^n}) = (c, G^{1/(x+c)})\big] \leq \mathsf{negl}(\lambda).$$

The n-BSDH assumption is defined identically to n-SDH except that now $\mathcal{A}$ is challenged to compute $(c, e(G, \tilde{G}^{1/(x+c)})$. Note that the n-BSDH assumption is already implied by the n-SDH assumption.

Definition 9 (n-RootDH Assumption).
For an adversary $\mathcal{A}$ consider the following game:

- *Step 1. $grp \leftarrow \mathcal{G}(1^\kappa)$*
- *Step 2. Pick random $\alpha, r \leftarrow \mathbb{Z}_p^*$, compute $X = (G^{\alpha \cdot \prod_{i=1}^{n}(\alpha - i)})^r$.*
- *Step 3. $(J, state) \leftarrow \mathcal{A}(G, \tilde{G}, \{G^{\alpha^i}, \tilde{G}^{\alpha^i}\}_{i=1}^{n+1}, X)$*
- *Step 4. Compute $Y = (G^{\prod_{i \in J}(\alpha - i)})^r$.*
- *Step 5. $J', Z \leftarrow \mathcal{A}(state, Y)$*

Then the group generator $\mathcal{G}$ satisfies the n-RootDH assumption if for all such PPT algorithms $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that the probability (over the choices of $\mathcal{G}$, α, r, and $\mathcal{A}$ that J and J' are subsets of $[1..n]$, $J' \neq J$, and $Z = (G^{\prod_{i \in J'}(\alpha - i)})^r$ 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 [56] and can be found in the full version [10].

Theorem 3. *The commitment scheme VC defined above is opening non-malleable under the n-RootDH assumption.* The proofs can be found in the full version [10].

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

Let R be an efficiently computable binary relation. For pairs $(W, Stmt) \in R$ 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:

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

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

$\Pi.\mathsf{Verify}(CRS, \pi, Stmt) \to 0/1$. 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 (f-extractable [13]) witnesses that can be (only partially) 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 (K), and for the f-extractable witnesses we use "existence" ($\exists$) notation. (If function f is constant, nothing can be extracted.) We define $\mathfrak{K}$ as a set of extractable witnesses and $\mathfrak{E}$ as a set of the 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. The following is an examplary statement:

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

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.

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

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

$\Pi.\mathsf{Extract}(CRS, td_{\mathsf{ext}}, \pi, Stmt) \to W$. 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, Extract, SimProve, and Verify algorithms – a message (label) L, which is non-malleably attached to the proof (i.e. the signature of knowledge is computed on this message). We provide a formal definition below.

Definition 10 (Simulation Extractability). *A proof system Π is called simulation extractable with labels if for any PPT adversary $\mathcal{A}$ and security parameter λ there exists a negligible function $\mathsf{negl}(\cdot)$ such that:*

$$\Pr[(CRS, td_{\mathsf{sim}}, td_{\mathsf{ext}}) \xleftarrow{\$} \mathsf{SimSetup}(1^\lambda); (Stmt^*, L^*, \pi) \leftarrow \mathcal{A}^{\mathcal{O}_{Sim}(td_{\mathsf{sim}}, \cdot, \cdot)}(CRS);$$
$$W \leftarrow \mathsf{Extract}(CRS, td_{\mathsf{ext}}, \pi, Stmt^*, L^*) : \mathsf{Verify}(CRS, \pi, Stmt^*, L^*) = 1 \wedge$$
$$(W, Stmt^*) \notin R \ \wedge \ \mathcal{O}_{Sim} \text{ was never queried with } (Stmt^*, L^*)] \leq \mathsf{negl}(\lambda).$$

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 [2, 3]. We denote the randomization algorithm of signatures by SPS.Rand. We denote the randomizable elements of a SPS signature Σ by $\psi_{\mathtt{rnd}}(\Sigma)$ and the other elements by $\psi_{\mathtt{wit}}(\Sigma)$. (For a non-randomizable SPS signature $\psi_{\mathtt{wit}}(\Sigma) = \Sigma$.)

Our construction does not utilize any randomizability in the vector commitment scheme itself. In the full version [10] we analyze batch-binding and opening non-malleability in presence of such a randomization algorithm.

Construction.

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

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

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

URS.Derive$(pk, I, \boldsymbol{m}, \sigma)$. First, compute $W = \mathsf{VC.Open}(pp, I, \boldsymbol{m}, r)$. 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 $\pi \leftarrow \Pi.\mathsf{Prove}(CRS; C, W, \psi_{\mathtt{wit}}(\Sigma'); \aleph\, C, W, \psi_{\mathtt{wit}}(\Sigma') \ : \ \mathsf{SPS.Verify}(pk_{sps}, \Sigma', C) \wedge \mathsf{VC.Check}(pp, C, \boldsymbol{m}_I, W)\ )$. Return $\sigma = (\psi_{\mathtt{rnd}}(\Sigma'), \pi)$ as the signature on $\boldsymbol{m}_I$.

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

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 provided in the full version [10].

Theorem 5. URS *is an unlinkable redactable signature scheme if the proof-of-knowledge system is witness indistinguishable.* The proofs are given in the full version of this paper [10].

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^*.

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

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

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

Sign, Derive, Verify are almost unchanged and use pk^* internally. SGenT and $\mathsf{ExtractKey}$ use the extraction setup and extractor of the proof system respectively, while $\mathsf{CheckKeys}$ 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 [62]. 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) [5] 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 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 the full version [10].

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 $\boldsymbol{a}$ 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 $\boldsymbol{a}_I$. 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.

A more detailed discussion of these algorithms is given in the full version [10]. 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 hard relation to generate credential

specific secrets. A hard relation has a generator $\mathsf{K}_{\mathsf{Rgap}}$ that generates a witness (X_{Cred} and a public value Y_{Cred}), and a verification algorithm $\mathsf{V}_{\mathsf{Rgap}}$, 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 a network adversary that observes the issuing protocol from impersonating the user.

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

To tame the complexity of definitions for credential systems with many different issuers, we chose to give a definition $\mathcal{F}_{\mathsf{Cred}}$ of a scheme for a single issuer only, that then can be used as building block to a a full-fledged credential system with multiple issuers. The single issuer functionally $\mathcal{F}_{\mathsf{Cred}}$ will just allow users to get a credential on a pseudonym from the issuer and to prove ownership of a credential by the issuer w.r.t. a given pseudonym.

To serve as a secure building block, $\mathcal{F}_{\mathsf{Cred}}$ must be carefully designed. On the one hand it must deal with the unforgeability of credentials and on the other hand it must provide the hooks such that colluding users cannot mix and match credentials from different issuers. To address the former $\mathcal{F}_{\mathsf{Cred}}$ binds issued credentials to the respective users' secret key X and for the latter $\mathcal{F}_{\mathsf{Cred}}$ will enforce that credential proofs will not verify w.r.t. a pseudonym P unless a corresponding credential got issued to the X underlying that pseudonym. Then, provided adversarial users are unable to provide different X's for the same pseudonym, credentials from different issuers issued to different users (i.e., different X's) cannot be matched. As a consequence of this, the generation of user secret keys and pseudonyms is not done inside $\mathcal{F}_{\mathsf{Cred}}$ but users are require to input their secret key X the pseudonym P (as cryptographic values) to $\mathcal{F}_{\mathsf{Cred}}$ on the calls to get credentials issued or to generate a credential proof. Thus we assume that algorithms $(\mathsf{SecGen}, \mathsf{NymGen}, \mathsf{NymVerify})$ to generate user secret keys, to generate pseudonyms, and to verify pseudonyms are available. $\mathcal{F}_{\mathsf{Cred}}$ is given $\mathsf{NymVerify}$ as an input parameter and will use this algorithm, to check the relation between P and X. For the security properties guaranteed by $\mathcal{F}_{\mathsf{Cred}}$, we do not make any assumptions about the security properties of these algorithms. However, for the security of the overall credential scheme, pseudonym need to be commitments to X, i.e., to be binding and hiding w.r.t. X.

In the following we provide the definition of $\mathcal{F}_{\mathsf{Cred}}$ and a protocol $\mathcal{R}_{\mathsf{Cred}}$ that realizes $\mathcal{F}_{\mathsf{Cred}}$ using $\mathcal{F}_{\mathsf{CA}}$ and $\mathcal{F}_{\mathsf{CRS}}$, assuming static corruptions.

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. These algorithms allow $\mathcal{F}_{\mathsf{Cred}}$ to

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 $(pk_{\mathsf{URS}}, sk_{\mathsf{URS}}) \leftarrow \mathsf{URS.KeyGen}(SP_{\mathsf{URS}})$, 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.

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

$\mathsf{Cred.RequestCred}(SP, pk, X, P, aux(P))$:
$(X_{Cred}, Y_{Cred}) \leftarrow \mathsf{K}_{\mathsf{Rgap}}$; $\pi_{X,P} \leftarrow \Pi.\mathsf{Prove}(CRS_X; (X, X_{Cred}, aux(P)); Stmt_P)$, where $Stmt_P = \big(\lambda\!\!\lambda\, X, aux(P) : \mathsf{NymVerify}(SP, X, P, aux(P)) = 1\ \big)$.
Add $X_{Cred}, Y_{Cred}, P, aux(P)$ to $aux(Cred)$ and Y_{Cred} to $\pi_{X,P}$.

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

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 $\sigma \leftarrow \mathsf{URS.Sign}(sk, (P, Y_{Cred}, \boldsymbol{a}))$ and output $Cred = \sigma$.

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

Presentation algorithms

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

1. Obtain X_{Cred}, Y_{Cred}, P, $aux(P)$ from $aux(Cred)$ and σ from $Cred$.
2. Run $\sigma_I \leftarrow \mathsf{URS.Derive}(pk, I, (P, Y_{Cred}, \boldsymbol{a}), \sigma))$.
3. Compute a proof of knowledge of the secret, pseudonym, where the context is non-malleably attached as a label to the proof:
   $\pi_{X,Cred} = \Pi.\mathsf{Prove}(CRS_X; (X, P,$
   $aux(P), aux(P)', Y_{Cred}, X_{Cred}); Stmt, cxt); Stmt =$
   $\big(\lambda\!\!\lambda\, X, P, aux(P), aux(P)', Y_{Cred}, X_{Cred} : \mathsf{NymVerify}(SP, X, P', aux(P)') =$
   $1 \wedge \mathsf{NymVerify}(SP, X, P, aux(P)) = 1 \wedge \mathsf{URS.Verify}(pk, \sigma_I, (P, Y_{Cred}, \boldsymbol{a})_I)) =$
   $1 \ \wedge\ \mathsf{V}_{\mathsf{Rgap}}(X_{Cred}, Y_{Cred}) = 1\big)$.
   Add σ_I to $\pi_{X,Cred}$ as a part of the public input.

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

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 $\big(SP = (SP_{\mathsf{URS}}, CRS_X, pp_{\mathsf{SPC}}), td_{ext} = (td, td_{ext}), td_{sim}\big)$.

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

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

1. $X \leftarrow \mathsf{SecGen}(SP)$; $(P, aux(P)) \leftarrow \mathsf{NymGen}(SP, X)$.
2. Let $\boldsymbol{a}_0$ be a Tier 1 message restored from $\boldsymbol{a}_I$ by replacing $\perp$-s with 0-s as if it was derived from the original message $\boldsymbol{a}$ by applying $\mathsf{Zero}(\boldsymbol{a}, I)$.
3. $\sigma \leftarrow \mathsf{URS.Sign}(sk, (P, Y_{Cred}, \boldsymbol{a}_0))$
4. $\sigma_I \leftarrow \mathsf{URS.Derive}(pk, I, (P, Y_{Cred}, \boldsymbol{a}_0), \sigma))$.
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

produce cryptographic artifacts for proofs of credential ownership and attribute disclosure, to verify such proofs (when they are coming from the adversary), and to extract values from proofs. (We note that there are no artifacts for the credentials themselves.) While these algorithms can be completely adversarial, $\mathcal{F}_{\mathsf{Cred}}$ will enforce that algorithms and the artifacts produced by them) satisfy the required unforgeability and privacy properties. In fact, because of the privacy properties, $\mathcal{F}_{\mathsf{Cred}}$ needs to run these algorithms itself and cannot ask the simulator for the artifacts as is sometimes done (cf. $\mathcal{F}_{\mathsf{URS}}$ and the UC-functionalities for blind signatures [6, 45]).

We now describe the steps of our ideal functionality $\mathcal{F}_{\mathsf{Cred}}$ (cf. Figure 1) and explain the security properties it ensures and how it does so. Note that because we consider static corruption, $\mathcal{F}_{\mathsf{Cred}}$ and $\mathcal{SIM}$ are aware of which parties are corrupted.

$\mathcal{F}_{\mathsf{Cred}}$ maintains two tables for bookkeeping: $\mathcal{M}_{ISS}$ stores information about issued credentials and $\mathcal{M}_{PRES}$ stores information about credentials that produced presentation proofs. It then handles requests as follows. Upon receiving a $(\texttt{keygen}, sid)$ message, $\mathcal{F}_{\mathsf{Cred}}$ performs a setup by asking the simulator for the system parameters, the keys of the issuer, trapdoors, a set of algorithms and a list of corrupted parties. The message $(\texttt{leakSK}, sid)$ is handled in exactly the same way as for redactable signatures.

Next, upon receiving a $(\texttt{issueCred}, sid, qid, \mathcal{U}, X, P, aux(P))$ message from a user $\mathcal{U}$, $\mathcal{F}_{\mathsf{Cred}}$ initiates credential issuing by sending a corresponding message to the issuer specified in $sid = (\mathcal{I}, sid')$. If $\mathcal{I}$ responds to the request with a list of attributes $\boldsymbol{a}$, $\mathcal{F}_{\mathsf{Cred}}$ verifies that X, P, and $aux(P)$ form a valid pseudonym (i.e., $\mathsf{NymVerify}$ outputs 1), and, if so, records in $\mathcal{M}_{ISS}$ that a credential with attributes $\boldsymbol{a}$ to user $\mathcal{U}$ w.r.t. secret X has been issued.

Upon receiving a credential proof request in the form of a $(\texttt{proveCred}, \ldots)$ message, $\mathcal{F}_{\mathsf{Cred}}$ verifies whether the provided X, P, and $aux(P)$ form a valid pseudonym and whether a credential with attributes $\boldsymbol{a}$ to user $\mathcal{U}$ w.r.t. secret X has been issued. Then, $\mathcal{F}_{\mathsf{Cred}}$ creates a cryptographic artifact for the proof using the $\mathsf{Cred.SimProve}$ algorithm where no information that must not be revealed is input to the algorithm. This will guarantee the privacy properties of the credential proof for honest users. Furthermore, before outputting the proof to the user, $\mathcal{F}_{\mathsf{Cred}}$ will verify it using $\mathsf{Cred.Verify}$ as to ensure correctness.

Finally, upon receiving the $(\texttt{verifyCredProof}, \ldots)$ message, $\mathcal{F}_{\mathsf{Cred}}$ has to determine whether or not the proof should be accepted. Here we need to deal with unforgeability of credential proofs (and thus of credentials) if the issuer is honest and its secret key has not been leaked. Naturally, $\mathcal{F}_{\mathsf{Cred}}$ should accept proofs that it has generated itself. Apart from that, $\mathcal{F}_{\mathsf{Cred}}$ could in principle just accept all proofs for which the revealed attributes correspond to a credential that was issued. This would allow the adversary to also produce proofs that match credentials that were not issued to dishonest users but only to an honest user. To prevent this, we require an extraction algorithm $\mathsf{Cred.Extract}$ which, on input a credential proof, will generate a user secret. Then, $\mathcal{F}_{\mathsf{Cred}}$ will accept a credential proof only if the revealed attributes correspond to a credential that

was issued to a corrupted users w.r.t. the X' extracted from the proof. That, however, would still allow (dishonest) users to mix and match their credentials. Therefore, $\mathcal{F}_{\mathsf{Cred}}$ will accept the proofs only if the extracted X' underlies the pseudonym P' w.r.t. which the proof verifies. Therefore, $\mathcal{F}_{\mathsf{Cred}}$ checks the latter using $\mathsf{NymVerify}$.

Realization of $\mathcal{F}_{\mathsf{Cred}}$. A protocol $\mathcal{R}_{\mathsf{Cred}}$ that realizes $\mathcal{F}_{\mathsf{Cred}}$ can be obtained from the algorithms described 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 replaced by running the corresponding algorithms. The detailed description of $\mathcal{R}_{\mathsf{Cred}}$ is given in the full version [10].

For efficiency reasons related to the integration of pseudonyms (which requires zero-knowledge proofs 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 hybrid steps of our security proof.

Theorem 7. *Let* URS *be an unlinkable redactable signature scheme according to Definition 1,* SPC *be a structure-preserving commitment scheme,* Rgap *be a verifiable relation, 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* Rgap *relation is hard.* The proof is provided in the full version [10].

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, together with algorithms ($\mathsf{SecGen}, \mathsf{NymGen}$, $\mathsf{NymVerify}$) to generate user secret keys, to generate pseudonyms, and to verify pseudonyms. The pseudonyms are required to be both hiding and binding w.r.t. the user secret to provide privacy to the honest users and to prevent colluding users from sharing credentials unless they all user the same user secret. A user now can generate a user secret and different pseudonyms on them and then use multiple calls to the $\mathcal{F}_{\mathsf{Cred}}$ instances for different issuers to get credentials on her pseudonyms. 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. Because the pseudonym is the same in different proofs and each proof guarantees the same underlying secret in the credential and the pseudonym, the collection of these proofs together results in a single proof for multiple credentials. Each proof block guarantees unlinkability and unforgeability, and because the pseudonym is both binding and hiding this composed proof is also unforgeable and unlinkable with respect to other proof collections. The verification is done by querying the corresponding $\mathcal{F}_{\mathsf{Cred}}$ instances for verification of each particular proof part and by checking that the pseudonym is the same in each proof part. A formal definition of a full-fledged credential scheme and proof that the scheme just sketched meets it is left as future work.

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

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

Fig. 1. The ideal functionality for single issuer anonymous credentials

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 [49], the structure-preserving commitment scheme of [4], and our unlinkable redactable signature scheme presented in §3.3. We use disjunctive proofs to make the proof system simulation-extractable [23], see [59] for the efficient instantiation with 48 group elements overhead in the XDH setting that forms the basis of our efficiency analysis. As a hard relation we pick the Computational Diffie-Hellman problem. The URS scheme is instantiate with the fully structure-preserving signature scheme by Abe et al. [5], 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, n-RootDH, n-BSDH, q-SDH, XDLIN, co-CDH, and DBP assumptions hold.* Consult building blocks for definitions of assumptions.

We refer to the full version [10] 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 178 group elements (148 when using the SPS of [2] instead of FSPS). This means that the communication efficiency for showing a credential with respect to a pseudonym is around 11 KB (9 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 comparable in efficiency with the credential system of Izabachène et al. [55] that has credential proofs of around 8 KB, while our UC scheme has larger proof sizes. Our scheme is much less efficient than the scheme of [51] 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 [51]—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 n-RootDH assumption of Definition 9.

Acknowledgments

The authors would like to thank Sherman Chow and the anonymous reviewers for their helpful comments and suggestions. The research leading to these results was supported in part by the European Community's Seventh Framework Programme for the project FutureID (grant agreement no. 318424).

References

1. 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.
2. M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. Structure-preserving signatures and commitments to group elements. CRYPTO 2010.
3. M. Abe, J. Groth, K. Haralambiev, and M. Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. CRYPTO 2011.
4. M. Abe, K. Haralambiev, and M. Ohkubo. Group to group commitments do not shrink. EUROCRYPT 2012.
5. M. Abe, M. Kohlweiss, M. Ohkubo, and M. Tibouchi. Fully structure-preserving signatures and shrinking commitments. EUROCRYPT 2015.
6. M. Abe and M. Ohkubo. A framework for universally composable non-committing blind signatures. ASIACRYPT 2009.
7. J. H. Ahn, D. Boneh, J. Camenisch, S. Hohenberger, A. Shelat, and B. Waters. Computing on authenticated data. TCC 2012.
8. N. Attrapadung, B. Libert, and T. Peters. Computing on Authenticated Data: New Privacy Definitions and Constructions. ASIACRYPT 2012.
9. M. H. Au, W. Susilo, and Y. Mu. Constant-size dynamic k-TAA. ISCN 2006.
10. J. Camenisch, M. Dubovitskaya, K. Haralambiev, and M. Kohlweiss. Composable & Modular Anonymous Credentials: Definitions and Practical Constructions. Cryptology ePrint Archive, Report 2015/580.
11. M. Backes, S. Meiser, and D. Schröder. Delegatable functional signatures. Cryptology ePrint Archive, Report 2013/408.
12. M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. Randomizable proofs and delegatable anonymous credentials. CRYPTO 2009.
13. M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. P-signatures and noninteractive anonymous credentials. TCC 2008.
14. M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. Compact e-cash and simulatable VRFs revisited. PAIRING 2009.
15. M. Bellare and G. Fuchsbauer. Policy-based signatures. Cryptology ePrint Archive, Report 2013/413.
16. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. EUROCRYPT 2003.
17. D. Boneh and X. Boyen. Short signatures without random oracles. EUROCRYPT 2004.
18. E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. Cryptology ePrint Archive, Report 2013/401.
19. S. Brands. Untraceable off-line cash in wallets with observers (extended abstract). CRYPTO 1994.
20. S. Brands. Restrictive blinding of secret-key certificates. EUROCRYPT 1995.
21. E. F. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. ACM CCS 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. ACNS 2010.

23. J. Camenisch, N. Chandran, and V. Shoup. A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. EUROCRYPT 2009.
24. J. Camenisch, M. Dubovitskaya, G. Neven, and G. M. Zaverucha. Oblivious transfer with hidden access control policies. PKC 2011.
25. J. Camenisch and T. Groß. Efficient attributes for anonymous credentials. ACM CCS 2008.
26. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. EUROCRYPT 2005.
27. J. Camenisch, A. Kiayias, and M. Yung. On the portability of generalized schnorr proofs. EUROCRYPT 2009.
28. J. Camenisch, S. Krenn, A. Lehmann G.L. Mikkelsen and G. Neven and M.. Pedersen Formal Treatment of Privacy-Enhancing Credential Systems. SAC 2015. Cryptology ePrint Archive, Report 2014/708.
29. J. Camenisch, S. Krenn, and V. Shoup. A framework for practical universally composable zero-knowledge protocols. ASIACRYPT 2011.
30. J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. EUROCRYPT 2001.
31. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. SCN 2002.
32. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. CRYPTO 2004.
33. J. Camenisch, G. Neven, and A. Shelat. Simulatable adaptive oblivious transfer. EUROCRYPT 2007.
34. J. Camenisch and E. Van Herreweghen. Design and implementation of the idemix anonymous credential system. ACM CCS 2002.
35. S. Canard and R. Lescuyer. Protecting privacy by sanitizing personal data: a new approach to anonymous credentials. ASIACCS 2013.
36. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. IEEE FOCS 2001.
37. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited (preliminary version). ACM STOC 1998.
38. D. Catalano and D. Fiore. Vector commitments and their applications. PKC 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.
41. M. Chase, M. Kohlweiss, A. Lysyanskaya, and S. Meiklejohn. Malleable signatures: New definitions and delegatable anonymous credentials. IEEE CSFS 2013.
42. M. Chase and A. Lysyanskaya. On signatures of knowledge. CRYPTO 2006.
43. D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. CRYPTO 1988.
44. S. S. M. Chow, Y. J. He, L. C. K. Hui, and S. M. Yiu. SPICE - simple privacy-preserving identity-management for cloud environment. ACNS 2012.
45. M. Fischlin. Round-optimal composable blind signatures in the common reference string model. CRYPTO 2006.
46. G. Fuchsbauer. Commuting signatures and verifiable encryption. EUROCRYPT 2011.
47. G. Fuchsbauer, C. Hanser, and D. Slamanig. Euf-cma-secure structure-preserving signatures on equivalence classes. Cryptology ePrint Archive, Report 2014/944.
48. V. Goyal. Reducing trust in the PKG in identity based cryptosystems. CRYPTO 2007.

49. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. EUROCRYPT 2008.
50. 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 2008.
51. C. Hanser and D. Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. ASIACRYPT 2014.
52. M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. EUROCRYPT 2000.
53. D. Hofheinz and E. Kiltz. Secure hybrid encryption from weakened key encapsulation. CRYPTO 2007.
54. D. Hofheinz and V. Shoup. Gnuc: A new universal composability framework. *IACR Cryptology ePrint Archive*, Report 2011/303.
55. M. Izabachène, B. Libert, and D. Vergnaud. Block-wise p-signatures and non-interactive anonymous credentials with efficient attributes. *IMA Int. Conf.* 2011.
56. A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. ASIACRYPT 2010.
57. A. Kiayias and M. Yung. Group signatures with efficient concurrent join. EUROCRYPT 2005.
58. A. Kiayias and H.-S. Zhou. Equivocal blind signatures and adaptive uc-security. TCC 2008.
59. M. Kohlweiss and I. Miers. Accountable tracing signatures. Cryptology ePrint Archive, Report 2014/824.
60. M. Kohlweiss and A. Rial. Optimally private access control. In ACM WPES 2013.
61. B. Libert and M. Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. TCC 2010.
62. S. Meiklejohn. An extension of the groth-sahai proof system. In *Brown University Masters thesis*, 2009.
63. R. Nojima, J. Tamura, Y. Kadobayashi, and H. Kikuchi. A storage efficient redactable signature in the standard model. ISC 2009.