Privacy-Enhancing Proxy Signatures from Non-Interactive Anonymous Credentials*

David Derler, Christian Hanser, and Daniel Slamanig

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology (TUG), Inffeldgasse 16a, 8010 Graz, Austria
{david.derler|christian.hanser|daniel.slamanig}@tugraz.at

Abstract. Proxy signatures enable an originator to delegate the signing rights for a restricted set of messages to a proxy. The proxy is then able to produce valid signatures only for messages from this delegated set on behalf of the originator. Recently, two variants of *privacy-enhancing proxy signatures*, namely blank signatures [27] and warrant-hiding proxy signatures [28], have been introduced. In this context, *privacy-enhancing* means that a verifier of a proxy signature does not learn anything about the delegated message set beyond the message being presented for verification.
We observe that this principle bears similarities with functionality provided by anonymous credentials. Inspired by this observation, we examine black-box constructions of the two aforementioned proxy signatures from non-interactive anonymous credentials, i.e., anonymous credentials with a non-interactive showing protocol, and show that the so obtained proxy signatures are secure if the anonymous credential system is secure. Moreover, we present two concrete instantiations using well-known representatives of anonymous credentials, namely Camenisch-Lysyanskaya (CL) and Brands' credentials.
While constructions of anonymous credentials from signature schemes with particular properties, such as CL signatures or structure-preserving signatures, as well as from special variants of signature schemes, such as group signatures, sanitizable and indexed aggregate signatures, are known, this is the first paper that provides constructions of special variants of signature schemes, i.e., privacy-enhancing proxy signatures, from anonymous credentials.

Keywords: Proxy signatures, anonymous credentials, cryptographic protocols, privacy, provable security.

1 Introduction

Proxy signatures allow an *originator* to delegate signing rights to a *proxy*, who is then able to issue signatures on behalf of the originator (cf. [8] for various secure constructions). To restrict the delegation, Mambo et al. [29] introduced the concept of a warrant, which basically encodes a policy describing the delegation of the originator and is signed by the originator using a conventional digital signature scheme as part of the delegation. For instance, such a warrant can be used to restrict the set of messages (message space) a proxy is allowed to sign messages from. In all known constructions, however, the warrant is revealed to every verifier, which could lead to

* The authors have been supported by the European Commission through project FP7-FutureID, grant agreement number 318424. This is the full version of [20].

privacy issues. When, for instance, delegating the signing rights for a contract containing multiple choices for a price to a proxy the whole price range would be revealed to any verifier. We call proxy signatures *privacy-preserving*, if they address this issue and do not reveal the warrant upon verification, while still allowing to check whether the message signed by the proxy is covered by the warrant. We note that this concept must not be confused with anonymous proxy signatures [24], which aim at hiding the identity of the delegatee and all intermediate delegators. In this paper, we consider two recently proposed instantiations of privacy-enhancing proxy signature schemes, namely warrant-hiding proxy signatures [28] (WHPS) as well as blank digital signatures [27] (BDS). Roughly speaking, WHPS allow to delegate the signing rights for a set of messages $\mathcal{M}$, e.g., $\mathcal{M} = \{M_1, \ldots, M_4\}$, to a proxy. Given a proxy signature anyone is able to verify the validity of such a signature and the delegation while not learning anything about the remaining delegated message space. Similarly, BDS allow for the delegation of the signing rights for a template $\mathcal{T}$ containing fixed and exchangeable strings (called elements) to a proxy, who is then able to sign a filled in version of such a template on behalf of the originator. Thereby, fixed elements can not be changed by the proxy, while exchangeable elements allow the proxy to choose one message out of a set of predefined messages, e.g., $\mathcal{T} = (M_1, \{M_{2_1}, M_{2_2}, M_{2_3}\}, M_3)$ with M_1 and M_3 being fixed elements. Upon verification, again, anyone is able to verify the validity of the signature and delegation while not learning anything about the unused choices in the exchangeable elements.

We observe, that this principle bears similarities with functionality provided by anonymous credentials. In an anonymous credential system, an organization issues a credential on attributes (which can be viewed as messages in the delegation) and the showing of a credential amounts to selectively opening some of the attributes (messages), while only proving knowledge of the undisclosed attributes. If the showing, thereby, is non-interactive and includes proving knowledge of a secret key, it can be seen as issuing a digital signature. Loosely speaking, for instance, in case of WHPS, one would use the messages in the warrant, i.e., $\mathcal{M} = \{M_1, \ldots, M_n\}$, and the public key of the proxy as attributes of the credential. A proxy signature then amounts to a non-interactive showing of the chosen message and the proxy public key, while only proving knowledge of the remaining message space and the proxy secret key (without revealing it).

1.1 Contribution

In this paper we provide black-box constructions of the two aforementioned privacy-enhancing proxy signature schemes from non-interactive anonymous credentials. Therefore, we provide an explicit encoding of message spaces to attributes of the credential systems. We show that a secure credential system together with this encoding implies the security of the respective privacy-enhancing proxy signature scheme. Furthermore, we present two instantiations based on non-interactive versions of well known Brands' [9] and CL [13] credentials, obtained by applying the Fiat-Shamir heuristic [22] and being secure in the random oracle model. Moreover, we compare the so obtained signature schemes to the originally proposed BDS and WHPS constructions and discuss why they may represent an alternative in specific scenarios. To the best of our knowledge, the presented constructions constitute the first approach to construct special signatures schemes from anonymous credentials, which may be of independent interest and inspiring for the design of other signatures.

1.2 Related Work

In [5], Belenkiy et al. propose a model for practical non-interactive anonymous credentials being secure in the standard model, which uses Groth-Sahai proofs [26]. In [6], Bellare and Fuchsbauer use similar building blocks, i.e., structure preserving signatures [2] and Groth-Sahai proofs, to construct what they call policy based signatures. This approach basically allows for defining policies enforcing certain properties on signed messages. In order to issue correct signatures w.r.t. a policy, one is additionally required to be in possession of a signing key for this particular policy. Their model allows for defining policies for any language in $\mathcal{NP}$, which is then restricted to group dependent languages due to using Groth-Sahai proofs in their instantiation. Furthermore, Backes et al. [4] propose a model for delegating the signing rights for messages being derivable from an initial message by applying a particular functionality to the message.

In [23], Fuchsbauer and Pointcheval introduce a generalized model for anonymous proxy signatures and group signatures. The latter concept is conceptually very similar to anonymous credentials and often anonymous credentials are built from group signatures. Though, to the best of our knowledge, no formal implications regarding the security models of the aforementioned concepts exist. Quite recently, two (black-box) constructions for anonymous credentials from aggregate signatures [16], as well as sanitizable signatures [17] were proposed. In a way, this is the opposite of what we are going to show in this paper.

2 Preliminaries

We use additive notation for groups $\mathbb{G}$ which are always of prime order p. A function $\epsilon : \mathbb{N} \to \mathbb{R}^+$ is called *negligible* if for all $c > 0$ there is a k_0 such that $\epsilon(k) < 1/k^c$ for all $k > k_0$. In the remainder of this paper, we use ϵ to denote such a negligible function.

Bilinear Map: A bilinear map (pairing) is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, with $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ being cyclic groups of prime order p. Let P and P' generate $\mathbb{G}_1$ and $\mathbb{G}_2$. We require e to be efficiently computable and to satisfy:

Bilinearity: $e(aP, bP') = e(P, P')^{ab} = e(bP, aP') \quad \forall a, b \in \mathbb{Z}_p$

Non-degeneracy: $e(P, P') \neq 1_{\mathbb{G}_T}$, i.e., $e(P, P')$ generates $\mathbb{G}_T$.

If $\mathbb{G}_1 = \mathbb{G}_2$, e is called *symmetric* and *asymmetric* otherwise.

Zero-knowledge Proofs of Knowledge: We use the notation from [14] for denoting the proof of knowledge (PoK) of a discrete logarithm $x = \log_P Y$ to the base P, i.e., $\mathsf{PoK}\{(\alpha) : Y = \alpha P\}$, whereas Greek letters always denote values whose knowledge will be proven. The non-interactive version of such a proof can be obtained using the Fiat-Shamir [22] transform, which is then also called a signature of knowledge (SoK) [18]. When such a proof includes proving knowledge of a secret key, it is a secure digital signature in the random oracle model. Such a signature is interpreted as the signature of the proxy in our setting and is followingly denoted as π. A formal definition of secure digital signature schemes is given in Appendix A.

3 Anonymous Credentials

In an anonymous credential system there is an organization as well as different users. Thereby, the organization issues credentials to users, who can then anonymously demonstrate possession

of these credentials to verifiers. Such a system is called *multi-show* when showings carried out by the same user cannot be linked and *one-show* otherwise. A credential cred_i for user i issued by the organization in such a system includes a set $\mathbb{A} = \{(\mathtt{attr}_\ell, dom(\mathtt{attr}_\ell))\}_{\ell=1}^{n}$ of attribute labels $\mathtt{attr}_\ell$ and corresponding domain $dom(\mathtt{attr}_\ell)$ from which attribute labels can take their values. When we speak of a set $\mathbb{A}_i$ for user i, we mean a subset of $\mathbb{A}$ such that for every $\mathtt{attr}_\ell$ contained in the set, the second element of the tuple takes some concrete value from $dom(\mathtt{attr}_\ell)$. Whenever a user i demonstrates possession of a credential for a subset $\mathbb{A}'_i$ of $\mathbb{A}_i$, we write $\mathbb{A}'_i \sqsubseteq \mathbb{A}_i$ to denote that the showing is compatible with $\mathbb{A}_i$. This means that all selectively disclosed attribute values have been issued for this credential and that all statements proven about attribute values can be proven from the issued attribute values.

3.1 Abstract Model of Anonymous Credentials

Subsequently, we give an abstract definition of an anonymous credential system.

$\mathsf{Setup}(\kappa, t)$: Gets a security parameter κ and an upper bound t for $|\mathbb{A}|$ and returns the global parameters pp.

$\mathsf{OrgKeyGen}(\mathsf{pp})$: Takes pp and produces an organization key pair $(\mathsf{osk}, \mathsf{opk})$.

$\mathsf{UserKeyGen}(\mathsf{pp}, i)$: Takes pp and $i \in \mathbb{N}$ and produces a key pair $(\mathsf{usk}_i, \mathsf{upk}_i)$ for user i.

$(\mathsf{Obtain}(\mathsf{pp}, \mathsf{opk}, \mathsf{usk}_i), \mathsf{Issue}(\mathsf{pp}, \mathsf{osk}, \mathsf{upk}_i, \mathbb{A}_i))$: These algorithms are run by user i and the organization, who interact during execution. Obtain takes input global parameters pp, the user's secret key usk_i and the organization's public key opk. Issue takes input pp, the user's public key upk_i, the organization's secret key osk and a set $\mathbb{A}_i$ of size n. At the end of this protocol, Obtain outputs a credential cred_i for $\mathbb{A}_i$ for user i and the (updated) secret key usk'_i.

$(\mathsf{Show}(\mathsf{pp}, \mathsf{opk}, \mathsf{usk}_i, \mathsf{cred}_i, \mathbb{A}_i, \mathbb{A}'_i), \mathsf{Verify}(\mathsf{pp}, \mathsf{opk}, \mathbb{A}'_i))$: These algorithms are run by user i and a verifier who interact during the execution. Show takes input global parameters pp, the user's secret key usk_i, the organization's public key opk, a credential cred_i with a corresponding set $\mathbb{A}_i$ of size n and a second set $\mathbb{A}'_i \sqsubseteq \mathbb{A}_i$ of size n' with $n' \leq n$. Verify takes input the public parameters pp, the public key opk and a set $\mathbb{A}'_i$. At the end of the protocol, Show outputs an (updated) credential cred'_i and the (updated) user's secret key usk'_i. Verify outputs $\mathtt{true}$ upon a valid showing and $\mathtt{false}$ otherwise.

We note that in some models the entire key generation is executed by the Setup algorithm. However, we find it more natural to split these algorithms into three algorithms. Furthermore, we note that if there are multiple organizations, then $\mathsf{OrgKeyGen}$ is run by every single organization (on potentially distinct pp).

There are various definitions of security for anonymous credential systems [3,12,16,17], which differ in their details as they are sometimes tailored to specific constructions. However, they are essentially only slightly different ways of defining the properties *unforgeability* and *anonymity* in addition to the usual *correctness* property. *Correctness* means that a showing of a credential w.r.t. a set $\mathbb{A}'_i$ of attributes and values must always verify if the credential was issued honestly w.r.t. $\mathbb{A}_i$ such that $\mathbb{A}'_i \sqsubseteq \mathbb{A}_i$. *Unforgeability* means that an adversary can not succeed in showing a credential which is accepted by a verifier, unless a credential w.r.t. to the shown attributes has been issued to it. *Anonymity* means that no adversary, even playing the role of the organization, should be able to identify the user when showing a credential. Furthermore, different showings of a user w.r.t. the same credential must be unlinkable in multi-show anonymous credential systems.

Finally, we require a property denoted as *selective disclosure*. This is not covered by the security definition of [3], which we are going to use, but is an informal requirement for all anonymous credential systems. There is a simulation based notion capturing this fact [5], which, however, turns out to be not useful for relating the security properties to our constructions. However, we can assume that any reasonable anonymous credential system satisfies this notion, i.e., even if the user is known, a showing transcript must not reveal any information about attributes beyond the attributes revealed during showing [10]. This is underpinned by the fact that all known anonymous credential systems employ (non-interactive) proofs of knowledge in their showing protocols and such proofs by definition do not reveal anything beyond what is shown. A formal definition of secure anonymous credential systems is provided in Appendix C.1.

Non-interactive anonymous credential systems: If interaction between the user and the verifier when executing (Show, Verify) algorithms is not required, we call an anonymous credential system non-interactive. These steps can, thus, be executed in isolation and the output of the Show algorithm serves as input for the Verify algorithm. In constructions of credential systems it is straightforward to make the showing non-interactive and the output of the Show algorithm can, thus, be considered as a signature of knowledge.

3.2 Two Concrete Anonymous Credential Systems

Camenisch-Lysyanskaya (CL) credentials [11, 13] are constructed from commitment schemes and efficient protocols for proving the equality of two committed values and a signature scheme with efficient protocols. Latter protocols are for obtaining a signature on a committed value (without revealing the value) and proving the knowledge of it. The used signature schemes support re-randomization, meaning that one can take a signature and compute another signature for the same message without the signing key, such that the signatures are unlinkable. Thus, the resulting credential systems are *multi-show*. Brands' credentials [9] are built from blind signatures which do not support re-randomization and, therefore, represent a one-show credential system.

The two aforementioned approaches are the basis for our instantiations of privacy-enhancing proxy signatures from non-interactive anonymous credential systems. A more detailed discussion of these schemes is provided in Appendix B.

3.3 Remarks on Anonymous Credentials in our Constructions

For our black-box constructions, we need to make some clarifications before being able to use an arbitrary anonymous credential system.

First of all, in order to model the delegation, the designated proxy's public key always needs to be encoded within an attribute, being opened upon every non-interactive Show. Therefore, we assume that the user's public key (corresponding to its secret signing key) fits to the system parameters of the anonymous credential scheme. If the proxy's key does not fit to the system parameters of the used scheme, one could include a hash value of the user's public key as an attribute and require the user to sign the output of the non-interactive Show algorithm using the corresponding secret key (latter is not considered here). Moreover, in the case of BDS also a second attribute containing the size of the template needs to be included and always opened during showing. As already mentioned, we require the showing of the anonymous credential scheme to be non-interactive and each non-interactive showing is required to include a proof of knowledge

of the secret key corresponding to the public key included in the first attribute. This constitutes a signature of knowledge and is interpreted as the proxy's signature.

Finally, we want to mention that the anonymity property of anonymous credential schemes is stronger than what is required for BDS or WHPS. While we only require the hiding of attributes (selective disclosure) which have not been opened, anonymous credentials also require unlinkability of issuing and showing, which is not necessary for BDS and WHPS, but does not influence our constructions. Similarly, we do not require the multi-show unlinkability, but it does not really influence our constructions as well. One may explicitly enforce breaking the unlinkability by requiring the credential issuer to additionally issue a conventional digital signature on the credential and accepting the credential only if the signature is valid. Conversely, the unlinkability may also be seen as an additional feature for BDS and WHPS, respectively (cf. Section 7).

4 Privacy-Enhancing Proxy-Type Signatures

This section is intended to give a brief overview of the privacy-enhancing proxy signature schemes. Section 4.1 discusses the Blank Digital Signature Scheme (BDSS) proposed in [27], whereas Section 4.2 discusses the Warrant-Hiding Proxy Signature Scheme (WHPSS) proposed in [28].

4.1 Blank Digital Signatures

The BDSS allows an *originator* to delegate the signing rights for a certain *template* to a *proxy*. Based on such a delegation, the *proxy* is able to issue a signature on a so called instance of a template on behalf of the *originator*. A template $\mathcal{T}$ is a sequence of non-empty sets of bitstrings T_i, where these sets are either called *fixed* or *exchangeable*, depending on the cardinality of the respective set. More precisely, exchangeable elements contain more than one bitstring, whereas fixed elements contain exactly one bitstring. Such a template is formally defined as $T_i = \{M_{i_1}, M_{i_2}, \ldots, M_{i_k}\},\ \mathcal{T} = (T_1, T_2, \ldots, T_n)$.

The template length is defined as the sequence length n of the template, while the template size $|\mathcal{T}|$ is defined as $|\mathcal{T}| = \sum_{i=1}^{n} |T_i|$. An *originator* issues a signature for a template, which also specifies the proxy. Based on this so-called template signature, the designated *proxy* can take the fixed elements, choose concrete values for each exchangeable element, and compute a so-called instance signature for an instance $\mathcal{M}$, which is formally defined as $\mathcal{M} = (M_i)_{i=1}^{n}$. If $\mathcal{M}$ is a correct instantiation of $\mathcal{T}$, we write $\mathcal{M} \preceq \mathcal{T}$.

Given an instance signature, anyone is able to verify its validity, i.e., verify the delegation, whether $\mathcal{M}$ has been signed by the proxy and if $\mathcal{M} \preceq \mathcal{T}$ holds. Thereby, the original template, that is, the unused values of the exchangeable elements of the template, can not be determined (the so called *privacy* property). Formally, a BDSS is defined as follows [27]:

$\mathsf{KeyGen}(\kappa, t)$**:** On input of a security parameter κ and an upper bound for the template size t the public parameters pp are generated. We assume pp to be an input to all subsequent algorithms.

$\mathsf{Sign}(\mathcal{T}, \mathsf{dsk_O}, \mathsf{dpk_P})$**:** Given a template $\mathcal{T}$, the secret signing key of the originator $\mathsf{dsk_O}$ and the public verification key of the proxy $\mathsf{dpk_P}$, this algorithm outputs a template signature $\sigma_{\mathcal{T}}$ and a secret template signing key for the proxy $\mathsf{sk_P^{\mathcal{T}}}$.

$\mathsf{Verify_T}(\mathcal{T}, \sigma_\mathcal{T}, \mathsf{dpk_O}, \mathsf{dpk_P}, \mathsf{sk_P^\mathcal{T}})$**:** Given a template $\mathcal{T}$, a template signature $\sigma_\mathcal{T}$, the public verification keys of originator and proxy ($\mathsf{dpk_O}, \mathsf{dpk_P}$) and the template signing key of the proxy $\mathsf{sk_P^\mathcal{T}}$, this algorithm checks whether $\sigma_\mathcal{T}$ is a valid signature for $\mathcal{T}$ and returns `true` on success and `false` otherwise.

$\mathsf{Inst}(\mathcal{T}, \sigma_\mathcal{T}, \mathcal{M}, \mathsf{dsk_P}, \mathsf{sk_P^\mathcal{T}})$**:** On input a template $\mathcal{T}$ with corresponding signature $\sigma_\mathcal{T}$, an instance $\mathcal{M} \preceq \mathcal{T}$, as well as the secret template signing key $\mathsf{sk_P^\mathcal{T}}$ and the secret signing key of the proxy $\mathsf{dsk_P}$, this algorithm outputs a signature $\sigma_\mathcal{M}$ for $\mathcal{M}$.

$\mathsf{Verify_M}(\mathcal{M}, \sigma_\mathcal{M}, \mathsf{dpk_O}, \mathsf{dpk_P})$**:** Given an instance $\mathcal{M}$, an instance signature $\sigma_\mathcal{M}$ and the public verification keys of originator and proxy ($\mathsf{dpk_O}, \mathsf{dpk_P}$), this algorithm verifies whether $\sigma_\mathcal{M}$ is a valid signature on $\mathcal{M}$ and $\mathcal{M} \preceq \mathcal{T}$ (for an unknown $\mathcal{T}$). On success, this algorithm outputs `true` and `false` otherwise.

The security of a BDSS is defined as follows [27]. *Correctness* states that for all honestly generated parameters and keys it is required that for any template $\mathcal{T}$ and honestly computed template signature $\sigma_\mathcal{T}$ and corresponding $\mathsf{sk_P^\mathcal{T}}$, the verification always succeeds and for the originator it is intractable to find a template signature that is valid for different templates (in the sense of non-repudiation of [31]). Furthermore, for any honestly computed instance signature $\sigma_\mathcal{M}$, the verification always succeeds. *Unforgeability* requires that without the knowledge of $\mathsf{dsk_O}, \mathsf{dsk_P}$ and $\mathsf{sk_P^\mathcal{T}}$ it is intractable to forge template or message signatures. *Immutability* means that for a proxy (in possession of $\mathsf{sk_P^\mathcal{T}}, \mathsf{dsk_P}, \mathcal{T}$ and $\sigma_\mathcal{T}$) it is intractable to forge template signatures or instance signatures which are not described in the respective template. *Privacy* captures that no verifier (except for the originator and the proxy) can learn anything about $\mathcal{T}$ besides what is revealed by instance signatures. More formal security definitions are provided in Appendix C.2.

4.2 Warrant-Hiding Proxy Signatures

A WHPSS allows an *originator* to delegate the signing rights for a message from a well defined message space $\mathcal{M}$ (sometimes also denoted as ω) to a *proxy*. The message space $\mathcal{M}$ is, thereby, a non-empty set of bitstrings (messages) M_i, i.e., $\mathcal{M} = \{M_1, \ldots, M_n\}$. A proxy is then able to choose one bitstring M_i from the message space $\mathcal{M}$ and issue a proxy signature σ_P on behalf of the originator for M_i. A verifier given M_i and σ_P can verify the validity of the signature and the delegation, while the remaining message space $(\mathcal{M} \setminus M_i)$ stays concealed.

One could argue that the functionality of WHPSS can be easily modeled by the originator by separately signing each message in $\mathcal{M}$ and to let the proxy then countersign a message of its choice. However, using this naive approach would allow the proxy to repudiate that a particular message was contained in the delegated message space. In contrast, one can open the warrant contained in the WHPSS proxy signature in case of a dispute in front of a judge.

Formally, a WHPSS is defined as follows [28] (we note that Setup is not included as a separate algorithm in [28], but this makes no real difference):

$\mathsf{Setup}(\kappa, t)$**:** On input of a security parameter κ and an upper bound t for the size of the message space, this algorithm generates the public system parameters ppk. Note that ppk is assumed to be an input to all subsequent algorithms.

$(\mathsf{D}(\mathcal{M}, \mathsf{pk}_i, \mathsf{sk}_i, j, \mathsf{pk}_j), \mathsf{P}(\mathsf{pk}_j, \mathsf{sk}_j, \mathsf{pk}_i))$**:** The originator and the proxy jointly compute a delegation for the message space $\mathcal{M}$ as well as a proxy signing key skp. The originator runs D and thereby computes the delegation and outputs the delegation σ computed using its signing key

sk_i, whereas the proxy verifies the delegation and obtains the proxy signing key skp, which consists of its private signing key sk_j and the originators delegation.

$\mathsf{PS}(\mathsf{skp}, M)$**:** This algorithm computes and outputs a proxy signature σ_P for message $M \in \mathcal{M}$ using the proxy signing key skp.

$\mathsf{PV}(\mathsf{pk}_i, M, \sigma_P)$**:** This algorithm verifies whether proxy signature σ_P is a valid proxy signature for message M under pk_j, delegated by pk_i. On success, this algorithm outputs `true`, and `false` otherwise.

$\mathsf{ID}(\sigma_P)$**:** This algorithm outputs the identity j of the proxy, when given a proxy signature σ_P.

The security of a WHPSS is defined as follows [28]. *Correctness* requires that for all honestly computed parameters and for all skp obtained by running (D, P), it holds that for all warrants and proxy signatures for a message M the algorithm PV accepts a signature for M if M is in the warrant and rejects it otherwise. Furthermore, ID is required to return the correct proxy. *Unforgeability* states that, without the knowledge of the originator's and the proxy's secret key, it is intractable to produce valid delegations and/or proxy signatures which are either inside or outside the warrant. *Privacy* requires that any verifier distinct form the originator and the proxy can not efficiently decide whether a given message (except the ones being revealed by proxy signatures) lies within the warrant when given a proxy signature. Formal security definitions are given in Appendix C.3.

5 From Anonymous Credentials to Proxy-Signatures

Subsequently, we show how privacy-enhancing proxy signatures can be built from non-interactive anonymous credential systems. Therefore, we use the abstract notion of an anonymous credential system introduced in Section 3 and map the algorithms to the corresponding algorithms of the respective proxy signature scheme. Furthermore, we introduce an encoding to attributes in order to achieve the same properties as the proxy signature schemes.

The basic idea behind using an anonymous credential system for modeling privacy-enhancing proxy signatures is that we interpret the elements of a template (or the warrant) together with the public key of the designated proxy and the template length as attributes of a credential issued by an originator (organization). On verification, the proxy only reveals the attributes belonging to the instantiation of the template (or reveals one attribute corresponding to a message from the warrant) while hiding all others. We note that the organization's keypair $(\mathsf{opk}, \mathsf{osk})$ in the anonymous credential scheme is interpreted as the keypair of the originator in the proxy signature schemes and the user's keypair $(\mathsf{upk}_i, \mathsf{usk}_i)$ is the keypair of proxy i. We use this notation of the anonymous credential model henceforth.

5.1 Mapping from Templates and Warrants to Attributes

In both proxy signature approaches, a finite sequence/set of strings needs to be encoded as attributes of a credential, where in the case of BDSS this sequence represents a template and in case of WHPSS the set represents a warrant. The ideas behind the encoding are quite similar, although the BDSS case is a little trickier. Before presenting the encodings, we require some operations on sets and sequences. Firstly, we define an operator $\mathsf{Expand}(\cdot, \cdot)$, which takes an integer k and a set $S = \{s_1, \ldots, s_n\}$ as input and returns a sequence of tuples. This operator

assigns a unique position to each element of the set, e.g., by means of their lexicographic order, and encodes the elements together with the integer k in a sequence. More precisely, we define an output sequence a as:

$$a = ((s_1, k), \ldots, (s_n, k)) := \mathsf{Expand}(k, \{s_1, \ldots, s_n\}).$$

When we apply the concatenation operator $||$ to two sequences, e.g., $(x)_{i=1}^{n}||(y)_{i=1}^{m}$, the result is a sequence of the form $(x_1, \ldots, x_n, y_1, \ldots, y_m)$. For the concatenation of $\ell \geq 2$ sequences $s_1, \ldots, s_\ell$ we write $||_{i=1}^{\ell} s_i$. Moreover, we require an operator $\mathsf{Hash}(\cdot)$ which takes a sequence a of tuples as input and returns the sequence a' of corresponding hash values obtained by applying a secure hash function $H : \{0,1\}^* \times \{0,1\}^* \to \mathbb{Z}_p$ to each element in the sequence. The i-th element of such a sequence a' obtained from a is further referred to as $h_i := H(s_i, k)$. Note that we use H to allow for messages/attribute values of arbitrary length.

BDSS: In the original construction of BDSS presented in [27], templates are encoded as polynomials and each template element constitutes a root of the so called encoding polynomial. With such an encoding polynomial at hand, one can not derive anything about the order of the elements within the template and, in further consequence, this property hides the structure of the template. In contrast, anonymous credential systems typically assume an ordering of the attributes within the credential (cf. Section 3.2), and, thus, would leak information about the structure of a template. Let us, for instance, consider a template $\mathcal{T} = (M_1, \{M_{2_1}, M_{2_2}, M_{2_3}\}, M_3, \{M_{4_1}, M_{4_2}\})$. Here, each element M_i would be encoded within one attribute in the credential. While the unused choices of the exchangeable elements are hidden upon Show, information on the cardinality and position of exchangeable elements can leak due to the order of the attributes.

Template encoding: In order to map templates $\mathcal{T}$ and instances $\mathcal{M}$, as defined in Section 4.1, the first processing step is to apply the following transformation:

$$\mathcal{T} \leftarrow \mathsf{Hash}(||_{i=1}^{n} \mathsf{Expand}(i, T_i)).$$

Subsequently prefixing $\mathcal{T}$ with the (authentic) public key upk_i of the designated proxy and the template size $|\mathcal{T}|$ would already deliver a suitable encoding for our constructions. However, as mentioned above, such an encoding can leak information about the structure of the template. In order to prevent this kind of leakage, we further apply a random permutation ϕ to the expanded and hashed template, i.e., $\mathcal{T} \leftarrow (\mathsf{upk}_i, |\mathcal{T}|, \phi(\mathcal{T}))$.

In doing so, the order of the attributes becomes independent of their position in the template, and, thus, the template structure is hidden as in the original BDSS construction. Subsequently, this mapping is denoted as $\mathsf{Enc}_{\mathcal{T}}^{\mathsf{BDS}}$.

Example: In the following, we illustrate the encoding of a template by a simple example: $\mathcal{T} = \{\{\text{"}\texttt{A}\text{"}, \text{"}\texttt{B}\text{"}\}, \text{"}\texttt{declares to pay}\text{"}, \{\text{"}\texttt{50\$}\text{"}, \text{"}\texttt{100\$}\text{"}\}\}$. Note that the example is only for illustration and not meant to reflect a real world application. After the operation $\mathsf{Hash}(||_{i=1}^{3} \mathsf{Expand}(i, T_i))$ we obtain the sequence:

$$(H(\text{"}\texttt{A}\text{"}, 1), H(\text{"}\texttt{B}\text{"}, 1), H(\text{"}\texttt{ declares to pay }\text{"}, 2), H(\text{"}\texttt{50\$.}\text{"}, 3), H(\text{"}\texttt{100\$.}\text{"}, 3)),$$

and continue by applying a random permutation ϕ:

$$(H(\text{"}\texttt{100\$.}\text{"}, 3), H(\text{"}\texttt{50\$.}\text{"}, 3), H(\text{"}\texttt{ declares to pay }\text{"}, 2), H(\text{"}\texttt{A}\text{"}, 1), H(\text{"}\texttt{B}\text{"}, 1)).$$

This leads to a template $\mathcal{T}^{enc} = (\mathsf{upk}_i, |\mathcal{T}|, h_{3_2}, h_{3_1}, h_2, h_{1_1}, h_{1_2})$.

Instance encoding: The encoding of instances $\mathcal{M}$ corresponding to a given template $\mathcal{T}$ does not substantially differ from the encoding of templates. Additionally to the public key upk_i of the proxy and the template size $|\mathcal{T}|$, the following information is included: a sequence $\mathcal{M}'$ containing tuples corresponding to the chosen elements, each containing the element itself, its position in the template and its position in the sequence $\mathcal{T}^{enc}$ according to the permutation ϕ. Furthermore, one includes a signature of knowledge (SoK) π, which represents a proof of knowledge of usk_i and the non-revealed template elements:

$$\mathcal{M}^{enc} \leftarrow (\mathsf{upk}_i, |\mathcal{T}|, \mathcal{M}', \pi).$$

For our further explanations, this mapping is denoted as $\mathsf{Enc}^{\mathsf{BDS}}_{\mathcal{M}}$. Observe that given $\mathcal{M}'$ in $\mathcal{M}^{enc}$, one can not directly use it in a verification, but for every tuple (s, i, j) in $\mathcal{M}'$ one has to compute $h_j = H(s, i)$, which then represents the value of the j'th attribute. Subsequently, we assume that this step is implicitly computed by a verifier whenever $\mathcal{M}^{enc}$ is provided for verification.

Example: Continuing the above example, we assume that "`B`" and "`50$`" are chosen as final values in the exchangeable elements, which leads to the following encoded message $\mathcal{M}^{enc}$:

$$\mathcal{M}^{enc} = (\mathsf{upk}_i, |\mathcal{T}|, (\text{"}\texttt{B}\text{"}, 1, 5), (\text{" }\texttt{declares to pay }\text{"}, 2, 3), (\text{"}\texttt{50\$}\text{"}, 3, 2), \pi).$$

Note that the indices indicating the position in the template sequence according to the permutation ϕ implicitly fix the indices for the sequence of unrevealed values. Furthermore, due to the random permutation ϕ being random and secret (only known to the originator and the proxy), the positions of the unrevealed attributes are independent of their position in the template, and, thus, no information about the template structure leaks. In contrast, when considering the example above, omitting the random permutation would lead to revealing that the first and the last template elements are exchangeable elements with 2 choices each.

We also emphasize that both, the encoding function $\mathsf{Enc}^{\mathsf{BDS}}_{\mathcal{T}}$ and the encoding function $\mathsf{Enc}^{\mathsf{BDS}}_{\mathcal{M}}$, take the secret random permutation ϕ (only known to the originator and the proxy) as additional parameter.

WHPSS: The mapping in terms of the WHPSS is a lot easier since, firstly, no explicit order has to be enforced within the messages in the warrant and, secondly, the order of the messages can not leak any useful information.

In order to encode a WHPSS message space for our setting, we redefine the operator $\mathsf{Expand}(\cdot)$ as a unary operator converting a set to a sequence by assigning a unique position to each element from the set. Furthermore, we also redefine H as $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$. The encoding of a message space $\mathcal{M}$ then looks as follows:

$$\mathcal{M}^{enc} \leftarrow (\mathsf{upk}_i, \mathsf{Hash}(\mathsf{Expand}(\mathcal{M}))).$$

Similarly, a message chosen by the proxy is encoded by choosing a message $M_k \in \mathcal{M}$ and computing a signature of knowledge (SoK) π of usk_i and the remaining messages in the warrant:

$$M \leftarrow (\mathsf{upk}_i, M_k, k, \pi).$$

Observe, that M_k cannot be directly used as an attribute value, but needs to be mapped to $H(M_k)$. However, as above we assume that this step is implicitly computed by the verifier whenever M_k is provided for verification. We refer to the encoding defined above as $\mathsf{Enc}_{\mathcal{M}}^{\mathsf{WHPS}}$ and $\mathsf{Enc}_{M}^{\mathsf{WHPS}}$ for our further explanations and note a secret random permutation ϕ is not required.

5.2 Constructing BDS from Anonymous Credentials

We assume that a credential is issued on an encoded template $\mathcal{T}^{enc}$ using the encoding defined above. Upon showing, the proxy chooses a concrete instantiation $\mathcal{M}^{enc}$ for a template by disclosing the elements corresponding to the instance $\mathcal{M}^{enc}$, while providing a signature of knowledge for the elements remaining in $\mathcal{T}^{enc}$. To be more precise, the proxy always discloses the attributes representing the public key and containing the size of the template, as well as at least one element for each position in the template, and provides a signature of knowledge of the secret signing key and the unused choices for the exchangeable elements. We assume that every user (proxy) i has run $\mathsf{AC.UserKeyGen}(\mathsf{pp}, i)$ to obtain $(\mathsf{usk}_i, \mathsf{upk}_i)$ compatible with pp locally. Furthermore, the template secret key $\mathsf{sk}_{\mathsf{P}}^{\mathcal{T}}$ is the secret random permutation ϕ. Below, we provide the abstract definition of the construction, where AC denotes an anonymous credential system with non-interactive showing.

$\mathsf{KeyGen}(\kappa, t)$: This algorithm computes the public parameters pp by running $\mathsf{AC.Setup}(\kappa, t)$ and specifies the encodings $\mathsf{Enc}_{\mathcal{T}}^{\mathsf{BDS}}$ and $\mathsf{Enc}_{\mathcal{M}}^{\mathsf{BDS}}$. Then, it runs $\mathsf{AC.OrgKeyGen}(\mathsf{pp})$ to obtain $(\mathsf{osk}, \mathsf{opk})$ and outputs all these parameters. The public parameters pp as well as a description of the encoding functions are assumed to be available to all subsequent algorithms.

$\mathsf{Sign}(\mathcal{T}, (\mathsf{opk}, \mathsf{osk}), \mathsf{upk}_i)$: This algorithm chooses a random permutation ϕ and computes $\mathcal{T}^{enc} \leftarrow \mathsf{Enc}_{\mathcal{T}}^{\mathsf{BDS}}(\mathcal{T}, \phi)$. Then, it locally runs $(\mathsf{AC.Obtain}(\mathsf{pp}, \mathsf{opk}, \mathsf{upk}_i)$[1], $\mathsf{AC.Issue}(\mathsf{pp}, \mathsf{osk}, \mathsf{upk}_i, \mathcal{T}^{enc}))$ and the results, i.e., the credential cred_i as template signature and the template-specific secret key ϕ for the proxy, are returned.

$\mathsf{Verify}_{\mathsf{T}}(\mathcal{T}, \mathsf{cred}_i, \mathsf{opk}, (\mathsf{upk}_i, \mathsf{usk}_i), \phi)$: This algorithm computes $\mathcal{T}^{enc} \leftarrow \mathsf{Enc}_{\mathcal{T}}^{\mathsf{BDS}}(\mathcal{T}, \phi)$ and checks the validity of the credential cred_i using usk_i and opk. On success, this algorithm returns `true`, and `false` otherwise.

$\mathsf{Inst}(\mathcal{T}, \mathsf{cred}_i, \mathcal{M}, (\mathsf{opk}, \mathsf{upk}_i, \mathsf{usk}_i), \phi)$: This algorithm computes an encoding $\mathcal{M}^{enc}$ of an instantiation $\mathcal{M}$ of the template $\mathcal{T}$ using ϕ by computing a SoK π including a proof of the user's secret key usk_i and the unused choices of the exchangeable elements, i.e., $\mathsf{AC.Show}$ is executed. The instance signature (π, cred_i) and the encoded message $\mathcal{M}^{enc}$ are returned.

$\mathsf{Verify}_{\mathsf{M}}(\mathcal{M}^{enc}, (\pi, \mathsf{cred}_i), \mathsf{opk}, \mathsf{upk}_i)$: This algorithm verifies whether π is a valid signature of knowledge w.r.t. $\mathcal{M}^{enc}$ and upk_i by executing $\mathsf{AC.Verify}$. On success, this algorithm returns `true`, and `false` otherwise.

5.3 Constructing WHPS from Anonymous Credentials

Similarly, we interpret the messages in the delegated message space $\mathcal{M}$ in the WHPSS as attributes. We assume that the credential is issued for $\mathcal{M}^{enc}$ using the encoding defined above. Upon showing, the proxy reveals the attributes representing the public key and the chosen message M together with a proof of knowledge for the unrevealed messages in the warrant and the

[1] As we assume that the user's key pair fits to the system parameters, we do not require usk_i as an input to the $\mathsf{AC.Obtain}$ algorithm and so the credential is issued using upk_i as public commitment to usk_i. This allows the originator to run both algorithms locally.

user's secret key. Again, we assume that every user j (proxy) has run $\mathsf{AC.UserKeyGen}(\mathsf{pp})$ to obtain $(\mathsf{usk}_j, \mathsf{upk}_j)$ compatible with pp locally. Furthermore, the secret proxy signing key is defined to be $(\mathsf{cred}_j, \mathsf{usk}_j)$. Below, we provide the abstract definition of the construction, where AC denotes an anonymous credential system with non-interactive showing. For the ease of presentation we include the key generation of the originators keys into the Setup algorithm.

$\mathsf{Setup}(\kappa, t)$: This algorithm creates public parameters pp by running $\mathsf{AC.Setup}(\kappa, t)$ and specifies the encodings $\mathsf{Enc}_{\mathcal{M}}^{\mathsf{WHPS}}$ and $\mathsf{Enc}_{M}^{\mathsf{WHPS}}$. Then it runs $\mathsf{AC.OrgKeyGen}(\mathsf{pp})$ to obtain $(\mathsf{osk}, \mathsf{opk})$ and outputs all these parameters. The public parameters pp as well as a description of the encoding functions are assumed to be available to all subsequent algorithms.

$(\mathsf{D}(\mathcal{M}, \mathsf{opk}, \mathsf{osk}, j, \mathsf{upk}_j), \mathsf{P}(\mathsf{upk}_j, \mathsf{usk}_j, \mathsf{opk}))$**:** This algorithm computes the encoding of the message space $\mathcal{M}^{enc} \leftarrow \mathsf{Enc}_{\mathcal{M}}^{\mathsf{WHPS}}(\mathcal{M})$. Then, $(\mathsf{AC.Obtain}(\mathsf{pp}, \mathsf{opk}, \mathsf{usk}_j), \mathsf{AC.Issue}(\mathsf{pp}, \mathsf{osk}, \mathsf{upk}_j, \mathcal{M}^{enc}))$ is jointly executed by the originator (acting as the organization) and the proxy (acting as the user). The user obtains as local output the credential cred_j for the sequence $\mathcal{M}^{enc}$ encoding the message space (which represents the proxy signing key skp).

$\mathsf{PS}((\mathsf{cred}_j, \mathsf{usk}_j), M)$: This algorithm computes a signature of knowledge π including a proof of the user's secret key usk_j and the remaining message space from $\mathcal{M}^{enc}$, i.e., executes the $\mathsf{AC.Show}$ algorithm non-interactively and returns the proxy signature (π, cred_j).

$\mathsf{PV}((\mathsf{opk}, \mathsf{upk}_j), M, (\pi, \mathsf{cred}_j))$: This algorithms verifies whether π is a valid signature of knowledge w.r.t. M and upk_j by executing $\mathsf{AC.Verify}$. On success, this algorithm outputs `true`, and `false` otherwise.

$\mathsf{ID}(\pi)$: Given the output π of the Show algorithm, this algorithm outputs the public key of the corresponding proxy.

5.4 From AC Security to BDS and WHPS Security

In this section, we argue that if we have a secure non-interactive anonymous credential system AC, the constructions of the BDS and WHPS schemes from AC are also secure. Consequently, when building such schemes in the proposed way, these schemes provide adequate security within their respective models.

We note that the unforgeability properties of all these approaches are compatible. Yet, the anonymity property required from a credential system is much stronger than what is required from BDS and WHPS. Basically, a goal achieved by an anonymous credential system is the indistinguishability of showings of different users, which have credentials to identical attributes, with respect to any verifier (including the issuer). In contrast, the goal of the proxy signature schemes is to hide the non-shown "attributes" from any external verifier, whereas the issuer (the originator) knows all attributes. Consequently, we relate the privacy of the schemes to the selective disclosure of the anonymous credential system. Next, we discuss both approaches.

Blank Digital Signatures: In BDS, the required correctness, unforgeability and immutability properties are implied by the unforgeability of an AC system. The privacy is implied by the selective disclosure property of the AC system. In Appendix D.1 we show the following:

Theorem 1. *If* AC *represents a secure anonymous credential system and the hash function used in the encodings* $\mathsf{Enc}_{\mathcal{T}}^{\mathsf{BDS}}$ *and* $\mathsf{Enc}_{\mathcal{M}}^{\mathsf{BDS}}$ *is secure, then the* BDS *from Section 5.2 based on* AC *is secure.*

Warrant-Hiding Proxy Signatures: In WHPS, the required correctness and unforgeability properties are implied by the unforgeability of an AC system. Furthermore, privacy is implied by the selective disclosure of the AC system. In Appendix D.2 we show the following:

Theorem 2. *If* AC *represents a secure anonymous credential system and the hash function used in the encoding* $\mathsf{Enc}_{\mathcal{M}}^{\mathsf{WHPS}}$ *is secure, then the* WHPS *scheme from Section 5.3 based on* AC *is secure.*

6 Instantiations from CL and Brands' Credentials

In this section, we provide two instantiations of BDS making use of CL [13] and Brands' [9] credentials, respectively. We omit the constructions of WHPS as after having seen the construction for BDS, the construction of WHPS is straightforward. In both presented schemes, we assume the keypair of the proxy $(\mathsf{upk}, \mathsf{usk})$ to be compatible with the system parameters, i.e., usk is a scalar in $\mathbb{Z}_p$ and $\mathsf{upk} = \mathsf{usk} \cdot P$, with P being a generator of the respective group.

Furthermore, with hide we denote the elements of $\mathcal{T}^{enc}$ corresponding to the elements in $\mathcal{T}$ without $\mathcal{M}$, whereas with show we denote the elements of $\mathcal{M}^{enc}$ corresponding to elements in $\mathcal{M}$.

In Scheme 1, we present our construction of BDS from CL credentials [13] in detail. Our second instantiation builds up on Brands' one-show credentials, following the *certificates based on Chaum-Pedersen signatures* approach proposed in [9]. In Scheme 2, we present our construction in detail.

Setup(κ, t)**:** Choose an appropriate group $\mathbb{G}$ of large prime order p such that a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ exists. Further, choose a generator P of $\mathbb{G}$, as well as $x, y \xleftarrow{R} \mathbb{Z}_p$. With t being the maximal template size, select $z_i \xleftarrow{R} \mathbb{Z}_p$ for $0 \leq i \leq t$ and compute $X \leftarrow xP, Y \leftarrow yP, Z_i \leftarrow z_i P$. The algorithm outputs $\mathsf{pp} = (\mathbb{G}, \mathbb{G}_t, e, P, p, \mathsf{Enc}_{\mathcal{T}}^{\mathsf{BDS}}, \mathsf{Enc}_{\mathcal{M}}^{\mathsf{BDS}})$, $\mathsf{opk} \leftarrow (X, Y, Z_1, \ldots, Z_t)$ and $\mathsf{osk} \leftarrow (x, y, z_1, \ldots, z_t)$.

Sign$(\mathcal{T}, (\mathsf{opk}, \mathsf{osk}), \mathsf{upk})$**:** Choose $\alpha \xleftarrow{R} \mathbb{Z}_p$ and compute $R \leftarrow \alpha P$, $A_i \leftarrow z_i R, B \leftarrow yR, B_i \leftarrow yA_i$. Further, choose a random permutation ϕ and compute $\mathcal{T}^{enc} \leftarrow \mathsf{Enc}_{\mathcal{T}}^{\mathsf{BDS}}(\mathcal{T}, \phi)$. Then, $\mathsf{upk}^* \leftarrow \alpha \cdot \mathsf{upk} = \alpha \cdot \mathsf{usk} \cdot P$. Compute $C \leftarrow x \cdot R + xy \cdot \mathsf{upk}^* + xy \cdot |\mathcal{T}| \cdot A_0 + \sum_{h_i \in \mathcal{T}^*} xy \cdot h_i A_i$ and return the credential $\mathsf{cred} \leftarrow (R, \{A_i\}, B, \{B_i\}, C)$ and the template-specific proxy secret key ϕ.

Verify$_{\mathsf{T}}(\mathcal{T}, \mathsf{cred}, \mathsf{opk}, (\mathsf{upk}, \mathsf{usk}), \phi)$**:** Compute $\mathcal{T}^{enc} \leftarrow \mathsf{Enc}_{\mathcal{T}}^{\mathsf{BDS}}(\mathcal{T}, \phi)$ and verify whether cred is a valid signature under opk, i.e., $e(R, Z_i) \stackrel{?}{=} e(P, A_i) \wedge e(R, Y) \stackrel{?}{=} e(P, B) \wedge e(A_i, Y) \stackrel{?}{=} e(P, B_i)$ and $e(X, R) \cdot e(X, B)^{\mathsf{usk}} \cdot e(X, B_0)^{|\mathcal{T}|} \prod_{h_i \in \mathcal{T}^{enc}} e(X, B_i)^{h_i} \stackrel{?}{=} e(P, C)$ holds and return `true` on success and `false` otherwise.

Inst$(\mathcal{T}, \mathsf{cred}, \mathcal{M}, (\mathsf{opk}, \mathsf{upk}, \mathsf{usk}), \phi)$**:** Using $\mathcal{T}^{enc}$ and $\mathcal{M}^{enc}$, obtained by applying the encoding functions w.r.t. ϕ and compute $\mathsf{v}_x \leftarrow e(X, R), \mathsf{v}_{xy} \leftarrow e(X, B), \mathsf{v}_{(xy,i)} \leftarrow e(X, B_i), \mathsf{v}_s \leftarrow e(P, C)$,

$$\pi \leftarrow \mathsf{SoK}\left\{ (\{(\mu_i)_{m_i \notin \mathcal{M}}, \chi_{\mathsf{usk}}\}) : \begin{array}{c} \mathsf{v}_s = \mathsf{v}_x \mathsf{v}_{xy}^{\chi_{\mathsf{usk}}} \mathsf{v}_{(xy,0)}^{|\mathcal{T}|} \prod\limits_{\mu_i \in \mathsf{hide}} \mathsf{v}_{(xy,i)}^{\mu_i} \prod\limits_{h_i \in \mathsf{show}} \mathsf{v}_{(xy,i)}^{h_i} \\ \wedge\ \chi_{\mathsf{usk}} \cdot P = \mathsf{upk} \end{array} \right\},$$

Return the instance signature (π, cred) and the encoded message $\mathcal{M}^{enc}$.

Verify$_{\mathsf{M}}(\mathcal{M}^{enc}, (\pi, \mathsf{cred}), \mathsf{opk}, \mathsf{upk})$**:** Compute $\mathsf{v}_x \leftarrow e(X, R)$, $\mathsf{v}_{xy} \leftarrow e(X, B)$, $\mathsf{v}_{(xy,i)} \leftarrow e(X, B_i)$ and $\mathsf{v}_s \leftarrow e(P, C)$, check whether $e(R, Z_i) \stackrel{?}{=} e(P, A_i) \wedge e(R, Y) \stackrel{?}{=} e(P, B) \wedge e(A_i, Y) \stackrel{?}{=} e(P, B_i)$ and verify the SoK π w.r.t. $\mathcal{M}^{enc}$, the public key upk and check whether $|\mathcal{T}|$ equals the number of message elements in the proof. On success, return `true` and `false` otherwise.

Scheme 1: BDSS from CL credentials

Setup(κ, t): Let $\mathbb{G}$ be a group of prime order p which is generated by P. Choose $y_0, y_1, \ldots, y_{t+2} \xleftarrow{R} \mathbb{Z}_p$ with t being the maximal template size and compute $H_0 \leftarrow y_0 P, P_1 \leftarrow y_1 P, \ldots, P_{t+2} \leftarrow y_{t+2} P$. The algorithm outputs $\mathsf{pp} \leftarrow (\mathbb{G}, P, p, \mathsf{Enc}_{\mathcal{T}}^{\mathsf{BDS}}, \mathsf{Enc}_{\mathcal{M}}^{\mathsf{BDS}})$, $\mathsf{opk} \leftarrow (H_0, P_1, \ldots, P_{t+2})$ and $\mathsf{osk} \leftarrow (y_0, \ldots, y_{t+2})$.

Sign($\mathcal{T}$, (opk, osk), upk) The originator and the proxy jointly compute a signature on the template $\mathcal{T}^{enc} \leftarrow \mathsf{Enc}_{\mathcal{T}}^{\mathsf{BDS}}(\mathcal{T}, \phi)$ as follows.

Originator		Proxy
$w_0 \xleftarrow{R} \mathbb{Z}_p, A_0 \leftarrow w_0 P$		$\alpha, \alpha_2, \alpha_3 \xleftarrow{R} \mathbb{Z}_p$
$H \leftarrow y_1 \mathsf{upk} + \lvert\mathcal{T}\rvert P_2 + \sum_{i=1}^{\lvert\mathcal{T}\rvert} h_i P_{i+2}$		
$B_0 \leftarrow w_0(H_0 + H), Z \leftarrow y_0(H_0 + H)$	$\xrightarrow{A_0, B_0, H, Z}$	$H' \leftarrow \alpha(H_0 + H), Z' \leftarrow \alpha Z$
		$A'_0 \leftarrow \alpha_2 H_0 + \alpha_3 P + A_0$
		$B'_0 \leftarrow \alpha_2 Z' + \alpha_3 H' + \alpha B_0$
		$c'_0 \leftarrow H(H' \lvert\lvert Z' \lvert\lvert A'_0 \lvert\lvert B'_0)$
	$\xleftarrow{c_0}$	$c_0 \leftarrow c'_0 + \alpha_2 \pmod p$
$r_0 \leftarrow c_0 \cdot y_0 + w_0 \pmod p$	$\xrightarrow{r_0}$	$r_0 P - c_0 H_0 \stackrel{?}{=} A_0$
		$r_0(H_0 + H) - c_0 Z \stackrel{?}{=} B_0$
		$r'_0 \leftarrow r_0 + \alpha_3$

Output the template signature $\mathsf{cred} \leftarrow (H', Z', A'_0, B'_0, r'_0, c'_0)$ and the template-specific proxy secret key (ϕ, α).

Verify$_{\mathsf{T}}$($\mathcal{T}$, cred, opk, (upk, usk), (ϕ, α)): Compute $\mathcal{T}^{enc} \leftarrow \mathsf{Enc}_{\mathcal{T}}^{\mathsf{BDS}}(\mathcal{T}, \phi)$ and $H \leftarrow \mathsf{usk} P_1 + \lvert\mathcal{T}\rvert P_2 + \sum_{i=1}^{\lvert\mathcal{T}\rvert} h_i P_{i+2}$ as well as $H' \leftarrow \alpha(H_0 + H)$, and check whether the value H' contained in cred is equal to the the computed value for H'. Check whether $r'_0(P + H') - c'_0(H_0 + Z') \stackrel{?}{=} A'_0 + B'_0$ holds and return `true` if all checks hold and `false` otherwise.

Inst($\mathcal{T}$, cred, $\mathcal{M}$, (opk, upk, usk), (ϕ, α)): Compute $\mathcal{T}^{enc}$ and $\mathcal{M}^{enc}$ from $\mathcal{T}$, $\mathcal{M}$ and ϕ as well as

$$\pi \leftarrow \mathsf{SoK}\left\{ \left((\mu_i)_{m_i \notin \mathcal{M}}, \alpha, \chi_{\mathsf{usk}}\right) : \begin{array}{l} H' = \alpha(H_0 + \chi_{\mathsf{usk}} P_1 + \lvert\mathcal{T}\rvert P_2 + \sum_{\mu_i \in \mathsf{hide}} \mu_i P_{i+2} + \\ \sum_{h_i \in \mathsf{show}} h_i P_{i+2}) \quad \wedge \quad \chi_{\mathsf{usk}} P = \mathsf{upk} \end{array} \right\}$$

and return the instance signature (π, cred) as well as the encoded message $\mathcal{M}^{enc}$.

Verify$_{\mathsf{M}}$($\mathcal{M}^{enc}$, (π, cred), opk, upk): Verify whether $r'_0(P + H') - c'_0(H_0 + Z') \stackrel{?}{=} A'_0 + B'_0$ holds, verify the SoK π w.r.t. $\mathcal{M}^{enc}$ and the public key upk and check whether $\lvert\mathcal{T}\rvert$ is equal to the number of message elements in the proof. Return `true` if all checks hold and `false` otherwise.

Scheme 2: BDSS from Brands' credentials

7 Comparison and Discussion

In this section, we compare the instantiations of the proxy signature schemes obtained from non-interactive anonymous credentials with the original instantiations of BDS and WHPS from [27,28]. Moreover, we discuss the pros and cons of the various approaches and provide an overview regarding computation, bandwidth and parameter sizes in Table 1.

Firstly, we note that for most practical usecases it can be assumed that template sizes are quite small. Consequently, under this assumption, the fact that in some cases the asymptotic computation times and signature sizes are linear in the size of the template does not have a notable influence on the overall performance of the schemes obtained from anonymous credentials. Though, when a usecase requires larger templates, the originally proposed schemes would be preferable.

However, the credential based constructions are flexible regarding the underlying anonymous credential scheme, which, in turn, could be exploited to reach additional properties. For instance, the unlinkability of multiple instances w.r.t. the same template can be realized by using a multi-show anonymous credential system. Such a multi-show setting immediately leads to a stronger notion of privacy, since even multiple instance signatures w.r.t. the same template, which, viewed all together, would reveal the whole template in the original constructions, would not reveal the whole template in a multi-show setting. Furthermore, the credential based constructions can be

Scheme	Computational effort				Signature size		
	Sign	$\mathsf{Verify}_\mathsf{T}$	Inst	$\mathsf{Verify}_\mathsf{M}$	Params	Cert	σ_P
BDSS	$\mathcal{O}(\lvert\mathcal{T}\rvert)$	$\mathcal{O}(\lvert\mathcal{T}\rvert)$	$\mathcal{O}(\lvert\mathcal{T}\rvert)$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(\lvert\mathcal{T}\rvert)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
$\mathsf{BDSS}_\mathsf{CL}$	$\mathcal{O}(\lvert\mathcal{T}\rvert)$	$\mathcal{O}(\lvert\mathcal{T}\rvert)$	$\mathcal{O}(\lvert\mathcal{T}\rvert)$	$\mathcal{O}(\lvert\mathcal{T}\rvert)$	$\mathcal{O}(\lvert\mathcal{T}\rvert)$	$\mathcal{O}(\lvert\mathcal{T}\rvert)$	$\mathcal{O}(\lvert\mathcal{T}\rvert)$
$\mathsf{BDSS}_\mathsf{Brands}$	$\mathcal{O}(\lvert\mathcal{T}\rvert)$	$\mathcal{O}(\lvert\mathcal{T}\rvert)$	$\mathcal{O}(\lvert\mathcal{T}\rvert)$	$\mathcal{O}(\lvert\mathcal{T}\rvert)$	$\mathcal{O}(\lvert\mathcal{T}\rvert)$	$\mathcal{O}(1)$	$\mathcal{O}(\lvert\mathcal{T}\rvert)$

Scheme	Computational effort					Signature size		
	D	P	PS	PV	ID	Params	Cert	σ_P
$\mathsf{WHPSS}_\mathsf{PolyCommit}$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
$\mathsf{WHPSS}_\mathsf{VectorCommit}$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(\log(\lvert\mathcal{M}\rvert))$	$\mathcal{O}(\log(\lvert\mathcal{M}\rvert))$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(\log(\lvert\mathcal{M}\rvert))$
$\mathsf{WHPSS}_\mathsf{CL}$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(1)$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$
$\mathsf{WHPSS}_\mathsf{Brands}$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(1)$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$	$\mathcal{O}(1)$	$\mathcal{O}(\lvert\mathcal{M}\rvert)$

Table 1. BDSS/WHPSS efficiency comparison.

augmented with an anonymity feature, hiding the identity of the signing proxy. This can be achieved by simply leaving out the part of the proof linking usk and upk ($\chi_{\mathsf{usk}} \cdot P = \mathsf{upk}$). Since usk is still required for a successful computation of the SoK, such a change would not influence the security of the respective scheme.

Moreover, and very important, due to multiple projects such as ABC4Trust [1] building high-level interfaces for credential systems such as IBM's idemix [11, 13, 15] or Microsoft's U-Prove [9, 30], there are quite some implementations of anonymous credential systems available to date. These implementations directly yield a basis for practical implementations of the schemes presented in this paper, which renders them very attractive from a practical point of view.

While the complexities of our instantiations are quite comparable to the originally proposed schemes, our proposed instantiations leave more freedom regarding the choice of groups since there is no pairing friendly elliptic curve group required in Brands' credentials [9] and one could also easily use the RSA based version of CL credentials [11]. This enables implementations on constrained devices such as smart cards (cf. [7, 21]). In contrast, the originally proposed instantiations of BDS as well as one of the instantiation of WHPS require pairing friendly elliptic curve groups.

Finally, we mention that in this paper the first approach for building special signature schemes from anonymous credentials is introduced, which might also be inspiring for other constructions. For instance, one could make use of the proposed encoding to encode finite sets of attribute values into credentials of an anonymous credential systems.

References

1. ABC4Trust Project - Attribute-based Credentials for Trust, `http://abc4trust.eu`
2. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-Preserving Signatures and Commitments to Group Elements. In: CRYPTO'10. LNCS, vol. 6223, pp. 209–236
3. Akagi, N., Manabe, Y., Okamoto, T.: An Efficient Anonymous Credential System. In: FC'08. LNCS, vol. 5143, pp. 272–286
4. Backes, M., Meiser, S., Schröder, D.: Delegatable Functional Signatures. IACR ePrint 2013, 408 (2013)
5. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and Noninteractive Anonymous Credentials. In: TCC '08. LNCS, vol. 4948, pp. 356–374
6. Bellare, M., Fuchsbauer, G.: Policy-Based Signatures. In: PKC'14. LNCS, ext.: IACR ePrint 2013/413
7. Bichsel, P., Camenisch, J., Groß, T., Shoup, V.: Anonymous credentials on a standard java card. In: ACM CCS'09. pp. 600–610. ACM

8. Boldyreva, A., Palacio, A., Warinschi, B.: Secure Proxy Signature Schemes for Delegation of Signing Rights. J. Cryptology 25(1), 57–115 (2012)
9. Brands, S.: Rethinking Public-Key Infrastructures and Digital Certificates: Building in Privacy. MIT Press (2000)
10. Camenisch, J., Groß, T.: Efficient attributes for anonymous credentials. ACM Trans. Inf. Syst. Secur. 15(1), 4 (2012)
11. Camenisch, J., Lysyanskaya, A.: A Signature Scheme with Efficient Protocols. In: SCN'02. LNCS, vol. 2576, pp. 268–289
12. Camenisch, J., Lysyanskaya, A.: An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In: EUROCRYPT'01. LNCS, vol. 2045, pp. 93–118
13. Camenisch, J., Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In: CRYPTO'04. LNCS, vol. 3152, pp. 56–72
14. Camenisch, J., Stadler, M.: Efficient Group Signature Schemes for Large Groups (Extended Abstract). In: CRYPTO'97. LNCS, vol. 1294, pp. 410–424
15. Camenisch, J., Van Herreweghen, E.: Design and implementation of the idemix anonymous credential system. In: ACM CCS'02. pp. 21–30. ACM
16. Canard, S., Lescuyer, R.: Anonymous credentials from (indexed) aggregate signatures. In: ACM DIM'11. pp. 53–62. ACM
17. Canard, S., Lescuyer, R.: Protecting privacy by sanitizing personal data: a new approach to anonymous credentials. In: ASIA CCS '13. pp. 381–392. ACM
18. Chase, M., Lysyanskaya, A.: On Signatures of Knowledge. In: CRYPTO'06. LNCS, vol. 4117, pp. 78–96
19. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In: CRYPTO'94. LNCS, vol. 839, pp. 174–187
20. Derler, D., Hanser, C., Slamanig, D.: Privacy-Enhancing Proxy Signatures from Non-interactive Anonymous Credentials. In: DBSec 2014. pp. 49–65. LNCS (2014)
21. Derler, D., Potzmader, K., Winter, J., Dietrich, K.: Anonymous Ticketing for NFC-Enabled Mobile Phones. In: INTRUST'11. LNCS, vol. 7222, pp. 66–83
22. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: CRYPTO'87. LNCS, vol. 263, pp. 186–194
23. Fuchsbauer, G., Pointcheval, D.: Anonymous consecutive delegation of signing rights: Unifying group and proxy signatures. In: Formal to Practical Security'09, LNCS, vol. 5458, pp. 95–115
24. Fuchsbauer, G., Pointcheval, D.: Anonymous Proxy Signatures. In: SCN'08. LNCS, vol. 5229, pp. 201–217
25. Goldwasser, S., Micali, S., Rivest, R.L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM J. Comput. 17(2), 281–308 (1988)
26. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: EUROCRYPT'08. LNCS, vol. 4965, pp. 415–432
27. Hanser, C., Slamanig, D.: Blank Digital Signatures. In: ACM ASIACCS'13. pp. 95–106. ACM, ext.: IACR ePrint 2013/130
28. Hanser, C., Slamanig, D.: Warrant-Hiding Delegation-by-Certificate Proxy Signature Schemes. In: INDOCRYPT'13. LNCS, vol. 8250. Ext.: IACR ePrint 2013/544
29. Mambo, M., Usuda, K., Okamoto, E.: Proxy signatures for delegating signing operation. In: ACM CCS'96. pp. 48–57. ACM
30. Microsoft: U-Prove, http://research.microsoft.com/en-us/projects/u-prove
31. Stern, J., Pointcheval, D., Malone-Lee, J., Smart, N.P.: Flaws in Applying Proof Methodologies to Signature Schemes. In: CRYPTO'02. LNCS, vol. 2442, pp. 93–110

A Digital Signature Schemes (from Σ-Protocols)

Below, we briefly define digital signature schemes and discuss digital signatures from Σ-protocols.

Definition 1 (Digital Signature Scheme). A *digital signature scheme* DSS is a triple $(\mathsf{DKeyGen}, \mathsf{DSign}, \mathsf{DVerify})$ of PPT algorithms. Thereby, $\mathsf{DKeyGen}$ is a key generation algorithm that takes a security parameter $\kappa \in \mathbb{N}$ as input and outputs a secret (signing) key sk and a public (verification) key pk. Further, DSign is a (probabilistic) algorithm, which takes a message $M \in \{0,1\}^*$ and a secret key sk as input, and outputs a signature σ. Finally, $\mathsf{DVerify}$ is a deterministic algorithm, which takes a signature σ, a message $M \in \{0,1\}^*$ and a public key pk as input, and outputs a single bit $b \in \{\mathtt{true}, \mathtt{false}\}$ indicating whether σ is a valid signature for M under pk.

A digital signature scheme is required to be *correct*, i.e., for all $(\mathsf{sk}, \mathsf{pk}) \in \mathsf{DKeyGen}(\kappa)$ and all $M \in \{0,1\}^*$ one requires $\mathsf{DVerify}(\mathsf{DSign}(M, \mathsf{sk}), M, \mathsf{pk}) = \mathtt{true}$. Additionally, for security one requires existential unforgeability under adaptively chosen-message attacks ($\mathsf{EUF\text{-}CMA}$) [25].

Signatures from Σ-Protocols: An essential ingredient for many constructions of anonymous credentials are honest-verifier zero-knowledge proofs of knowledge (Σ-protocols). Thereby, we will use the notation from [14], i.e., a proof of knowledge of a discrete logarithm $x = \log_P Y$ to the base P will be denoted as $\mathsf{PoK}\{(\alpha) : Y = \alpha P\}$, whereas Greek letters always denote values whose knowledge will be proven. We note that compositions of single Σ-protocols using conjunctions and disjunctions can be efficiently realized [19]. Furthermore, non-interactive versions of (composed) proofs can be obtained by applying the Fiat-Shamir transform [22].

The resulting non-interactive version of a proof PoK is denoted as SoK and often also called a signature of knowledge [18]. If this proof includes proving the knowledge of a secret key with respect to a public key, this constitutes a secure digital signature in the random oracle model. Such a signature is interpreted as the signature of a proxy in our setting and is denoted as π in this paper.

B Credential Systems used in our Constructions

Subsequently, we briefly review both, CL and Brands' credentials, and denote attribute values by a_i which are considered to be values in $\mathbb{Z}_p$. To support larger attributes, the hash-then-commit approach can be used, i.e., hashing a_i to $\mathbb{Z}_p$.

CL Credentials: CL credentials are based on the re-randomizable CL signature scheme [11,13], which allows to sign a message sequence in a way that single elements of the sequence may be provided as (unconditionally hiding) commitments and one can selectively prove knowledge of the signature and message elements. In this work, the pairing-based version [13] is used (cf. Section 6), but, the construction works in similar vein for the RSA based variant of the signature scheme [11]. In CL credentials, basically, the message sequence is treated as an attribute sequence. Thereby, a signature $\sigma = (R, A_i, B, B_i, C)$ for a sequence of $n+1$ attributes $(a_0, \ldots, a_n) \in \mathbb{Z}_p^{n+1}$ w.r.t. to the secret key $(x, y, z_1, \ldots, z_n)$ and a random value $R \in \mathbb{G}$ is computed as follows

$$A_i \leftarrow z_i R, \quad B \leftarrow yR, \quad B_i \leftarrow yA_i, \quad C \leftarrow (x + xya_0)R + \sum_{i=1}^{n-1} xya_iA_i$$

and can be verified using pairings.

The principle of the credential system is that such signatures are issued on commitments to the actual attributes and the user can then selectively disclose attributes (messages) contained in C. However, in our construction we do not require all of the attributes to be blinded upon issuance of the credential, since the attributes (representing messages) can be known to the issuer (originator).

Brands' Credentials: The idea behind Brands' credentials [9] working in a group $\mathbb{G}$ of prime order p is to commit to attributes $(a_1, \ldots, a_n) \in \mathbb{Z}_p^n$ by making use of a discrete logarithm representation (DLREP) w.r.t. public generators $(P_1, \ldots, P_n)$:

$$H \leftarrow \sum_{i=1}^{n} a_i P_i.$$

Once such a DLREP H of the attributes $(a_1, \ldots, a_n)$ is computed, a variant of a blind signature, jointly computed by the organization and the user, is generated. Note that H is additionally blinded in Brands' construction, i.e., is a generalized Pedersen commitment, which is omitted in our above explanation for the sake of simplicity. Showing a credential amounts to checking the validity of the blind signature on H and the user proving knowledge of the attributes encoded in the DLREP. Thereby, certain attributes can be disclosed to the verifier while others stay concealed. In this paper, we rely on the Chaum-Pedersen signature version of Brands' credentials (cf. Section 6), as only this version allows to issue credentials, where not all attributes are disclosed, i.e., known to the issuer, during the issuing protocol. This is required in order to be able to include the user's secret key as an attribute into the credential without revealing it.

C Security Models

C.1 Security of Anonymous Credentials

We use the definition of [3] for a secure anonymous credential system. It requires unforgeability and anonymity (called anonymity and unlinkability there) and we augment it by selective disclosure. We call an (non-interactive) anonymous credential system *secure* if all subsequent properties hold.

Unforgeability: An anonymous credential system is called *unforgeable* if for all PPT adversaries $\mathcal{A}$ there is a negligible function ϵ such that

$$\mathsf{Pr}\left[\begin{array}{c}\mathsf{pp} \leftarrow \mathsf{Setup}(\kappa, t), (\mathsf{osk}, \mathsf{opk}) \leftarrow \mathsf{OrgKeyGen}(\mathsf{pp}), (\mathsf{state}, \mathbb{A}^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, \mathsf{opk}), \\ (\cdot, b^*) \leftarrow (\mathcal{A}(\mathsf{state}), \mathsf{Verify}(\mathsf{pp}, \mathsf{opk}, \mathbb{A}^*)) : b^* = \mathtt{true} \wedge \mathbb{A}^* \not\sqsubseteq \mathbb{A}_i \quad \forall i\end{array}\right] \leq \epsilon(\kappa)$$

where $\mathcal{A}$ has access to an oracle $\mathcal{O}$ that acts as the Issue algorithm and the values $\mathbb{A}_i$ are the attribute sets queried to $\mathcal{O}$.

Anonymity: An anonymous credential system is called *anonymous* if for all PPT adversaries $\mathcal{A}$ there is a negligible function ϵ such that

$$\mathsf{Pr}\left[(\mathsf{state}, \mathsf{pp}, \mathsf{opk}) \leftarrow \mathcal{A}(\kappa, t), b \xleftarrow{R} \{0,1\}, b^* \leftarrow \mathcal{A}^{\mathcal{O}_I, \mathcal{O}_V, \mathcal{O}_{LoR}}(\mathsf{state}) : b^* = b\right] \leq \frac{1}{2} + \epsilon(\kappa)$$

where $\mathcal{A}$ has access to oracle $\mathcal{O}_I$ once, which he queries with $\mathbb{A}$. $\mathcal{O}_I$ runs the Obtain protocol with $\mathcal{A}$ for two users 0 and 1 with respect to attributes $\mathbb{A}$. Furthermore, the adversary can query an

oracle $\mathcal{O}_V$ acting as a verifier for user 0 or user 1 as well as a left-or-right oracle $\mathcal{O}_{LoR}$, which plays the role of the Show algorithm on behalf of user b. Both latter oracles can be queried an arbitrary number of times. Note that in case of one-show credentials, $\mathcal{A}$ is restricted to query $\mathcal{O}_{LoR}$ *once* and is *not* allowed to query $\mathcal{O}_V$ for user 0 or 1.

Selective disclosure: For *selective disclosure*, we require that for any (computationally bounded) verifier distinct from the issuer it holds that: For any user i in possession of an honestly issued credential for $\mathbb{A}_i$, revealing $(\mathbb{A}'_i, \pi, \mathsf{cred}, \mathsf{upk}_i)$ for $\mathbb{A}'_i \sqsubseteq \mathbb{A}_i$ a polynomial number of times, reveals nothing about $\mathbb{A}_i \setminus \mathbb{A}'_i$.

C.2 Security of BDSS

Unforgeability: For all PPT adversaries $\mathcal{A}$ there exists a negligible function ϵ such that

$$\Pr\left[\begin{array}{c}\mathsf{pp} \leftarrow \mathsf{KeyGen}(\kappa, t), (\mathsf{dsk}_\mathsf{O}, \mathsf{dpk}_\mathsf{O}) \leftarrow \mathsf{DKeyGen}(\kappa), (\mathsf{dsk}_\mathsf{P}, \mathsf{dpk}_\mathsf{P}) \leftarrow \mathsf{DKeyGen}(\kappa), \\ ((\mathcal{T}^*, \sigma_{\mathcal{T}^*}, \mathsf{sk}_\mathsf{P}^{\mathcal{T}^*}), (\mathcal{M}^*, \sigma_{\mathcal{M}^*})) \leftarrow \mathcal{A}^{\mathcal{O}_\mathcal{T}, \mathcal{O}_\mathcal{M}}(\mathsf{pp}, \mathsf{dpk}_\mathsf{O}, \mathsf{dpk}_\mathsf{P}) : \\ (\mathsf{Verify}_\mathsf{T}(\mathcal{T}^*, \sigma_{\mathcal{T}^*}, \mathsf{dpk}_\mathsf{O}, \mathsf{sk}_\mathsf{P}^{\mathcal{T}^*}, \mathsf{dpk}_\mathsf{P}) = \mathtt{true} \ \wedge \ \mathcal{T}^* \notin Q_\mathcal{T}) \ \vee \\ (\mathsf{Verify}_\mathsf{M}(\mathcal{M}^*, \sigma_{\mathcal{M}^*}, \mathsf{dpk}_\mathsf{P}, \mathsf{dpk}_\mathsf{O}) = \mathtt{true} \ \wedge \ \mathcal{M}^* \notin Q_{\mathcal{M}^*})\end{array}\right] \leq \epsilon(\kappa)$$

where $Q_\mathcal{T}$ is the list of queried templates and $Q_{\mathcal{M}^*}$ is the list of queried instances for template $\mathcal{T}^*$. The adversary $\mathcal{A}$ has access to a template signing oracle $\mathcal{O}_\mathcal{T}$ issuing templates signatures for templates of its choice and an instance signing oracle $\mathcal{O}_\mathcal{M}$ issuing instance signatures for previously queried templates $\mathcal{T}_i$.

Immutability: For all PPT adversaries $\mathcal{A}$ there exists a negligible function ϵ such that

$$\Pr\left[\begin{array}{c}\mathsf{pp} \leftarrow \mathsf{KeyGen}(\kappa, t), (\mathsf{dsk}_\mathsf{O}, \mathsf{dpk}_\mathsf{O}) \leftarrow \mathsf{DKeyGen}(\kappa), (\mathsf{dsk}_\mathsf{P}, \mathsf{dpk}_\mathsf{P}) \leftarrow \mathsf{DKeyGen}(\kappa), \\ ((\mathcal{T}^*, \sigma_{\mathcal{T}^*}, \mathsf{sk}_\mathsf{P}^{\mathcal{T}^*}), (\mathcal{M}^*, \sigma_{\mathcal{M}^*})) \leftarrow \mathcal{A}^{\mathcal{O}_\mathcal{T}}(\mathsf{pp}, \mathsf{dpk}_\mathsf{O}, \mathsf{dpk}_\mathsf{P}, \mathsf{dsk}_\mathsf{P}) : \\ (\mathsf{Verify}_\mathsf{T}(\mathcal{T}^*, \sigma_{\mathcal{T}^*}, \mathsf{dpk}_\mathsf{O}, \mathsf{sk}_\mathsf{P}^{\mathcal{T}^*}, \mathsf{dpk}_\mathsf{P}) = \mathtt{true} \ \wedge \ \mathcal{T}^* \notin Q_\mathcal{T}) \ \vee \\ (\mathsf{Verify}_\mathsf{M}(\mathcal{M}^*, \sigma_{\mathcal{M}^*}, \mathsf{dpk}_\mathsf{P}, \mathsf{dpk}_\mathsf{O}) = \mathtt{true} \ \wedge \ \mathcal{M}^* \not\preceq \mathcal{T}^*)\end{array}\right] \leq \epsilon(\kappa)$$

where $Q_\mathcal{T}$ is the list of queried templates. The adversary $\mathcal{A}$ has access to a template signing oracle $\mathcal{O}_\mathcal{T}$ issuing templates signatures for templates of its choice and returning the respective template secret keys as well as an instance signing oracle $\mathcal{O}_\mathcal{M}$ issuing instance signatures for previously queried templates $\mathcal{T}_i$.

Privacy: For all PPT adversaries $\mathcal{A}$ there exists a negligible function ϵ such that

$$\Pr\left[\begin{array}{c}\mathsf{pp} \leftarrow \mathsf{KeyGen}(\kappa, t), (\mathsf{dsk}_\mathsf{O}, \mathsf{dpk}_\mathsf{O}) \leftarrow \mathsf{DKeyGen}(\kappa), (\mathsf{dsk}_\mathsf{P}, \mathsf{dpk}_\mathsf{P}) \leftarrow \mathsf{DKeyGen}(\kappa), \\ (\mathcal{T}_0, \mathcal{T}_1, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}_\mathcal{T}, \mathcal{O}_\mathcal{M}}(\mathsf{pp}, \mathsf{dpk}_\mathsf{O}, \mathsf{dpk}_\mathsf{P}), \\ (\mathcal{T}_b, \sigma_{\mathcal{T}_{b'}}) \leftarrow \mathcal{A}^{\mathcal{O}_\mathcal{T}, \mathcal{O}_\mathcal{M}, \mathcal{O}_{\mathcal{M}'}}(\mathsf{state}, \sigma_{\mathcal{T}_0}, \sigma_{\mathcal{T}_1}) : b = b'\end{array}\right] \leq \frac{1}{2} + \epsilon(\kappa)$$

where the adversary in the first phase has access to $\mathcal{O}_\mathcal{T}$ and $\mathcal{O}_\mathcal{M}$ as in the unforgeability definition and ends this phase by outputting two templates which allow instantiation of $k > 1$ equal messages. Then in the second phase the adversary gets the two templates signatures in a randomly permuted order, has access to $\mathcal{O}_\mathcal{T}$ and $\mathcal{O}_\mathcal{M}$ as in phase 1 (excluding $\mathcal{T}_0$ and $\mathcal{T}_1$) and can additionally query a modified instantiation oracle $\mathcal{O}_{\mathcal{M}'}$ at most k times for instances $\mathcal{M} \preceq \mathcal{T}_0, \mathcal{T}_1$ and obtains the output of Inst for both templates in a randomly permuted order.

C.3 Security of WHPSS

Unforgeability: For all PPT adversaries $\mathcal{A}$ there exists a negligible function ϵ such that

$$\mathsf{Pr}\left[\begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(\kappa, t), (\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{DKeyGen}(\kappa), \\ ((M,\sigma),(M,\sigma_p,\mathsf{pk}_i)) \leftarrow \mathcal{A}^{\mathcal{O}_R,\mathcal{O}_D,\mathcal{O}_E,\mathcal{O}_S,\mathcal{O}_{PS}}(\mathsf{pp},\mathsf{pk}_1) : \\ (\mathsf{DVerify}(\sigma, M, \mathsf{pk}_1) = \mathtt{true} \wedge M \notin Q_S) \ \vee \\ (\mathsf{PV}(\mathsf{pk}_i, M, \sigma_P) = \mathtt{true} \ \wedge \ i \neq 1 \ \wedge \ \mathsf{ID}(\sigma_P) = 1 \ \wedge \ M \notin Q_{PS}) \ \vee \\ (\mathsf{PV}(\mathsf{pk}_i, M, \sigma_P) = \mathtt{true} \ \wedge \ i = 1 \ \wedge \ \mathsf{ID}(\sigma_P) = 1 \ \wedge \ M \notin Q_{PS}) \ \vee \\ (\mathsf{PV}(\mathsf{pk}_i, M, \sigma_P) = \mathtt{true} \ \wedge \ i = 1 \ \wedge \ \forall \mathcal{M}_{\mathsf{ID}(\sigma_P)} \ M \notin \mathcal{M}_{\mathsf{ID}(\sigma_P)}) \end{array}\right] \leq \epsilon(\kappa)$$

where the adversary has access to an oracle $\mathcal{O}_R$ to register public keys for users, to $\mathcal{O}_D$ for the designation of signing rights for message spaces $\mathcal{M}$, to $\mathcal{O}_E$ which exposes the ℓ-th proxy signing key produced during self-delegation (user 1 to user 1), to $\mathcal{O}_S$ to obtain standard signatures from user 1 and to a proxy sign oracle $\mathcal{O}_{PS}$ for proxy signatures by user 1. Furthermore, Q_S and Q_{PS} are the lists of queried standard and proxy signatures, respectively. With σ we denote a standard signature.

Privacy: For all PPT adversaries $\mathcal{A}$ there exists a negligible function ϵ such that

$$\mathsf{Pr}\left[\begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(\kappa, t), (\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{DKeyGen}(\kappa), \\ ((i,c),\mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}_R,\mathcal{O}_D,\mathcal{O}_E,\mathcal{O}_S,\mathcal{O}_{PS}}(\mathsf{pp},\mathsf{pk}_1), \\ \mathsf{skp}^* = (\mathsf{D}(\mathcal{M}_r, \mathsf{pk}_1, \mathsf{sk}_1, i, \mathsf{pk}_i), \mathsf{P}(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{pk}_1)) \\ \mathcal{M}^* \leftarrow \mathcal{A}^{\mathcal{O}_R,\mathcal{O}_D,\mathcal{O}_E,\mathcal{O}_S,\mathcal{O}_{PS},\mathcal{O}_{PS'}}(\mathsf{state}) : \mathcal{M}^* = \mathcal{M}_r \end{array}\right] \leq \frac{1}{|\mathbb{M}'|} + \epsilon(\kappa)$$

where the adversary has access to the same oracles as in the unforgeability game in the first phase. The adversary then outputs $c > 1$ and i, a random warrant $\mathcal{M}_r$ of size $c+1$ is chosen from a message space $\mathbb{M}$ and a delegation of user i computed. Then, in the second phase the adversary has access to the same oracles as in phase 1 and an additional oracle $\mathcal{O}_{PS'}$ to obtain proxy signatures w.r.t. skp^* for randomly chosen messages from $\mathcal{M}_r$ at most c times. Note, that the adversary has no direct access to skp^* and thus is not aware of $\mathcal{M}_r$. The goal of the adversary is to guess the warrant $\mathcal{M}_r$ with negligible probability away from $\frac{1}{|\mathbb{M}'|}$ where $\mathbb{M}'$ represents the space of all potential messages $\mathbb{M}$ minus all message queried to $\mathcal{O}_{PS'}$.

D Proofs

D.1 Proof of Theorem 1

In the following, we sketch the proof which shows that an adversary $\mathcal{A}$ against the BDS constructed from AC can be turned into an adversary $\mathcal{B}$ against AC. We do not analyze the correctness of BDS here, since this can be verified by construction of the BDS approach from AC. The proof outline is as follows.

First we will show that if $\mathcal{A}$ breaks the immutability of BDS, $\mathcal{A}$ can be used to break the unforgeability of AC. We note that when constructing BDS from AC, security against immutability then implies security against unforgeability, since the only difference is that $\mathcal{A}$ does not get to know the secret key usk of the user (proxy) and, thus, immutability represents a stronger adversary in the same setting. Then we will show that if $\mathcal{A}$ breaks the privacy of BDS, $\mathcal{A}$ can be used to break the selective disclosure of AC.

We denote by $\mathcal{S}$ the challenger interacting with $\mathcal{A}$ ($\mathcal{A}$ and $\mathcal{S}$ form algorithm $\mathcal{B}$) in the respective BDS game and $\mathcal{S}$ interacts with the challenger $\mathcal{C}$ from the respective AC game.

Immutability: Initially, $\mathcal{C}$ runs $\mathsf{Setup}(\kappa, t)$ to obtain pp and $\mathsf{OrgKeyGen}(\mathsf{pp})$ to obtain $(\mathsf{osk}, \mathsf{opk})$. The latter is given to $\mathcal{S}$, who generates $(\mathsf{usk}, \mathsf{upk})$ and runs $\mathcal{A}$ on pp, opk and $(\mathsf{usk}, \mathsf{upk})$. Now, if $\mathcal{A}$ makes a call to the template signing oracle $\mathcal{O}_{\mathcal{T}}$ with template $\mathcal{T}$ and key upk, then $\mathcal{S}$ uses the query from $\mathcal{A}$, chooses ϕ, computes $\mathcal{T}^{enc}$, and engages in an issuing protocol by querying oracle $\mathcal{O}$ of $\mathcal{C}$ with attributes $\mathcal{T}^{enc}$. Then, $\mathcal{S}$ returns to $\mathcal{A}$ the cred and the template secret key ϕ as template signature. If $\mathcal{A}$ queries the signing oracle $\mathcal{O}_{\mathcal{M}}$ for an instance $\mathcal{M}$ of template $\mathcal{T}$, $\mathcal{S}$ executes the oracle as in the real game. If at some point $\mathcal{A}$ outputs either a valid forgery for a template signature $(\mathcal{T}^*, \mathsf{cred}^*, \mathsf{usk}^*)$ (defined as $(\mathcal{T}^*, \sigma_{\mathcal{T}^*}, \mathsf{sk}_{\mathsf{P}}^{\mathcal{T}^*})$ in the original game) or a valid forgery for an instance signature $(M^*, (\pi^*, \mathsf{cred}^*))$ (defined as $(\mathcal{M}^{enc*}, \sigma_{\mathcal{M}^*})$ in the original game), then we have three cases:

Case 1: $\mathcal{A}$ has broken the encoding $\mathcal{T}^{enc}$ which means that $\mathcal{S}$ can output a colliding pair of inputs for the hash function used in the encoding and aborts.

Case 2: Note that if $(\mathcal{T}^*, \mathsf{cred}^*, \mathsf{usk}^*)$ is a valid forgery then $\mathcal{T}^* \notin Q_{\mathcal{T}}$. Thus, $\mathcal{S}$ can save its state. Then, $\mathcal{C}$ runs $\mathcal{B}$ on state again and $\mathcal{S}$ then simply produces a showing w.r.t. $\mathcal{T}^*$ for cred^* using usk^*. Clearly, $\mathcal{B}$ wins the unforgeability game as no credential for $\mathcal{T}^{enc*}$ has been requested.

Case 3: The case for a valid forgery $(M^*, (\pi^*, \mathsf{cred}^*))$ is identical to case 2 with the only difference that $\mathcal{S}$ does not produce a showing but gives π^* as showing for M^* to $\mathcal{C}$. Clearly, $\mathcal{B}$ wins the unforgeability game as $\mathcal{M}^* \not\preceq \mathcal{T}^*$.

Privacy: Observe that if an adversary $\mathcal{A}$ is able to break the privacy property of the BDS, when given showings of a credential (instance signatures) and is, then, able to determine the full template (remaining attributes), this immediately gives an adversary $\mathcal{B}$ against the selective disclosure of the anonymous credential system. □

D.2 Proof of Theorem 2

In the following we sketch the proof which shows that an adversary $\mathcal{A}$ against the WHPS constructed from AC can be turned into an adversary $\mathcal{B}$ against AC. The proof outline is as follows.

First we will show that if $\mathcal{A}$ breaks the unforgeability of WHPS, $\mathcal{A}$ can be used to break the unforgeability of AC. Furthermore, if $\mathcal{A}$ breaks the privacy of WHPS, A can be used to break the selective disclosure of AC.

We denote by $\mathcal{S}$ the challenger interacting with $\mathcal{A}$ ($\mathcal{A}$ and $\mathcal{S}$ form algorithm $\mathcal{B}$) in the respective WHPS game and $\mathcal{S}$ interacts with the challenger $\mathcal{C}$ from the respective AC game.

Remarks on unforgeability: In the original definition of unforgeability for WHPS, users use the same keys to sign messages, delegations as well as issue proxy signatures. Consequently, the security model requires arbitrary delegations from users i to user 1, where i may be some user different from user 1. Furthermore, this also allows self delegation queries of user 1 to user 1, which does not match with the construction from anonymous credentials since $(\mathsf{sk}_1, \mathsf{pk}_1) = (\mathsf{osk}, \mathsf{opk})$, as this key pair does not represent a signing key pair of a standard signature scheme. Consequently, there is no direct self delegation, but user 1 would need another key pair of a digital signature scheme (and thus this is covered by the delegation to any other user i) and the oracle $\mathcal{O}_E$ is not

required. When constructing WHPS from anonymous credentials, we can assume that the keys used in the role of an originator (keys of an organization in the credential system) and the keys in the role of a proxy (keys of a standard digital signature scheme) are distinct. Consequently, we only consider delegations from user 1 to other users $i \neq 1$, which can be registered by an adversary. Furthermore, if we considered delegation from some user $i \neq 1$ to user 1, then we would just run another unforgeability game of the credential system with a new organization key pair for user i. In the WHPS security model it is assumed that the originator's key pair $(\mathsf{sk}_1, \mathsf{pk}_1)$ is from a standard signature scheme and, thus, $\mathcal{O}_S$ is a standard signing oracle . However, here we have $(\mathsf{sk}_1, \mathsf{pk}_1) = (\mathsf{osk}, \mathsf{opk})$, which does not support standard signature queries but only credential issuing queries. Consequently, the access to $\mathcal{O}_S$ as well as $\mathcal{O}_{PS}$ is not reasonable in this construction.

Taking all together, the forgeries remaining to be considered in the WHPS unforgeability game are those of type 4, i.e., $(\mathsf{PV}(\mathsf{pk}_i, M, \sigma_P) = \mathtt{true} \wedge i = 1 \wedge \forall \mathcal{M}_{\mathsf{ID}(\sigma_P)}\ M \notin \mathcal{M}_{\mathsf{ID}(\sigma_P)})$.

Unforgeability: Initially $\mathcal{C}$ runs $\mathsf{Setup}(\kappa, t)$ to obtain pp and $\mathsf{OrgKeyGen}(\mathsf{pp})$ to obtain $(\mathsf{osk}, \mathsf{opk})$ which is given to $\mathcal{S}$. Then, $\mathcal{S}$ runs $\mathcal{A}$ on pp and opk. Now, we need to discuss how the queries of $\mathcal{A}$ are simulated by $\mathcal{S}$ (note from the above discussion that $\mathcal{O}_E$ and $\mathcal{O}_S$ are not available):

$\mathcal{O}_R$**:** $\mathcal{S}$ obtains upk_i and simply stores it.
$\mathcal{O}_D$**:** If $\mathcal{A}$ requests a delegation for $\mathcal{M}$ for user i, then $\mathcal{S}$ forwards the communication between $\mathcal{A}$ and the issuing oracle of $\mathcal{C}$.

If at some point $\mathcal{A}$ outputs either a valid forgery $(M^*, (\pi^*, \mathsf{cred}^*), \mathsf{upk}^*)$ (defined as $(M, \sigma_p, \mathsf{pk}_i)$ in the original game), then we have two cases:

Case 1: $\mathcal{A}$ has broken the encoding $\mathcal{M}^{enc}$ which means that $\mathcal{S}$ can output a colliding pair of inputs for the hash function used in the encoding and aborts.

Case 2: $\mathcal{B}$ engages in a showing with $\mathcal{C}$ using $(M^*, (\pi^*, \mathsf{cred}^*), \mathsf{upk}^*)$ and wins the unforgeability game as no credential for $\mathcal{M}^*$ has been requested.

Privacy: Observe that if an adversary $\mathcal{A}$ is able to break the privacy property of WHPS, when given showings of a credential and is able to determine the unshown messages (attributes), this immediately gives an adversary $\mathcal{B}$ against the selective disclosure of the anonymous credential system. □