Privacy-Enhancing Proxy Signatures from Non-interactive Anonymous Credentials

David Derler, Christian Hanser, Daniel Slamanig

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 [25] and warrant-hiding proxy signatures [26], 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

[⋆] The authors have been supported by the European Commission through project FP7-FutureID, grant agreement number 318424. An extended version of this paper can be found at [19].

secure constructions). To restrict the delegation, Mambo et al. [27] 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 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 [23], 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 [26] (WHPS) as well as blank digital signatures [25] (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 [21] 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 [24]. 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. 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 [22], 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.

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 [21] 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.

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 `true` upon a valid showing and `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. For more formal security definitions, we refer the reader to the extended version [19].

Non-interactive anonymous credential systems: If interaction between the user and the verifier when executing $(\mathsf{Show}, \mathsf{Verify})$ 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. Further details are given in the extended version [19].

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 [25], whereas Section 4.2 discusses the Warrant-Hiding Proxy Signature Scheme (WHPSS) proposed in [26].

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 [25]:

$\mathsf{KeyGen}(\kappa, t)$**:** 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.

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

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

$\mathsf{Inst}(\mathcal{T}, \sigma_\mathcal{T}, \mathcal{M}, \mathsf{dsk}_\mathsf{P}, \mathsf{sk}_\mathsf{P}^\mathcal{T})$**:** 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 $\mathsf{sk}_\mathsf{P}^\mathcal{T}$ and the secret signing key of the proxy dsk_P, this algorithm outputs a signature $\sigma_\mathcal{M}$ for $\mathcal{M}$.

$\mathsf{Verify}_\mathsf{M}(\mathcal{M}, \sigma_\mathcal{M}, \mathsf{dpk}_\mathsf{O}, \mathsf{dpk}_\mathsf{P})$**:** Given an instance $\mathcal{M}$, an instance signature $\sigma_\mathcal{M}$ and the public verification keys of originator and proxy $(\mathsf{dpk}_\mathsf{O}, \mathsf{dpk}_\mathsf{P})$, 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 [25]. *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 $\mathsf{sk}_\mathsf{P}^\mathcal{T}$, 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 [29]). 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 the extended version [19].

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.

The security of a WHPSS is defined as follows [26]. *Correctness* requires that for all honestly computed parameters and for all proxy signing keys obtained by running the delegation algorithm, it holds that for all warrants and proxy signatures for a message M the verification algorithm for proxy signatures accepts a signature for M if M is in the warrant and rejects it otherwise. Furthermore, the proxy-identification algorithm 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. More formal definitions are provided in the extended version [19].

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 $H : \{0,1\}^* \times \{0,1\}^* \rightarrow \mathbb{Z}_p$ 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 [25], 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 $\mathsf{Enc}_{\mathcal{T}}^{\mathsf{BDS}}$.

For example, $\mathcal{T} = \{\{"\texttt{A}", "\texttt{B}"\}, "\texttt{declares to pay}", \{"\texttt{50\$}", "\texttt{100\$}"\}\}$, would yield a permuted and hashed sequence $(H("\texttt{100\$.}", 3), H("\texttt{50\$.}", 3), H("\texttt{ declares to pay }", 2), H("\texttt{A}", 1), H("\texttt{B}", 1))$.

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 $\mathsf{Enc}_{\mathcal{M}}^{\mathsf{BDS}}$. 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.

Choosing "$\texttt{B}$" and "$\texttt{50\$}$" in the example above, leads to an encoded message $\mathcal{M}^{enc} = (\mathsf{upk}_i, |\mathcal{T}|, (("\texttt{B}", 1, 5), ("\texttt{ declares to pay }", 2, 3), ("\texttt{50\$}", 3, 2), \pi)$.

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. A more detailed example of the encoding is given in [19].

We also emphasize that both, the encoding function $\mathsf{Enc}_{\mathcal{T}}^{\mathsf{BDS}}$ and the encoding function $\mathsf{Enc}_{\mathcal{M}}^{\mathsf{BDS}}$, 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

[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.

as template signature and the template-specific secret key ϕ for the proxy, are returned.

$\mathsf{Verify_T}(\mathcal{T}, \mathsf{cred}_i, \mathsf{opk}, (\mathsf{upk}_i, \mathsf{usk}_i), \phi)$: This algorithm computes $\mathcal{T}^{enc} \leftarrow \mathsf{Enc}^{\mathsf{BDS}}_{\mathcal{T}}(\mathcal{T}, \phi)$ 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.

$\mathsf{Verify_M}(\mathcal{M}^{enc}, (\pi, \mathsf{cred}_i), \mathsf{opk}, \mathsf{upk}_i)$: 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

The construction of WHPS from anonymous credentials is very similar to the BDS construction. Due to limited space the reader is referred to the extend version of this paper [19] for a detailed discussion.

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 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. The remaining properties of the schemes are related to the unforgeability of the anonymous credential scheme. In the extended version of this paper [19], we prove the following theorems:

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

Theorem 2. *If* AC *represents a secure anonymous credential system and the hash function used in the encoding* $\mathsf{Enc}^{\mathsf{WHPS}}_{\mathcal{M}}$ *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 $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ 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$(\mathcal{T}$, cred, $\mathcal{M}$, (opk, upk, usk), $\phi)$: Using $\mathcal{T}^{enc}$ and $\mathcal{M}^{enc}$, obtained by applying the encoding functions w.r.t. ϕ and compute $\mathsf{v}_x \leftarrow e(X, R), \mathsf{v}_{xy} \leftarrow e(X, B), \mathsf{v}_{(xy,i)} \leftarrow e(X, B_i), \mathsf{v}_s \leftarrow e(P, C)$,

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

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

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 [25, 26]. Moreover, we discuss the pros and

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

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$_{\mathsf{T}}$($\mathcal{T}, \mathsf{cred}, \mathsf{opk}, (\mathsf{upk}, \mathsf{usk}), (\phi, \alpha)$): 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, $\mathcal{M}$, $(\mathsf{opk}, \mathsf{upk}, \mathsf{usk}), (\phi, \alpha)$): 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$_{\mathsf{M}}$($\mathcal{M}^{enc}, (\pi, \mathsf{cred}), \mathsf{opk}, \mathsf{upk}$): 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

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

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

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

Table 1. BDSS/WHPSS efficiency comparison.

system. Furthermore, an anonymity feature, hiding the proxy's identity, could be obtained by skipping the proof part which links usk and upk ($\chi_{\mathsf{usk}} \cdot P = \mathsf{upk}$).

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, 28], 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, 20]). 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 Non-interactive 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. Derler, D., Hanser, C., Slamanig, D.: Privacy-Enhancing Proxy Signatures from Non-Interactive Anonymous Credentials. IACR ePrint 2014, 285 (2014)
20. Derler, D., Potzmader, K., Winter, J., Dietrich, K.: Anonymous Ticketing for NFC-Enabled Mobile Phones. In: INTRUST'11. LNCS, vol. 7222, pp. 66–83
21. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: CRYPTO'87. LNCS, vol. 263, pp. 186–194
22. 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
23. Fuchsbauer, G., Pointcheval, D.: Anonymous Proxy Signatures. In: SCN'08. LNCS, vol. 5229, pp. 201–217
24. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: EUROCRYPT'08. LNCS, vol. 4965, pp. 415–432
25. Hanser, C., Slamanig, D.: Blank Digital Signatures. In: ACM ASIACCS'13. pp. 95–106. ACM, ext.: IACR ePrint 2013/130
26. Hanser, C., Slamanig, D.: Warrant-Hiding Delegation-by-Certificate Proxy Signature Schemes. In: INDOCRYPT'13. LNCS, vol. 8250. Ext.: IACR ePrint 2013/544
27. Mambo, M., Usuda, K., Okamoto, E.: Proxy signatures for delegating signing operation. In: ACM CCS'96. pp. 48–57. ACM
28. Microsoft: U-Prove, http://research.microsoft.com/en-us/projects/u-prove
29. 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