Blank Digital Signatures*

Christian Hanser and Daniel Slamanig

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

Abstract. In this paper we present a novel type of digital signatures, which we call *blank digital signatures*. The basic idea behind this scheme is that an originator can define and sign a *message template*, describing fixed parts of a message as well as multiple choices for exchangeable parts of a message. One may think of a form with blank fields, where for such fields the originator specifies all the allowed strings to choose from. Then, a proxy is given the power to sign an *instantiation* of the template signed by the originator by using some secret information. By an instantiation, the proxy commits to one allowed choice per blank field in the template. The resulting *message signature* can be publicly verified under the originator's and the proxy's signature verification keys. Thereby, no verifying party except the originator and the proxy learn anything about the "unused" choices from the message template given a message signature. Consequently, the template is hidden from verifiers.
We discuss several applications, provide a formal definition of *blank digital signature schemes* and introduce a security model. Furthermore, we provide an efficient construction of such a blank digital signature scheme from any secure digital signature scheme, pairing-friendly elliptic curves and polynomial commitments, which we prove secure in our model. Finally, we outline several open issues and extensions for future work.

Keywords: Digital signature scheme, blank digital signatures, elliptic curves, pairings, polynomial commitments

1 Changes to the AsiaCCS'13 Proceedings Version

- The construction of the BDSS has been greatly simplified. This makes the construction much more intuitive, enhances the performance significantly and allows the use of an indistinguishability notion for the privacy property.
- The privacy property has been restated to an indistinguishability style security game.
- This version incorporates full security proofs and uses explicit reductions in the security proofs (whenever appropriate).

2 Introduction

Digital signatures provide the means to achieve source authentication and data integrity for digital messages in a publicly verifiable way meaning that at signing time a signer commits himself to a concrete message. In this paper, we propose the novel concept of a *blank digital signature scheme*. Here, an originator can define and sign a *message template*, describing fixed parts of a message as well as several choices for exchangeable parts of a message (one may think of a form with blank fields, where for such fields the originator specifies all the allowed strings to choose from), for which he can delegate signing permissions to a proxy. This proxy is given the power to sign *template instantiations* of the template given by the originator by using some secret information. The resulting *message signature* can be publicly verified under the originator's and the proxy's signature verification keys. Thereby, no verifying party except the originator and the proxy learn anything about the "unused" choices from the message template and, consequently, about the template when given a message

* This is a major revision of the version of the AsiaCCS'13 paper.

signature. In order to construct such a scheme it is helpful to look at existing variants of digital signature schemes to figure out, whether they can be used to instantiate blank digital signatures.

Conventional digital signatures require the signer to be available during signature creation, e.g., when a contract is signed. To overcome this limitation, the concept of proxy signatures [14] has been introduced. Basically, a proxy signature scheme allows an entity (the delegator) to delegate his signing capabilities to another entity (the proxy) that can then construct signatures on behalf of the delegator. This concept has seen a considerable amount of interest since then [5]. Surprisingly, only quite recently a suitable security model for proxy signatures has been introduced [4], and been extended to multi-level and identity-based proxy signature schemes later on [17]. Since in a practical application, the delegator may not want to give the proxy the power to sign *any* message on behalf of the signer, the *delegation by warrant* [14] approach was proposed. Here, a signed *warrant* is used to describe the delegation. Thereby, any type of security policy may be included in the warrant to describe the restrictions under which the delegation is valid. This approach seems to be particularly attractive and received the most attention, since the designator can define a message space for which he delegates his signing rights. In state of the art schemes [17,5], a *warrant* consists of the description ω of the message space for which signing is being delegated, together with a "certificate", which is a signature on ω under the delegators private signing key. We are given a similar requirement and, consequently, could ask whether proxy signatures can be used in this setting. In proxy signatures, this warrant is an abstract description, which could, for instance, be a context-free grammar, a regular expression, or as in [4], the description of a polynomial-time Turing machine computing the characteristic function of all potential messages, i.e., given a message to decide, whether the message is covered by ω or not. However, in proxy signatures the proxy is allowed to sign arbitrary messages from this abstract message space with the downside that the verifier learns the entire message space. Consequently, our requirement that the proxy can sign instantiations of a template without a verifier learning the corresponding template can not be realized by using existing proxy signature schemes.

Conventional digital signature schemes do not allow alterations of a signed document without invalidating the signature. Since it may be valuable to have the possibility to replace or remove (specified) parts of a message after signature creation such that the original signature stays valid (and no interaction with the original signer is required), redactable [19,11] as well as sanitizable signature schemes [1] have been introduced. Signature schemes, which allow *removal* of content (replacement by some special symbol $\perp$) by *any* party are called redactable [19,11], while signature schemes allowing (arbitrary) *replacements* of *admissible* parts by a *designated* party are called sanitizable signature schemes [1], cf. [16] for a comparison. As in our setting, the proxy should be allowed to choose from a list of predefined replacements for designated parts of the message, one could ask whether redactable or sanitizable signatures can be used in this setting. Since in redactable signature schemes any party is allowed to modify signed messages by removing message parts, such signature schemes are obviously not compatible with our requirements. The original concept of sanitizable signatures [1] allows designated sanitizers to replace designated parts of a message. However, here the sanitizer does not have the role of a proxy meaning that it does not sign the modified message. Furthermore, a sanitizer can replace the designated parts with arbitrary strings, which is clearly not meeting our requirements. The concept of sanitizable signatures was later on extended to allow only permitted replacements [13], yet, the Bloom filter [3] based construction does not meet cryptographic security requirements and the cryptographic accumulator [2] based approach [13,7] allows to securely restrict replacements. Yet, both approaches are not designed and also do not support the hiding of the set of accumulated values (allowed replacements) and, thus, are not suitable for our construction.

To sum this up, our concept has more in common with proxy signatures than with sanitizable signatures. This is due to our requirements that the signature of the originator is not publicly verifiable as it is the case in sanitizable signatures and only instantiations can be publicly verified as it is the case for proxy signatures.

2.1 Contribution

Since, however, none of the existing concepts covers all our requirements, we propose the novel concept of a *blank digital signature scheme*. Here, an originator, i.e., the signer delegating signing

permissions, can define and sign a *message template*, describing fixed parts of a message as well as several choices for exchangeable parts of a message. One may think of a form with blank fields, where for such fields the originator specifies all the allowed strings to choose from. Then, a proxy is given the power to sign *template instantiations* of the template given by the originator by using some secret information. The resulting *message signature* can be publicly verified under the originator's and the proxy's signature verification keys. Thereby, no verifying party except the originator and the proxy learn anything about the "unused" choices from the message template and, consequently, about the template given a message signature. Since this setting is quite different from the security requirements of proxy signatures and sanitizable signatures, most importantly, the template should be hidden from verifiers, we define a novel type of signature scheme along with a suitable security model. Similar to proxy signatures and sanitizable signatures, we require a public key infrastructure meaning that the originator and proxy are in possession of authentic signing keys. Moreover, since we use polynomial commitments in our construction, we need the parameters to be generated by a trusted third party.

A naive approach to realize blank digital signatures is that the originator produces n signatures for all n possible instantiations together with the public key of the proxy using a standard digital signature scheme, whereas the proxy simply signs the originator's signature for the chosen instantiations. However, the number of signatures issued by the originator would then be $O(n)$, which gets impractical very soon with increasing number of choices in exchangeable parts. By using randomized Merkle hash trees [15] as in redactable signatures, the number of signatures of the originator could be reduced to $O(1)$, whereas the signature of the proxy would then, however, be of size $O(\log n)$. At first glance, this may seem attractive, yet in Section 6.3 we illustrate that this approach also becomes soon impractical with an increasing number of choices. In our construction, the number of signatures of the originator is $O(1)$, whereas the size of both signatures, of the originator and the proxy, are also $O(1)$ and, in particular, very small and constant. Clearly, this is far more appealing than the aforementioned naive approaches.

2.2 Outline

In Section 3, we sketch some application scenarios for blank digital signatures. Section 4 discusses the mathematical and cryptographic preliminaries. Then, in Section 5 we introduce the notion of blank digital signatures and the corresponding security model. A construction of a blank digital signature scheme along with its security proof and a comparison to the naive approaches are given in Section 6. Finally, Section 7 concludes the paper and lists open issues for future work.

3 Applications

Here, we sketch some application scenarios which we envision for this novel type of digital signatures.

Partially blank signed contracts: Suppose a person is willing to sign a contract under certain predefined conditions, e.g., set of potential prices, range of possible contract dates, but is not able to sign the contract in person. Then, this person can elegantly delegate this task to another semi-trusted party, e.g., his attorney, by using blank digital signatures. The third party is then able to conclude the contract on behalf of his client. The client can do so by defining a contract template thereby leaving certain positions "blank", i.e., defining certain potential choices for the position without committing to one, and signing the template. Then, at a later point in time, the attorney is able to "fill in the gaps" by choosing from predefined choices, whereas the original signature of the client remains valid, and then signing the resulting contract as a proxy.

"Sanitizable" signatures: We may interpret exchangeable parts of message templates as replacements (with a potentially empty string) and, thus, can achieve a scheme with similar capabilities, but different meaning and strength as a non-interactive publicly accountable sanitizable signature

scheme [6][1], which supports controlled replacements [7,13]. Note that such a sanitizable signature scheme does not yet exist. However, there are some key differences. In contrast to sanitizable signatures, our template signature is not intended to be publicly verifiable, i.e., can only be verified by the proxy and, thus, the originator does not commit to a concrete instantiation of the template. Furthermore, in blank digital signatures, the allowed replacements are hidden, which is not supported by sanitizable signatures allowing such replacements [7,13]. Consequently, blank digital signatures may be seen as signature schemes supporting sanitizing capabilities, but are a different concept as it is clear from the differences mentioned above.

4 Preliminaries

In this section we firstly provide an overview of required mathematical and cryptographic preliminaries.

4.1 Mathematical Background

An elliptic curve over the finite field $\mathbb{F}_q$ is a plane, smooth curve described by the Weierstrass equation:

$$E : Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6, \tag{1}$$

where $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}_q$. The set $E(\mathbb{F}_q)$ of points $(x, y) \in \mathbb{F}_q^2$ satisfying Equation (1) plus the point at infinity ∞, which is the neutral element, forms an additive Abelian group, whereas the group law is determined by the *chord-and-tangent* method [18].

Let G be a cyclic group and p be a divisor of its group order, then there exists a subgroup of order p, which we subsequently denote by $G[p]$.

Definition 1 (Bilinear Map). Let G_1, G_2 and G_T be three cyclic groups of the same prime order p, where G_1, G_2 are additive groups and G_T is a multiplicative group. We call the map $e : G_1 \times G_2 \to G_T$ a *bilinear map* or *pairing*, if the following conditions hold:

Bilinearity: For all $P_1, P_2 \in G_1$ and $P_1', P_2' \in G_2$ we have:
- $e(P_1 + P_2, P') = e(P_1, P') \cdot e(P_2, P')$ for all $P' \in G_2$,
- $e(P, P_1' + P_2') = e(P, P_1') \cdot e(P, P_2')$ for all $P \in G_1$.

Non-degeneracy: If P is a generator of G_1 and P' a generator of G_2, then $e(P, P')$ is a generator of G_T, i.e., $e(P, P') \neq 1_{G_T}$.

Efficiently computable: e can be computed efficiently.

If $G_1 = G_2$, then e is called *symmetric* and *asymmetric* otherwise. The former type is also called *Type-1* pairing, whereas in case of the latter we distinguish between *Type-2* and *Type-3* pairings. For Type-2 pairings there is an efficiently computable isomorphism $\Psi : G_2 \to G_1$ [8] and for Type-3 pairings such an efficiently computable isomorphism does not exist. In our setting, G_1 and G_2 are p-order elliptic curve group over $\mathbb{F}_q$ and $G_T = \mathbb{F}_{q^k}^*[p]$, which is an order p subgroup of $\mathbb{F}_{q^k}^*$. Note that k, the so called *embedding degree*, is defined as $k = \min\{\ell \in \mathbb{N} : p \mid q^\ell - 1\}$.

A function $\epsilon : \mathbb{N} \to \mathbb{R}^+$ is called *negligible* if for all $c > 0$ there is a k_0 such that $e(k) < 1/k^c$ for all $k > k_0$. In the remainder of this paper, we use ϵ to denote such a negligible function.

Definition 2 (Discrete Logaritm Problem (DLP)). Let p be a prime of bitlength κ, G be a group of order p and $\alpha \in_R \mathbb{Z}_p^*$. Then, for every PPT adversary $\mathcal{A}$

$$\Pr\left(\mathcal{A}(P, \alpha P) = \alpha\right) = \epsilon(\kappa).$$

[1] In such a sanitizable signature scheme, when given a signature anybody can verify, whether a modification has been conducted by the original signer or the sanitizer without interacting with any party.

If G is an elliptic curve group, we call the corresponding DLP *elliptic curve discrete logarithm problem (ECDLP).*

Definition 3 (t-Strong Diffie Hellman Assumption (t-SDH)). Let p be a prime of bitlength κ, G be a group of order p, $\alpha \in_R \mathbb{Z}_p^*$ and let $(P, \alpha P, \alpha^2 P, \ldots, \alpha^t P) \in G^{t+1}$ for some $t > 0$. Then, for every PPT adversary $\mathcal{A}$

$$\Pr\left(\mathcal{A}(P, \alpha P, \alpha^2 P, \ldots, \alpha^t P) = \left(c, \frac{1}{\alpha + c} P\right)\right) = \epsilon(\kappa)$$

for any $c \in \mathbb{Z}_p \setminus \{-\alpha\}$.

4.2 Digital Signatures

Here, we briefly recall the definition of a standard digital signature scheme.

Definition 4 (Digital Signature Scheme). A digital signature scheme DSS is a tuple ($\mathsf{DKeyGen}$, DSign, $\mathsf{DVerify}$) of polynomial-time algorithms:

$\mathsf{DKeyGen}(\kappa)$: Is a key generation algorithm that takes as input a security parameter $\kappa \in \mathbb{N}$ and outputs a private (signing) key dsk and a public (verification) key dpk.

$\mathsf{DSign}(M, \mathsf{dsk})$: Is a (probabilistic) algorithm taking input a message $M \in \{0,1\}^*$, a private key dsk and outputs a signature σ.

$\mathsf{DVerify}(\sigma, M, \mathsf{dpk})$: Is a deterministic algorithm taking input a signature σ, a message $M \in \{0,1\}^*$, a public key dpk and outputs a single bit $b \in \{\texttt{true}, \texttt{false}\}$ indicating whether σ is a valid signature for M.

Furthermore, we require the digital signature scheme to be correct, i.e., for all $(\mathsf{dsk}, \mathsf{dpk}) \in \mathsf{KeyGen}(\kappa)$ and all $M \in \{0,1\}^*$, $\mathsf{DVerify}(\mathsf{DSign}(M, \mathsf{dsk}), M, \mathsf{dpk}) = \texttt{true}$ must hold. A digital signature scheme is secure, if it is existentially unforgeable under adaptively chosen-message attacks ($\mathsf{UF\text{-}CMA}$) [10]. Note that in practice, the sign and verify algorithms will typically use a hash function to map input messages to constant size strings, which is also known as the *hash-then-sign paradigm.*

4.3 Polynomial Commitments

In [12], Kate et al. introduced the notion of constant-size polynomial commitments. The authors present two distinct commitment schemes, whereas one is computationally hiding ($\mathsf{PolyCommit_{DL}}$) and the other is unconditionally hiding ($\mathsf{PolyCommit_{Ped}}$). For our scheme, we are using an unconditionally hiding and computationally binding variant of $\mathsf{PolyCommit_{DL}}$ for monic polynomials. The constructions of [12] use an algebraic property of polynomials $f(X) \in \mathbb{Z}_p[X]$. Namely, that $(X - \lambda)$ perfectly divides the polynomial $f(X) - f(\lambda)$ for $\lambda \in \mathbb{Z}_p$: Now, we briefly present the $\mathsf{PolyCommit_{DL}}$ construction of [12].

$\mathsf{Setup}(\kappa, t)$**:** Pick two groups G, G_T of the same prime order p (with p being a prime of bitlength κ) having a symmetric pairing $e : G \times G \to G_T$ such that the t-SDH assumption holds. Choose two generators $P \in G$ and $\alpha \in_R \mathbb{Z}_p^*$ and output $\mathsf{ppk} = (G, G_T, p, e, P, \alpha P, \ldots, \alpha^t P)$ as well as $\mathsf{psk} = \alpha$.

$\mathsf{Commit}(\mathsf{ppk}, f(X))$**:** Given $f(X) \in \mathbb{Z}_p[X]$ with $\deg(f) \leq t$ and compute the commitment $\mathcal{C} = f(\alpha)P \in G$ and output $\mathcal{C}$.

$\mathsf{Open}(\mathsf{ppk}, \mathcal{C}, f(X))$**:** Output $f(X)$.

$\mathsf{Verify}(\mathsf{ppk}, \mathcal{C}, f(X))$**:** Verify whether

$$\mathcal{C} = f(\alpha)P$$

holds and output $\texttt{true}$ on success and $\texttt{false}$ otherwise. [2]

[2] Subsequently, we use $f(\alpha)P$ as short-hand notation for $\sum_{i=0}^{\deg(f)} f^{(i)}(\alpha^i P)$, although $f(\alpha)$ as such can only be evaluated in case of a trapdoor commitment, i.e., with α known.

CreateWit(ppk, $f(X), \gamma$): Compute $\phi(X) = \frac{f(X)-f(\gamma)}{X-\gamma}$ and $W_\gamma = \phi(\alpha)P$ and output $(\gamma, f(\gamma), W_\gamma)$.
VerifyWit(ppk, $\mathcal{C}, \gamma, f(\gamma), W_\gamma$): Verify that $f(\gamma)$ is the evaluation of unknown f at point γ. This is done by checking whether

$$e(\mathcal{C}, P) = e(W_\gamma, \alpha P - \gamma P) \cdot e(f(\gamma)P, P)$$

holds. Output `true` on success and `false` otherwise.

A polynomial commitment scheme is *secure* if it is correct, polynomial binding, evaluation binding and hiding (cf. [12]). The above scheme can be proven secure under the t-SDH assumption in G, as long as $t < \sqrt{2^\kappa}$. For the proof we refer the reader to [12]. Notice that α must remain unknown to the committer (and thus the setup has to be run by a trusted third party), since, otherwise, it would be a trapdoor commitment scheme.

Moreover, we note that we do not need the algorithms CreateWit and VerifyEval in our construction, since we do not need to prove valid evaluations of the polynomial $f(X)$ without revealing the polynomial itself.

5 Blank Digital Signatures

In this section we introduce the notion of blank digital signatures as well as the according security model. As a prerequisite we first need to introduce representations and encodings for message templates and template instantiations.

5.1 Template and Message Representation

In the following we introduce a representation for message templates. A message template $\mathcal{T}$ describes all potential template instantiations that correspond to a single template. More formally, a message template is defined as follows.

Definition 5 (Message Template). A message template $\mathcal{T}$ is a sequence of non-empty sets $T_i = \{M_{i_1}, \ldots, M_{i_l}\}$ of bitstrings M_{i_j} and uniquely identified by $id_\mathcal{T}$. If the size of T_i is one, then the set T_i is called *fixed element* of $\mathcal{T}$ and *exchangeable element* otherwise. The set of all message templates is denoted by $\mathbb{T}$.

An exchangeable element T_i represents allowed substitutions, i.e., T_i can be replaced by any of its elements M_{i_j} in order to obtain an *instantiation* of the template. Let n be the sequence length of $\mathcal{T}$, then n is called *length of template* $\mathcal{T}$. Finally, with $|\mathcal{T}|$ we denote the *size of template* $\mathcal{T}$, that is $|\mathcal{T}| = \sum_{i=1}^{n} |T_i|$.

Definition 6 (Template Instantation). A *template instantiation* $\mathcal{M}$ of some template $\mathcal{T} = (T_i)_{i=1}^n$ is derived from $\mathcal{T}$ as follows. For each $1 \leq i \leq n$ choose exactly one element $M_i \in T_i$ and set $\mathcal{M} = (M_i)_{i=1}^n$. A template instantiation $\mathcal{M}$ is called *valid*, which we denote by $\mathcal{M} \preceq \mathcal{T}$, if it represents choices that were intended by the originator of template $\mathcal{T}$. Furthermore, we use $\mathbb{M}_\mathcal{T} = \{\mathcal{M} : \mathcal{M} \preceq \mathcal{T}\}$ to denote the set of all possible template instantiations of a template $\mathcal{T}$.

A message template $\mathcal{T}$ is called

- *trivial* if it does not contain any exchangeable elements. Note that this implies $|\mathcal{T}| = n$, and
- *minimal* if no two fixed elements are adjacent.

The *minimal* property guarantees that the number of fixed elements is kept minimal. The *complement* of the template instantiation $\mathcal{M}$ denoted as $\overline{\mathcal{M}}$ is a sequence of sets of bitstrings and represents all unused choices in the exchangeable elements, that is $\overline{\mathcal{M}} = (T_i \setminus \{M_i\})_{i=1}^n$ for an instantiation $\mathcal{M} = (M_i)_{i=1}^n$ of $\mathcal{T} = (T_i)_{i=1}^n$.

Now, we give a short example to illustrate our concept.

Example 1. Let $\mathcal{T} = (T_1, T_2, T_3)$ with

– $T_1 = \{$"I, hereby, declare to pay "$\}$,
– $T_2 = \{$"100$", "120$", "150$"$\}$ and
– $T_3 = \{$" for this tablet device."$\}$.

Here, T_1 and T_3 are fixed elements and T_2 is an exchangeable element with three choices. A template instantiation could, for instance, be

$$\mathcal{M} = (\text{"I, hereby, declare to pay "}, \text{"120\$"}, \text{" for this tablet device."}).$$

The complement of template instantiation $\mathcal{M}$ is then $\overline{\mathcal{M}} = (\emptyset, \{\text{"100\$"}, \text{"150\$"}\}, \emptyset)$.

In the following, we define encodings of templates and template instantiations, for which we use polynomials in the Euclidean ring $\mathbb{Z}_p[X]$. This allows us to perform polynomial division with remainder, which is essential to our construction.

Definition 7 (Template Encoding). Let $\mathcal{T}=(T_i)_{i=1}^n$ be a message template and $H : \{0,1\}^* \to \mathbb{Z}_p$ be a full-domain cryptographic hash function. A *template encoding function* $t : \mathbb{T} \to \mathbb{Z}_p[X]$ is defined as follows:

$$\mathcal{T} \mapsto \prod_{i=1}^{n} \prod_{M \in T_i} \big(X - H(M \| id_{\mathcal{T}} \| i)\big).$$

The evaluation $t(\mathcal{T})$ results in a so-called *template encoding polynomial* $t_{\mathcal{T}} \in \mathbb{Z}_p[X]$ of degree $|\mathcal{T}|$.

Note that the degree of the resulting polynomial needs to be bounded by $\sqrt{2^\kappa}$, as otherwise the security of the polynomial commitment scheme is no longer guaranteed. However, this has no impact in practice, since the polynomial can only be created by a polynomial time algorithm.

Definition 8 (Message Encoding). Similar to Definition 7, a *message encoding function* $m_{\mathcal{T}} : \mathbb{M}_{\mathcal{T}} \to \mathbb{Z}_p[X]$ with respect to a message template $\mathcal{T}$ is defined as follows:

$$\mathcal{M} \mapsto \prod_{i=1}^{n} (X - H(M_i \| id_{\mathcal{T}} \| i)),$$

where $\mathcal{M} = (M_i)_{i=1}^n$ is an instantiation of $\mathcal{T} = (T_i)_{i=1}^n$. We call $m_{\mathcal{M}} = m_{\mathcal{T}}(\mathcal{M}) \in \mathbb{Z}_p[X]$ *message encoding polynomial*. Furthermore, we define the *complementary message encoding function* $\overline{m}_{\mathcal{T}} : \mathbb{M}_{\mathcal{T}} \to \mathbb{Z}_p[X]$:

$$\mathcal{M} \mapsto \prod_{i=1}^{n} \prod_{M \in (T_i \setminus \{M_i\})} \big(X - H(M \| id_{\mathcal{T}} \| i)\big).$$

We call $m_{\overline{\mathcal{M}}} = \overline{m}_{\mathcal{T}}(\mathcal{M}) = (\frac{t_{\mathcal{T}}}{m_{\mathcal{M}}}) \in \mathbb{Z}_p[X]$ the *complementary message encoding polynomial* of $m_{\mathcal{M}}$ with respect to template $\mathcal{T}$.

In the following, we consider all polynomials to be expanded. To do so, we assume that an algorithm Exp, which carries out the polynomial expansion, is applied implicitly to all polynomials.

Typically, a template instantiation $\mathcal{M} = (M_i)_{i=1}^n \preceq \mathcal{T}$ will be mapped to a single bitstring $M = M_1 \| \ldots \| M_n$. We denote the mapping leading from M to $\mathcal{M}$ by $\lambda(M, \mathcal{I}) = \mathcal{M}$, where $\mathcal{I} = (|M_i|)_{i=1}^n$ is a descriptional sequence holding the lengths of the n elements of $\mathcal{M}$ as given by template $\mathcal{T}$. For sake of simplicity, we consider all templates to be non-trivial as well as minimal and do not differentiate between a template instantiation $\mathcal{M}$ and its corresponding bitstring M.

5.2 Blank Digital Signature Scheme

Now, we are able to formally define what we mean by a blank digital signature scheme.

Definition 9 (Blank Digital Signature Scheme). A *blank digital signature scheme* BDSS consists of a tuple $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}_{\mathcal{T}}, \mathsf{Inst}, \mathsf{Verify}_{\mathcal{M}})$ of polynomial-time algorithms:

$\mathsf{KeyGen}(\kappa, t)$**:** This probabilistic algorithm gets the security parameter $\kappa \in \mathbb{N}$ and a value $t \in \mathbb{N}$ specifying the maximum template size. It generates public parameters pp and returns them.

$\mathsf{Sign}(\mathcal{T}, \mathsf{pp}, \mathsf{dsk_O}, \mathsf{dpk_P})$**:** This probabilistic algorithm takes a message template $\mathcal{T}$, the public parameters pp, the originator's signing key $\mathsf{dsk_O}$, the proxy's verification key $\mathsf{dpk_P}$ and outputs a template signature $\sigma_{\mathcal{T}}$ and a template dependent private key for the proxy $\mathsf{sk_P^{\mathcal{T}}}$.

$\mathsf{Verify}_{\mathcal{T}}(\mathcal{T}, \sigma_{\mathcal{T}}, \mathsf{pp}, \mathsf{dpk_O}, \mathsf{sk_P^{\mathcal{T}}}, \mathsf{dpk_P})$**:** This deterministic algorithm takes a template $\mathcal{T}$, its signature $\sigma_{\mathcal{T}}$, the public parameters pp, the originator's signature verification key $\mathsf{dpk_O}$, the private key of the proxy $\mathsf{sk_P^{\mathcal{T}}}$, and the proxy's verification key $\mathsf{dpk_P}$. It outputs a bit $b \in \{\mathtt{true}, \mathtt{false}\}$ indicating whether $\sigma_{\mathcal{T}}$ is a valid signature for $\mathcal{T}$.

$\mathsf{Inst}(\mathcal{T}, \mathcal{M}, \sigma_{\mathcal{T}}, \mathsf{pp}, \mathsf{sk_P^{\mathcal{T}}}, \mathsf{dsk_P})$**:** This probabilistic algorithm takes a template $\mathcal{T}$, an instantiation $\mathcal{M}$, a template signature $\sigma_{\mathcal{T}}$, the public parameters pp, the private key of the proxy $\mathsf{sk_P^{\mathcal{T}}}$, the signing key $\mathsf{dsk_P}$ of the proxy and outputs a message signature $\sigma_{\mathcal{M}}$.

$\mathsf{Verify}_{\mathcal{M}}(\mathcal{M}, \sigma_{\mathcal{M}}, \mathsf{pp}, \mathsf{dpk_P}, \mathsf{dpk_O})$**:** This deterministic algorithm takes a template instantiation $\mathcal{M}$ of $\mathcal{T}$, the signature $\sigma_{\mathcal{M}}$, the public parameters pp, the signature verification key of the proxy $\mathsf{dpk_P}$, and the signature verification key of the originator $\mathsf{dpk_O}$. It outputs a bit $b \in \{\mathtt{true}, \mathtt{false}\}$ indicating whether $\sigma_{\mathcal{M}}$ is a valid signature for $\mathcal{M} \preceq \mathcal{T}$.

5.3 Security Definitions

In the following, we define the security properties a blank digital signature scheme needs to satisfy in order to be secure. Therefore, we start with a brief overview of the required properties.

Correctness: The scheme must be correct in terms of signature correctness, signature soundness and instantiation correctness, i.e., both template and message signatures are accepted when valid and template signatures of the originator are binding.

Unforgeability: No entity without knowledge of the signing keys $\mathsf{dsk_O}, \mathsf{dsk_P}$ and $\mathsf{sk_P^{\mathcal{T}}}$ should be able to forge template or message signatures. This is analogous to the security of traditional digital signatures.

Immutability: The proxy having access to $\mathsf{sk_P^{\mathcal{T}}}$ and $\mathsf{dsk_P}$, when given a template signature $\sigma_{\mathcal{T}}$ for template $\mathcal{T}$ should not be able to forge template or message signatures.

Privacy: No entity without knowledge of the signing keys $\mathsf{dsk_O}, \mathsf{dsk_P}$ and $\mathsf{sk_P^{\mathcal{T}}}$ should be able to determine elements of templates, which have not so far been revealed through instantiations.

Subsequently, the security definitions are discussed in more detail.

Correctness For a blank digital signature scheme the usual correctness properties are required to hold, i.e., genuinely signed templates and message signatures are accepted. Furthermore, we require template signatures to be sound, i.e., the originator commits to exactly one template by creating a template signature.

Signature correctness: For any key pairs $(\mathsf{dsk_O}, \mathsf{dpk_O}) \in \mathsf{DKeyGen}(\kappa)$ and $(\mathsf{dsk_P}, \mathsf{dpk_P}) \in \mathsf{DKeyGen}(\kappa)$, any BDSS parameters $\mathsf{pp} \in \mathsf{KeyGen}(\kappa, \mathsf{t})$, any template $\mathcal{T}$ and any honestly computed template signature

$$\sigma_{\mathcal{T}} = \mathsf{Sign}(\mathcal{T}, \mathsf{pp}, \mathsf{dsk_O}, \mathsf{dpk_P}),$$

we require that the verification

$$\mathsf{Verify}_{\mathcal{T}}(\mathcal{T}, \sigma_{\mathcal{T}}, \mathsf{pp}, \mathsf{dpk_O}, \mathsf{sk_P^{\mathcal{T}}}, \mathsf{dpk_P}) = \mathtt{true}$$

holds.

Signature soundness: For any key pairs $(\mathsf{dsk_O}, \mathsf{dpk_O}) \in \mathsf{DKeyGen}(\kappa)$ and $(\mathsf{dsk_P}, \mathsf{dpk_P}) \in \mathsf{DKeyGen}(\kappa)$, any BDSS parameters $\mathsf{pp} \in \mathsf{KeyGen}(\kappa, \mathsf{t})$, any template $\mathcal{T}$ and any honestly computed template signature

$$\sigma_{\mathcal{T}} = \mathsf{Sign}(\mathcal{T}, \mathsf{pp}, \mathsf{dsk_O}, \mathsf{dpk_P}),$$

we require that for any $(sk_P^{\mathcal{T}^*}, \mathcal{T}^*) \neq (sk_P^{\mathcal{T}}, \mathcal{T})$ the probability that the verification

$$\mathsf{Verify}_{\mathcal{T}}(\mathcal{T}^*, \sigma_{\mathcal{T}}, \mathsf{pp}, \mathsf{dpk_O}, \mathsf{sk_P^{\mathcal{T}^*}}, \mathsf{dpk_P}) = \mathtt{true}$$

holds is negligibly small as a function of the security parameter κ.

Instantiation correctness: For any key pairs $(\mathsf{dsk_O}, \mathsf{dpk_O}) \in \mathsf{DKeyGen}(\kappa)$ and $(\mathsf{dsk_P}, \mathsf{dpk_P}) \in \mathsf{DKeyGen}(\kappa)$, any BDSS parameters $\mathsf{pp} \in \mathsf{KeyGen}(\kappa, t)$, any template $\mathcal{T}$, any honestly computed signature $\sigma_{\mathcal{T}}$ and corresponding $\mathsf{sk_P^{\mathcal{T}}}$ such that

$$\mathsf{Verify}_{\mathcal{T}}(\mathcal{T}, \sigma_{\mathcal{T}}, \mathsf{pp}, \mathsf{dpk_O}, \mathsf{sk_P^{\mathcal{T}}}, \mathsf{dpk_P}) = \mathtt{true},$$

any honestly computed message signature

$$\sigma_{\mathcal{M}} = \mathsf{Inst}(\mathcal{T}, \mathcal{M}, \sigma_{\mathcal{T}}, \mathsf{pp}, \mathsf{sk_P^{\mathcal{T}}}, \mathsf{dsk_P}),$$

we require that the verification

$$\mathsf{Verify}_{\mathcal{M}}(\mathcal{M}, \sigma_{\mathcal{M}}, \mathsf{pp}, \mathsf{dpk_P}, \mathsf{dpk_O}) = \mathtt{true}$$

holds.

Definition 10 (Correctness). A BDSS is correct if it satisfies signature correctness, signature soundness and instantiation correctness.

Unforgeability Unforgeability in the context of blank digital signatures resembles the notion of (existential) unforgeability against adaptive chosen message attacks (UF-CMA) in classic digital signature schemes. We adapt the classical notion to our setting in Game 1. Unforgeability is a protection against attacks mounted by parties not having access to any secret information. Here, the adversary obtains the public keys from the challenger in the setup phase. In the query phase, A has access to two signing oracles, a template signing and a message signing oracle.

Definition 11 (Unforgeability). A BDSS is called *unforgeable*, if for any polynomial-time algorithm A the probability of winning Game 1 is negligible as a function of security parameter κ.

Immutability Immutability guarantees that no malicious proxy can compute signatures for templates or template instantiations not intended by the signer. Immutability is similar to unforgeability, but, in contrast to unforgeability, immutability deals with malicious insiders (proxies).

The immutability game, as stated in Game 2, differs only slightly from the unforgeability game. Here, the adversary additionally obtains $\mathsf{dsk_P}$ from the challenger in the setup phase. In the query phase, A has access to two signing oracles, where the template signing oracle additionally returns the corresponding private key $\mathsf{sk_P^{\mathcal{T}_i}}$. In this game, A wins if he outputs valid forgeries of type **T1**, **T2** or **M1**, **M2b**.

Definition 12 (Immutability). A BDSS is called *immutable*, if for any polynomial-time algorithm A the probability of winning Game 2 is negligible as a function of security parameter κ.

Privacy Privacy captures that any verifier except for the originator and the proxy, which is given a signature for a template instantiation $\mathcal{M}$ of a non-trivial template $\mathcal{T}$, can not gain any information about $\overline{\mathcal{M}}$ and thereby learn about $\mathcal{T}$. This means that even if all but one choice of a single exchangeable element has been revealed no verifier should be able to gain complete knowledge of $\mathcal{T}$. This is formalized in Game 3.

Definition 13 (Privacy). A BDSS is called *private*, if for any polynomial-time algorithm A the probability of winning Game 3 is negligibly close to 1/2 as a function of security parameter κ and *unconditionally private*, if for any computationally unbounded algorithm A the probability of winning Game 3 is 1/2.

Setup: The challenger C runs $\mathsf{KeyGen}(\kappa, t)$ to obtain pp. Furthermore, C runs $\mathsf{DKeyGen}(\kappa)$ of a secure digital signature scheme twice to generate $(\mathsf{dsk_O}, \mathsf{dpk_O})$ and $(\mathsf{dsk_P}, \mathsf{dpk_P})$. It gives the adversary A the resulting public parameters and keys $\mathsf{pp}, \mathsf{dpk_O}$ and $\mathsf{dpk_P}$ and keeps the private keys $\mathsf{dsk_O}$ and $\mathsf{dsk_P}$ to itself.

Query: The adversary A has access to a template signing oracle $\mathcal{O}_{\mathcal{T}}$ and access to a message signing oracle $\mathcal{O}_{\mathcal{M}}$. Both oracles are simulated by the challenger C.

- On receiving a template signing query $\mathcal{T}_i$, C checks whether such a query has already been issued. If so, C returns $\sigma_{\mathcal{T}_i}$. Otherwise, C runs $\mathsf{Sign}(\mathcal{T}_i, \mathsf{pp}, \mathsf{dsk_O}, \mathsf{dpk_P})$ and returns $\sigma_{\mathcal{T}_i}$ and stores the so obtained $(\mathcal{T}_i, \mathsf{sk_P^{\mathcal{T}_i}}, \sigma_{\mathcal{T}_i})$.
- On receiving a message signing query $(\mathcal{T}_i, \mathcal{M}_{i_j})$, C checks, whether a template signing query for $\mathcal{T}_i$ has already been made and whether $\mathcal{M}_{i,j} \preceq \mathcal{T}_i$. If not, C returns $\perp$. Otherwise, C obtains $(\mathcal{T}_i, \mathsf{sk_P^{\mathcal{T}_i}}, \sigma_{\mathcal{T}_i})$, checks whether the query $(\mathcal{T}_i, \mathcal{M}_{i_j})$ has already been made. If so, C returns $\sigma_{\mathcal{M}_{i_j}}$. Otherwise, C runs $\mathsf{Inst}(\mathcal{T}_i, \mathcal{M}_{i_j}, \sigma_{\mathcal{T}_i}, \mathsf{pp}, \mathsf{sk_P^{\mathcal{T}_i}}, \mathsf{dsk_P})$, returns $\sigma_{\mathcal{M}_{i_j}}$ and stores the tuple $(\mathcal{T}_i, \mathcal{M}_{i_j}, \sigma_{\mathcal{M}_{i_j}})$.

All of these queries can be made adaptively.

Output: The adversary A outputs either a triple $(\mathcal{T}^*, \sigma_{\mathcal{T}^*}, \mathsf{sk_P^{\mathcal{T}^*}})$ or a pair $(\mathcal{M}^*, \sigma_{\mathcal{M}^*})$. A wins if either

T1 $\mathsf{Verify}_{\mathcal{T}}(\mathcal{T}^*, \sigma_{\mathcal{T}_i}, \mathsf{pp}, \mathsf{dpk_O}, \mathsf{sk_P^{\mathcal{T}_i}}, \mathsf{dpk_P}) = \mathtt{true}$, where $\mathcal{T}^*$ is an unqueried template and $\mathsf{sk_P^{\mathcal{T}_i}}$, and $\sigma_{\mathcal{T}_i}$ correspond to one queried template $\mathcal{T}_i$,

T2 $\mathsf{Verify}_{\mathcal{T}}(\mathcal{T}^*, \sigma_{\mathcal{T}^*}, \mathsf{pp}, \mathsf{dpk_O}, \mathsf{sk_P^{\mathcal{T}^*}}, \mathsf{dpk_P}) = \mathtt{true}$ for some unqueried template $\mathcal{T}^*$ with corresponding unqueried $\mathsf{sk_P^{\mathcal{T}^*}}$, and $\sigma_{\mathcal{T}^*}$,

M1 $\mathsf{Verify}_{\mathcal{M}}(\mathcal{M}^*, \sigma_{\mathcal{M}_{i_j}}, \mathsf{pp}, \mathsf{dpk_P}, \mathsf{dpk_O}) = \mathtt{true}$, where $\mathcal{M}^* \neq \mathcal{M}_{i_j}$ is an unqueried message and and $\sigma_{\mathcal{M}_{i_j}}$ corresponds to one queried $\mathcal{M}_{i_j} \preceq \mathcal{T}_i$, or

M2 $\mathsf{Verify}_{\mathcal{M}}(\mathcal{M}^*, \sigma_{\mathcal{M}^*}, \mathsf{pp}, \mathsf{dpk_P}, \mathsf{dpk_O}) = \mathtt{true}$ for some unqueried message $\mathcal{M}^*$ from template signature $\sigma_{\mathcal{T}_i}$ for some previously queried template $\mathcal{T}_i$, such that either

(a) $\mathcal{M} \preceq \mathcal{T}_i$, or

(b) $\mathcal{M} \not\preceq \mathcal{T}_i$.

Game 1: Unforgeability Game

Setup: The challenger C runs $\mathsf{KeyGen}(\kappa, t)$ to obtain pp. Furthermore, C runs $\mathsf{DKeyGen}(\kappa)$ of a secure digital signature scheme twice to generate $(\mathsf{dsk_O}, \mathsf{dpk_O})$ and $(\mathsf{dsk_P}, \mathsf{dpk_P})$. It gives the adversary A the resulting public parameters and keys $\mathsf{pp}, \mathsf{dpk_O}$ and $\mathsf{dpk_P}$ as well as $\mathsf{dsk_P}$ and keeps the private key $\mathsf{dsk_O}$ to itself.

Query: The adversary A has access to a template signing oracle $\mathcal{O}_{\mathcal{T}}$ and access to a message signing oracle $\mathcal{O}_{\mathcal{M}}$. Both oracles are simulated by the challenger C.

- On receiving a template signing query $\mathcal{T}_i$, C checks whether such a query has already been issued. If so, C returns $\sigma_{\mathcal{T}_i}$. Otherwise, C runs $\mathsf{Sign}(\mathcal{T}_i, \mathsf{pp}, \mathsf{dsk_O}, \mathsf{dpk_P})$ and returns $(\mathsf{sk_P^{\mathcal{T}_i}}, \sigma_{\mathcal{T}_i})$ and stores the so obtained $(\mathcal{T}_i, \mathsf{sk_P^{\mathcal{T}_i}}, \sigma_{\mathcal{T}_i})$.
- On receiving a message signing query $(\mathcal{T}_i, \mathcal{M}_{i_j})$, C checks, whether a template signing query for $\mathcal{T}_i$ has already been made and whether $\mathcal{M}_{i,j} \preceq \mathcal{T}_i$. If not, C returns $\perp$. Otherwise, C obtains $(\mathcal{T}_i, \mathsf{sk_P^{\mathcal{T}_i}}, \sigma_{\mathcal{T}_i})$, checks whether the query $(\mathcal{T}_i, \mathcal{M}_{i_j})$ has already been made. If so, C returns $\sigma_{\mathcal{M}_{i_j}}$. Otherwise, C runs $\mathsf{Inst}(\mathcal{T}_i, \mathcal{M}_{i_j}, \sigma_{\mathcal{T}_i}, \mathsf{pp}, \mathsf{sk_P^{\mathcal{T}_i}}, \mathsf{dsk_P})$, returns $\sigma_{\mathcal{M}_{i_j}}$ and stores the tuple $(\mathcal{T}_i, \mathcal{M}_{i_j}, \sigma_{\mathcal{M}_{i_j}})$.

All of these queries can be made adaptively.

Output: The adversary A outputs either a triple $(\mathcal{T}^*, \sigma_{\mathcal{T}^*}, \mathsf{sk_P^{\mathcal{T}^*}})$ or a pair $(\mathcal{M}^*, \sigma_{\mathcal{M}^*})$. A wins if either

T1 $\mathsf{Verify}_{\mathcal{T}}(\mathcal{T}^*, \sigma_{\mathcal{T}_i}, \mathsf{pp}, \mathsf{dpk_O}, \mathsf{sk_P^{\mathcal{T}_i}}, \mathsf{dpk_P}) = \mathtt{true}$, where $\mathcal{T}^*$ is an unqueried template and $\mathsf{sk_P^{\mathcal{T}_i}}$, and $\sigma_{\mathcal{T}_i}$ correspond to one queried template $\mathcal{T}_i$,

T2 $\mathsf{Verify}_{\mathcal{T}}(\mathcal{T}^*, \sigma_{\mathcal{T}^*}, \mathsf{pp}, \mathsf{dpk_O}, \mathsf{sk_P^{\mathcal{T}^*}}, \mathsf{dpk_P}) = \mathtt{true}$ for some unqueried template $\mathcal{T}^*$ with corresponding unqueried $\mathsf{sk_P^{\mathcal{T}^*}}$, and $\sigma_{\mathcal{T}^*}$,

M1 $\mathsf{Verify}_{\mathcal{M}}(\mathcal{M}^*, \sigma_{\mathcal{M}_{i_j}}, \mathsf{pp}, \mathsf{dpk_P}, \mathsf{dpk_O}) = \mathtt{true}$, where $\mathcal{M}^* \neq \mathcal{M}_{i_j}$ is an unqueried message and and $\sigma_{\mathcal{M}_{i_j}}$ corresponds to one queried $\mathcal{M}_{i_j} \preceq \mathcal{T}_i$, or

M2 $\mathsf{Verify}_{\mathcal{M}}(\mathcal{M}^*, \sigma_{\mathcal{M}^*}, \mathsf{pp}, \mathsf{dpk_P}, \mathsf{dpk_O}) = \mathtt{true}$ for some unqueried message $\mathcal{M}^*$ from template signature $\sigma_{\mathcal{T}_i}$ for some previously queried template $\mathcal{T}_i$, such that

(b) $\mathcal{M} \not\preceq \mathcal{T}_i$.

Game 2: Immutability Game

Setup: The challenger C runs $\mathsf{KeyGen}(\kappa, t)$ to obtain pp. Furthermore, C runs $\mathsf{DKeyGen}(\kappa)$ of a secure digital signature scheme twice to generate $(\mathsf{dsk_O}, \mathsf{dpk_O})$ and $(\mathsf{dsk_P}, \mathsf{dpk_P})$. It gives the adversary A the resulting public parameters and keys $\mathsf{pp}, \mathsf{dpk_O}$ and $\mathsf{dpk_P}$ and keeps the private keys $\mathsf{dsk_O}$ and $\mathsf{dsk_P}$ to itself.

Query 1: The adversary A has access to a template signing oracle $\mathcal{O}_{\mathcal{T}}$ and access to a message signing oracle $\mathcal{O}_{\mathcal{M}}$. Both oracles are simulated by the challenger C.

- On receiving a template signing query $\mathcal{T}_i$, C checks whether such a query has already been issued. If so, C returns $\sigma_{\mathcal{T}_i}$. Otherwise, C runs $\mathsf{Sign}(\mathcal{T}_i, \mathsf{pp}, \mathsf{dsk_O}, \mathsf{dpk_P})$ and returns $(\mathsf{sk_P^{\mathcal{T}_i}}, \sigma_{\mathcal{T}_i})$ and stores the so obtained $(\mathcal{T}_i, \mathsf{sk_P^{\mathcal{T}_i}}, \sigma_{\mathcal{T}_i})$.
- On receiving a message signing query $(\mathcal{T}_i, \mathcal{M}_{i_j})$, C checks, whether a template signing query for $\mathcal{T}_i$ has already been made and whether $\mathcal{M}_{i,j} \preceq \mathcal{T}_i$. If not, C returns $\perp$. Otherwise, C obtains $(\mathcal{T}_i, \mathsf{sk_P^{\mathcal{T}_i}}, \sigma_{\mathcal{T}_i})$, checks whether the query $(\mathcal{T}_i, \mathcal{M}_{i_j})$ has already been made. If so, C returns $\sigma_{\mathcal{M}_{i_j}}$. Otherwise, C runs $\mathsf{Inst}(\mathcal{T}_i, \mathcal{M}_{i_j}, \sigma_{\mathcal{T}_i}, \mathsf{pp}, \mathsf{sk_P^{\mathcal{T}_i}}, \mathsf{dsk_P})$, returns $\sigma_{\mathcal{M}_{i_j}}$ and stores the tuple $(\mathcal{T}_i, \mathcal{M}_{i_j}, \sigma_{\mathcal{M}_{i_j}})$.

All of these queries can be made adaptively.

Challenge: At some point A signals C that he is ready to be challenged by choosing two unqueried, distinct templates $\mathcal{T}_0'$ and $\mathcal{T}_1'$ of sizes less than t, which are shaped in such a way that from both templates k equal instantiations $\mathcal{M}_1', \ldots, \mathcal{M}_k'$ can be derived, and sending them to C. Then, C signs these two templates, stores the tuples $(sk_P^{\mathcal{T}_0'}, \sigma_{\mathcal{T}_0'})$, $(sk_P^{\mathcal{T}_1'}, \sigma_{\mathcal{T}_1'})$ and returns the corresponding template signatures $\sigma_{\mathcal{T}_0'}$ and $\sigma_{\mathcal{T}_1'}$ in a randomly permuted order to A.

Query 2: The adversary A is allowed to issue an arbitrary number of queries as in query phase 1, excluding templates $\mathcal{T}_0'$ and $\mathcal{T}_1'$.

- Additionally, A can send instantiation queries for $\mathcal{M}_l'$ with $1 \leq l \leq k$ to an additional oracle $\mathcal{O}'_{\mathcal{M}}$, which is simulated by C. On receiving such a query $\mathcal{M}_l'$, C checks whether such a query has already been made. If so, C retrieves $(\mathcal{M}_l', \sigma_{\mathcal{M}'_{0_l}}, \sigma_{\mathcal{M}'_{1_l}})$ and returns $\sigma_{\mathcal{M}'_{0_l}}$ and $\sigma_{\mathcal{M}'_{1_l}}$ in a randomly permuted order. Otherwise, C retrieves $(sk_P^{\mathcal{T}_b'}, \sigma_{\mathcal{T}_b'})$ and runs $\mathsf{Inst}(\mathcal{T}_b', \mathcal{M}'_{b_l}, \sigma_{\mathcal{T}_b'}, \mathsf{pp}, \mathsf{sk_P^{\mathcal{T}_b'}}, \mathsf{dsk_P})$ for $b = 0, 1$, stores $(\mathcal{M}_l', \sigma_{\mathcal{M}'_{0_l}}, \sigma_{\mathcal{M}'_{1_l}})$ and returns $\sigma_{\mathcal{M}'_{0_l}}$ and $\sigma_{\mathcal{M}'_{1_l}}$ in a randomly permuted order.

All of these queries can be made adaptively.

Output: The adversary A outputs $(\mathcal{T}_b, \sigma_{\mathcal{T}_{b'}})$ and wins if $b = b'$.

Game 3: Privacy Game

Security Now, we can define what constitutes a secure blank digital signature scheme.

Definition 14 (Security). We call a BDSS *secure*, if it is correct, unforgeable, immutable and (unconditionally) private.

6 Construction

In this section we detail our construction and prove its security, i.e., we show that our presented construction is correct, unforgeable, immutable and private.

6.1 Intuition

Before we present the detailed construction, we provide some intuition in order to make our design choices comprehensible. As already noted, we use standard digital signatures, such as ECDSA [9], as a building block assuming the respective signature keys to be available to every participant in an authentic fashion. Note that this requires the availability of public key infrastructures, which are, however, commonly used in practice today. In our construction, DSS signatures provide authenticity of template and message signatures.

As already discussed in Section 5, we use polynomials to represent templates and template instantiations. The intuition is that the originator commits to a template polynomial. By construction every allowed template instantiation is represented by a message encoding polynomial that perfectly

KeyGen: On input (κ, t), choose an elliptic curve $E(\mathbb{F}_q)$ with a subgroup of large prime order p generated by $P \in E(\mathbb{F}_q)[p]$, such that the bitlength of p is κ. Choose a pairing $e : E(\mathbb{F}_q)[p] \times E(\mathbb{F}_q)[p] \to \mathbb{F}_{q^k}^*[p]$ and a full-domain cryptographic hash function $H : \{0,1\}^* \to \mathbb{Z}_p$ for use with the encoding functions. Pick $\alpha \in_R \mathbb{Z}_p^*$, compute $(\alpha P, \ldots, \alpha^t P)$ and output $\mathsf{pp} = (H, E(\mathbb{F}_q), e, p, P, \alpha P, \ldots, \alpha^t P)$.

Sign: Given $\mathcal{T}, \mathsf{pp}, \mathsf{dsk_O}$ and $\mathsf{dpk_P}$, where $\mathcal{T}$ is a template of size $|\mathcal{T}| = \ell$ and length n with $\ell > n$, this algorithm picks a unique $id_{\mathcal{T}} \in_R \{0,1\}^\kappa$, computes $t_{\mathcal{T}} = t(\mathcal{T}) \in \mathbb{Z}_p[X]$, picks a secret $\rho \in_R \mathbb{Z}_p^*$ and computes

$$\mathcal{C} = e(\rho \cdot t_{\mathcal{T}}(\alpha)P, P) \quad \text{and} \quad \tau = \mathsf{DSign}(id_{\mathcal{T}} \| \mathcal{C} \| n \| \mathsf{dpk_P}, \mathsf{dsk_O})$$

and returns the template signature $\sigma_{\mathcal{T}} = (id_{\mathcal{T}}, \mathcal{C}, n, \tau)$ as well as $\mathsf{sk}_\mathsf{P}^{\mathcal{T}} = \rho$.

Verify$_{\mathcal{T}}$: Given $\mathcal{T}, \sigma_{\mathcal{T}}, \mathsf{pp}, \mathsf{dpk_O}, \mathsf{sk}_\mathsf{P}^{\mathcal{T}}$ and $\mathsf{dpk_P}$, where $\mathcal{T}$ is a template of size $|\mathcal{T}| = \ell$ and length n with $\ell > n$, this algorithm checks whether $|\mathcal{T}| \leq t$. If not, it returns `false`. Otherwise, it computes $t_{\mathcal{T}} = t(\mathcal{T})$ and checks whether

$$\mathsf{DVerify}(\tau, id_{\mathcal{T}} \| \mathcal{C} \| n \| \mathsf{dpk_P}, \mathsf{dpk_O}) = \texttt{true} \quad \wedge \quad e(\rho \cdot t_{\mathcal{T}}(\alpha)P, P) = \mathcal{C}.$$

If so, return `true` and `false` otherwise.

Inst: Given $\mathcal{T}, \mathcal{M}, \sigma_{\mathcal{T}}, \mathsf{pp}, \mathsf{sk}_\mathsf{P}^{\mathcal{T}}$ and $\mathsf{dsk_P}$, where $\mathcal{T}$ is a template of size $|\mathcal{T}| = \ell$ and length n with $\ell > n$, this algorithm computes $m_{\overline{\mathcal{M}}} = \overline{m}_{\mathcal{T}}(\mathcal{M}) \in \mathbb{Z}_p[X]$. Then, it computes

$$\mathcal{C}_{\overline{\mathcal{M}}} = \rho \cdot m_{\overline{\mathcal{M}}}(\alpha)P \quad \text{and} \quad \mu = \mathsf{DSign}(\tau \| \mathcal{C}_{\overline{\mathcal{M}}} \| \mathcal{I}, \mathsf{dsk_P}).$$

It returns $\sigma_{\mathcal{M}} = (\mu, \mathcal{C}_{\overline{\mathcal{M}}}, \mathcal{I}, \sigma_{\mathcal{T}})$.

Verify$_{\mathcal{M}}$: Given $\mathcal{M}, \sigma_{\mathcal{M}} = (\mu, \mathcal{C}_{\overline{\mathcal{M}}}, \mathcal{I} = (|M_i|)_{i=1}^n, \sigma_{\mathcal{T}})$, pp, $\mathsf{dpk_P}$ and $\mathsf{dpk_O}$ this algorithm verifies whether

$$\mathsf{DVerify}(\tau, id_{\mathcal{T}} \| \mathcal{C} \| n \| \mathsf{dpk_P}, \mathsf{dpk_O}) = \texttt{true} \ \wedge \ \mathsf{DVerify}(\mu, \tau \| \mathcal{C}_{\overline{\mathcal{M}}} \| \mathcal{I}, \mathsf{dpk_P}) = \texttt{true} \ \wedge$$

$$|\mathcal{I}| = n \quad \wedge \quad \sum_{i=1}^{n} |M_i| = |\mathcal{M}|$$

On failure return `false`, otherwise evaluate $m_{\mathcal{M}} = m_{\mathcal{T}}(\mathcal{M})$ and check whether

$$e(m_{\mathcal{M}}(\alpha)P, \mathcal{C}_{\overline{\mathcal{M}}}) = \mathcal{C}$$

On success return `true` and `false` otherwise.

Scheme 1: Blank Digital Signature Scheme

divides the template polynomial. A proxy can now commit to a message polynomial, by computing and signing a commitment to the complementary message encoding polynomial. However, he can not choose arbitrary divisors of the template polynomial, as the indexes of message elements are incorporated into the encoding and the length of the message, i.e., the degree of the message polynomial, is fixed by the originator.

In the verification, the verifier computes a commitment to the message polynomial and checks whether the computed commitment and the commitment given by the proxy relate to the commitment given by the originator. We need a trusted third party, as the originator should not know the value α. Otherwise, he could exchange the template polynomial after signature generation for another polynomial having the same evaluation at the point α. Note that in the context of polynomial commitments the setup must always be run by a trusted third party, as otherwise these commitments represent trapdoor commitments, i.e., the knowledge of α allows to open the commitment to arbitrary polynomials.

We use polynomial commitments of the form $\mathcal{C} = \rho \cdot f(\alpha)P$ for some random value $\rho \in_R \mathbb{Z}_p^*$ to hide the committed polynomials. This provides unconditional hiding of unknown factors of $f(X)$ as long as ρ stays unknown.

6.2 Scheme

In Scheme 1, we present the detailed construction of our proposed BDSS. Moreover, in Protocol 1, we illustrate a typical scenario for the interaction of the originator, the proxy and the verifier.

We assume that the originator as well as the proxy both own an authentic key pair for a secure digital signature scheme $(\mathsf{dsk_O}, \mathsf{dpk_O})$ and $(\mathsf{dsk_P}, \mathsf{dpk_P})$, respectively.

$\mathsf{Setup_T}$**:** The trusted third party T chooses a suitable security parameter κ and a value $t \in \mathbb{N}$ representing the maximum template length, runs $\mathsf{KeyGen}(\kappa, t)$ and publishes pp in an authentic fashion.

$\mathsf{Issue_O}$**:** O defines a message template $\mathcal{T}$, runs $\mathsf{Sign}(\mathcal{T}, \mathsf{pp}, \mathsf{dsk_O}, \mathsf{dpk_P})$ and gives $\sigma_\mathcal{T} = (id_\mathcal{T}, \mathcal{C}, n, \tau)$ as well as $\mathsf{sk}_\mathsf{P}^\mathcal{T}$ to P.

$\mathsf{Issue_P}$**:** P runs $\mathsf{Verify}_\mathcal{T}(\mathcal{T}, \sigma_\mathcal{T}, \mathsf{pp}, \mathsf{dpk_O}, \mathsf{sk}_\mathsf{P}^\mathcal{T}, \mathsf{dpk_P})$ to check whether $\sigma_\mathcal{T}$ is a valid signature for $\mathcal{T}$ issued by O. On success, P, on behalf of O, defines a template instantiation $\mathcal{M} \preceq \mathcal{T}$ and runs $\mathsf{Inst}(\mathcal{T}, \mathcal{M}, \sigma_\mathcal{T}, \mathsf{pp}, \mathsf{sk}_\mathsf{P}^\mathcal{T}, \mathsf{dsk_P})$ and publishes $(\mathcal{M}, \sigma_\mathcal{M})$.

Verify: Anybody in possession of the public keys can now take $(\mathcal{M}, \sigma_\mathcal{M})$ and run $\mathsf{Verify}_\mathcal{M}(\mathcal{M}, \sigma_\mathcal{M}, \mathsf{pp}, \mathsf{dpk_P}, \mathsf{dpk_O})$ to check whether $\sigma_\mathcal{M}$ is a valid signature for $\mathcal{M}$ issued by O and P.

Protocol 1: Blank Digital Signature Protocol

We note that Scheme 1 can easily be turned into a scheme using asymmetric pairings giving flexibility in the choice of curves and pairings as well as improved efficiency. In case of Type-2 pairings there are only minor modifications necessary, as there is an efficiently computable isomorphism between G_1 and G_2, whereas in the Type-3 setting this comes at the costs of doubling the size of pp. This is because the values $\alpha P, \ldots, \alpha^t P \in G_1$ also need to be mapped to elements of group G_2, i.e., we need to put the additional points $P', \alpha P', \ldots, \alpha^t P' \in G_2$ into pp, where P' is a generator of G_2.

6.3 Comparison to the Naive Approaches

Recall that the first naive approach given in Section 2.1 would require the originator to produce one signature for every possible template instantiation. Let us look at the above example, where we have 20 fixed elements and 25 exchangeable elements with 5 choices each. Note that this is an absolutely reasonable example, which is far from being overstated. Then, the originator would have to compute $5^{25} \approx 298 \cdot 10^{15}$ signatures, which is obviously impractical.

The second naive approach we have mentioned is the use of Merkle hash trees to reduce the number of signatures that need to be computed by the originator at the expense of higher computational costs and an increased size of the signature. This means that the originator needs to build a complete binary tree, where the number of leaves equals the number of possible template instantiations. Furthermore, each leaf would need to include a random string as additional input to the hash function in order to hide the instantiations from a verifier as it is done in redactable signatures [11]. In our above example, the number of leaves would then be $5^{25} \approx 298 \cdot 10^{15}$. In order to build the hash tree, the originator would need to perform one hash evaluation per node in the tree. Note that for a complete binary tree with n leaves there would be at most $2n - 1$ nodes in the tree. For our above example, this would yield at most $2 \cdot 5^{25} - 1$ hash evaluations and the same number of PRF evaluations to randomize the tree (see [11] for more details on how to compute the random strings using a PRF). Although the verification of a signature for an instantiation in this construction would be quite efficient, as it can be carried in logarithmic time in the number of possible instantiations, the $\mathsf{Sign}, \mathsf{Verify}_\mathcal{T}$ and Inst algorithms all require the computation of the full hash tree rendering this approach impractical.

We emphasize that in our approach the signature size stays constant, regardless of the number of possible template instantiations. This is due to the fact that the template polynomial, whose degree grows only linearly in the template size, is mapped to a point on the curve, which is further mapped to a field element and then hashed. Notice that in our construction, the computational effort is independent of the number of potential template instantiation. Instead, it grows only linearly with the template size, i.e., with the number and the cardinality of exchangeable elements and the number of fixed elements.

6.4 Security

Subsequently, we investigate the security of our construction in the proposed security model by considering all the required security properties.

Theorem 1. *Assuming the existence of secure hash functions and that the t-SDH assumption holds in $E(\mathbb{F}_q)$, Scheme 1 is correct with respect to Definition 10.*

Proof. See Appendix A.1.

Theorem 2. *Assuming the existence of secure hash functions and secure digital signature schemes and that the t-SDH assumption holds in $E(\mathbb{F}_q)$, Scheme 1 is unforgeable with respect to Definition 11.*

Proof. See Appendix A.2.

Theorem 3. *Assuming the existence of secure hash functions and secure digital signature schemes and that the t-SDH assumption holds in $E(\mathbb{F}_q)$, Scheme 1 is immutable with respect to Definition 12.*

Proof. See Appendix A.3.

Theorem 4. *Scheme 1 is unconditionally private with respect to Definition 13.*

Proof. See Appendix A.4.

Taking Theorem 1-Theorem 4 together, we obtain the following corollary.

Corollary 1. *Scheme 1 is a secure* BDSS.

7 Conclusions

In this paper we have introduced a new notion of digital signatures, namely so-called blank digital signatures. We have provided the abstract scheme, a security model and a concrete construction of such a scheme from any secure digital signature scheme, pairing-friendly elliptic curves and polynomial commitments. Moreover, we have proven the security of our construction and have given several use cases, such as delegated contract signing.

7.1 Future Work

Since blank digital signatures are a novel concept, there are several open issues for future work, which we outline subsequently. One issue for future work is to get rid of the trusted third party for key generation. Furthermore, it would be desirable to generalize the blank digital signature scheme and its security model to multiple designated proxies, which seems to be straight-forward by inclusion of multiple proxy signature verification keys into the template signature. However, in this naive construction every proxy and every verifier can determine the set of designated proxies. This may not be desirable in practice, whereas to achieve this goal does not seem to be that straight-forward. Another issue is to find alternative designated constructions for blank signatures potentially without relying on standard digital signature schemes. Additionally, it would be desirable to prove the security of our construction under weaker assumptions and to impose further restrictions on allowed template instantiations, i.e., to further limit the allowed combinations of choices over all exchangeable elements of templates. Also, allowing blank fields, which can be substituted with arbitrary strings, would be desirable. Finally, it may be interesting to investigate concepts applied in the construction of blank digital signatures in the proxy signature setting.

8 Acknowledgements

We would like to thank Jiangtao Li for his many valuable suggestions for improving the presentation of this paper. The work of both authors has been supported by the European Commission through project FP7-FutureID, grant agreement number 318424.

References

1. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable Signatures. In: ESORICS. LNCS, vol. 3679, pp. 159–177. Springer (2005)
2. Benaloh, J.C., de Mare, M.: One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In: Advances in Cryptology - EUROCRYPT '93. Lecture Notes in Computer Science, vol. 765, pp. 274–285 (1993)
3. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM 13(7), 422–426 (1970)
4. Boldyreva, A., Palacio, A., Warinschi, B.: Secure proxy signature schemes for delegation of signing rights. IACR Cryptology ePrint Archive 2003, 96 (2003)
5. Boldyreva, A., Palacio, A., Warinschi, B.: Secure Proxy Signature Schemes for Delegation of Signing Rights. J. Cryptology 25(1), 57–115 (2012)
6. Brzuska, C., Pöhls, H.C., Samelin, K.: Non-interactive public accountability for sanitizable signatures. In: Proc. of the 9th European PKI Workshop: Research and Applications (EuroPKI 2012). LNCS, Springer-Verlag (2012)
7. Canard, S., Jambert, A.: On Extended Sanitizable Signature Schemes. In: Topics in Cryptology - CT-RSA 2010. LNCS, vol. 5985, pp. 179–194 (2010)
8. Chatterjee, S., Menezes, A.: On cryptographic protocols employing asymmetric pairings - the role of ψ revisited. Discrete Applied Mathematics 159(13), 1311–1322 (2011)
9. Gallagher, P., Furlani, C.: FIPS PUB 186-3 federal information processing standards publication digital signature standard (dss) (2009)
10. 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)
11. Johnson, R., Molnar, D., Song, D.X., Wagner, D.: Homomorphic Signature Schemes. In: Topics in Cryptology - CT-RSA 2002. LNCS, vol. 2271, pp. 244–262 (2002)
12. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Advances in Cryptology - ASIACRYPT 2010. pp. 177–194 (2010)
13. Klonowski, M., Lauks, A.: Extended sanitizable signatures. In: ICISC. pp. 343–355 (2006)
14. Mambo, M., Usuda, K., Okamoto, E.: Proxy Signatures for Delegating Signing Operation. In: ACM Conference on Computer and Communications Security (CCS 1996). pp. 48–57 (1996)
15. Merkle, R.C.: A digital signature based on a conventional encryption function. In: CRYPTO. pp. 369–378 (1987)
16. Samelin, K., Poehls, H.C., Posegga, J., de Meer, H.: Redactable vs. Sanitizable Signatures. Tech. Rep. Number MIP-1208, Department of Informatics and Mathematics, University of Passau (2012)
17. Schuldt, J.C.N., Matsuura, K., Paterson, K.G.: Proxy Signatures Secure Against Proxy Key Exposure. In: 11th International Workshop on Practice and Theory in Public-Key (PKC 2008). LNCS, vol. 4939, pp. 141–161 (2008)
18. Silverman, J.: The Arithmetic of Elliptic Curves, Graduate Texts in Mathematics, vol. 106. Springer (1986)
19. Steinfeld, R., Bull, L., Zheng, Y.: Content Extraction Signatures. In: ICISC 2001. LNCS, vol. 2288, pp. 285–304. Springer (2001)

A Proofs

This section contains the proofs of Theorem 1-Theorem 4.

A.1 Proof of Theorem 1

We show the signature correctness and soundness as well as the instantiation correctness, where we omit showing that the DSS verification as well as the signature verification and message signature verification work, since they are clear from the construction.

What remains to show is the signature soundness. Note that for a given template signature $\sigma_{\mathcal{T}}$ for template $\mathcal{T}$ the commitment $\mathcal{C}$ of the template encoding polynomial is fixed. Consequently, breaking the signature soundness requires finding $(\mathsf{sk}_{\mathsf{P}}^{\mathcal{T}}, \mathcal{T}) \neq (\mathsf{sk}_{\mathsf{P}}^{\mathcal{T}^*}, \mathcal{T}^*)$ such that $\mathcal{C}^* = \mathcal{C}$.

This requires either collisions in the hash function H used in the encoding functions, finding second preimages in H or breaking the polynomial binding of the construction. If an adversary A

is able to attack the latter case, then we can construct an efficient adversary B, which uses A to solve the t-SDH problem in $E(\mathbb{F}_q)$. B gets input an instance $(P, \alpha P, \ldots, \alpha^t P)$ of the t-SDH problem. Then B runs $\mathsf{DKeyGen}$ twice to produce DSS key pairs for the originator and the proxy, sets up the public parameters $\mathsf{pp} = (H, E(\mathbb{F}_q), e, p, P, \alpha P, \ldots, \alpha^t P)$ and runs A on pp and the DSS key pairs. If A is able to output a template signature $\sigma_{\mathcal{T}}$ to $\mathcal{T}$ as well as $(\mathsf{sk}_{\mathsf{P}}^{\mathcal{T}} = \rho, \mathcal{T})$ and $(\mathsf{sk}_{\mathsf{P}}^{\mathcal{T}^*} = \rho^*, \mathcal{T}^*)$ with $(\mathsf{sk}_{\mathsf{P}}^{\mathcal{T}}, \mathcal{T}) \neq (\mathsf{sk}_{\mathsf{P}}^{\mathcal{T}^*}, \mathcal{T}^*)$ such that

$$\mathsf{Verify}_{\mathcal{T}}(\mathcal{T}^*, \sigma_{\mathcal{T}}, \mathsf{pp}, \mathsf{dpk}_{\mathsf{O}}, \mathsf{sk}_{\mathsf{P}}^{\mathcal{T}^*}, \mathsf{dpk}_{\mathsf{P}}) = \mathtt{true},$$

then

$$e(\rho \cdot t(\alpha)P, P) = e(\rho^* \cdot t^*(\alpha)P, P)$$

holds. This implies that

$$e(P,P)^{\rho t(\alpha)} = e(P,P)^{\rho^* t^*(\alpha)} = e(P,P)^{\rho t(\alpha) - \rho^* t^*(\alpha)} = 1.$$

Hence, α is a root of the polynomial $t'(X) = \rho t(X) - \rho^* t^*(X)$. As factoring of $t'(X)$ yields α, B can efficiently obtain α and by choosing $c \in \mathbb{Z}_p \setminus \{-\alpha\}$, B can output a solution $(c, \frac{1}{\alpha + c})$ of the t-SDH problem in $E(\mathbb{F}_q)$. □

A.2 Proof of Theorem 2

The proof consists of two parts. The first part covers unforgeability of template signatures, whereas the second part covers the unforgeability of message signatures. Both parts consist of two cases covering the reuse of queried signatures and existential forgeries of signatures, as detailed in Section 5.3.

In the following, let q and q_i be the number of template queries and the number of instantiation queries for template $\mathcal{T}_i$ with $1 \leq i \leq q$ issued by A.

Case T1. This case covers the infeasability of finding some $\mathcal{T}^* \neq \mathcal{T}_i$ for all previously queried $\mathcal{T}_i$ such that $\mathcal{T}^*$ verifies under some queried template signature $\sigma_{\mathcal{T}_i}$. If A was able to find - with non-negligible probability - a $\mathcal{T}^* \neq \mathcal{T}_i$ for all $1 \leq i \leq q$ such that

$$\mathcal{C}_i = e(\rho^* \cdot t_{\mathcal{T}^*}(\alpha)P, P),$$

for some $1 \leq i \leq q$, then A has found a template $\mathcal{T}^* \neq \mathcal{T}_i$ with $\deg(t_{\mathcal{T}^*}) \leq t$ such that either

1. $t_{\mathcal{T}_i}(X) = t_{\mathcal{T}^*}(X)$, or
2. $t_{\mathcal{T}_i}(\alpha) = t_{\mathcal{T}^*}(\alpha)$.

In case one, A has found second preimages in H with non-negligible probability. More precisely, $\mathsf{sk}_{\mathsf{P}}^{\mathcal{T}}$ and $\sigma_{\mathcal{T}_i}$ are fixed. If A is able to generate a $\mathcal{T}^* \neq \mathcal{T}_i$ with $t_{\mathcal{T}_i}(X) = t_{\mathcal{T}^*}(X)$ with non-negligible probability, then it must have found second preimages in H with non-negligible probability for ℓ_i roots of $t(\mathcal{T}_i)$, whereas the suffix of each preimage must be of the form $id_{\mathcal{T}^*} \| j$ and only the value of M is arbitrary.

In the second case, if A is able to do so, then A must have found $\mathcal{T}^* \neq \mathcal{T}_i$ such that $t_{\mathcal{T}_i}(\alpha) = t_{\mathcal{T}^*}(\alpha)$, i.e., the template encoding polynomials need to have the same evaluation at the unknown point α, for some $1 \leq i \leq q$. If A is able to find such a template $\mathcal{T}^*$, then we can construct an adversary B that breaks the t-SDH problem in $E(\mathbb{F}_q)$. The reduction is identical to the reduction in the proof of signature soundness in Appendix A.1 with the only exception that we have $\rho^* = \rho_i$.

Case T2. This case covers the infeasability of computing a valid signature $\sigma_{\mathcal{T}^*}$ for some $\mathcal{T}^*$ giving $\mathcal{C}^*$, which differs from all previously queried template signatures. If A was able to find a pair $(\mathcal{T}^*, \sigma_{\mathcal{T}^*}) \neq (\mathcal{T}_i, \sigma_{\mathcal{T}_i})$ for all $1 \leq i \leq q$ such that

$$\mathsf{DVerify}(\tau, id_{\mathcal{T}^*} \| \mathcal{C}^* \| n^* \| \mathsf{dpk}_{\mathsf{P}}, \mathsf{dpk}_{\mathsf{O}}) = \mathtt{true}$$

then A must be able to forge signatures of the digital signature scheme DSS under $\mathsf{dsk_O}$.

Case M1. This case covers the infeasability of finding some $\mathcal{M}^* \neq \mathcal{M}_{i_j}$ for all previously queried $\mathcal{M}_{i_j}$ such that $\mathcal{M}^*$ verifies under some issued message signature $\sigma_{\mathcal{M}_{i_j}}$, i.e.,

$$\mathsf{Verify}_{\mathcal{M}}(\mathcal{M}^*, \sigma_{\mathcal{M}_{i_j}}, \mathsf{pp}, \mathsf{dpk_P}, \mathsf{dpk_O}) = \mathtt{true}.$$

In particular, an adversary must be able to find $\mathcal{M}^*$ such that either

1. $m_{\mathcal{M}_{i_j}}(X) = m_{\mathcal{M}^*}(X)$, or
2. $m_{\mathcal{M}_{i_j}}(\alpha) = m_{\mathcal{M}^*}(\alpha)$.

In the first case, A needs to find - with non-negligible probability - an $\mathcal{M}^* \neq \mathcal{M}_{i_j}$ for all $1 \leq i \leq q$ and $1 \leq j \leq q_i$ such that

$$|\mathcal{M}^*| = |\mathcal{M}_{i_j}| \quad \wedge \quad \deg(m_{\mathcal{M}^*}) = n \quad \wedge \quad e(\mathcal{C}_{\mathcal{M}^*}, \mathcal{C}_{\overline{\mathcal{M}}_{i_j}}) = \mathcal{C}_i$$

for some $1 \leq i \leq q$ and $1 \leq j \leq q_i$. Since all values in the verification relation are fixed due to μ, the only way for A to output an $\mathcal{M}^*$ that passes the signature verification for an existing signature, is to find an $\mathcal{M}^*$ such that $m_{\mathcal{M}^*}(X) = m_{\mathcal{M}_{i_j}}(X)$ and $|\mathcal{M}^*| = |\mathcal{M}_{i_j}|$. A can do so by computing second preimages in H, i.e.,

$$H(M_l^* \| id_{\mathcal{T}_i} \| l) = H(M_{i_{j_l}} \| id_{\mathcal{T}_i} \| l)$$

for all n roots of the polynomial $m_{\mathcal{M}_{i_j}}$, whereas the suffix of each preimage must be of the form $id_{\mathcal{T}_i} \| l$ and only the value of M_l^* is arbitrary.

In the second case, we show that if A is able to come up with such a forgery, then we can construct an adversary B against the t-SDH assumption in $E(\mathbb{F}_q)$. Adversary B works as follows. B obtains an instance $(P, \alpha P, \ldots, \alpha^t P)$ to the t-SDH problem in $E(\mathbb{F}_q)$, sets $\mathsf{pp} = (H, E(\mathbb{F}_q), e, p, P, \alpha P, \ldots, \alpha^t P)$ and runs A on pp and the public keys of the originator and proxy. When A is able to deliver such a forged $\mathcal{M}^*$, we know that:

$$e(\mathcal{C}_{\mathcal{M}^*}, \mathcal{C}_{\overline{\mathcal{M}}_{i_j}}) = e(\mathcal{C}_{\mathcal{M}_{i_j}}, \mathcal{C}_{\overline{\mathcal{M}}_{i_j}})$$
$$e(\mathcal{C}_{\mathcal{M}^*} - \mathcal{C}_{\mathcal{M}_{i_j}}, \mathcal{C}_{\overline{\mathcal{M}}_{i_j}}) = 1$$

As $\mathcal{C}_{\overline{\mathcal{M}}_{i_j}} \neq \mathcal{O}$ with overwhelming probability, it follows that:

$$m_{\mathcal{M}^*}(\alpha)P = m_{\mathcal{M}_{i_j}}(\alpha)P$$
$$m_{\mathcal{M}^*}(\alpha)P - m_{\mathcal{M}_{i_j}}(\alpha)P = \mathcal{O}$$

Consequently, α is a root of the polynomial $m_{\mathcal{M}^*}(X) - m_{\mathcal{M}_{i_j}}(X) \in \mathbb{Z}_p[X]$. By factoring this polynomial, $\mathcal{B}$ can efficiently obtain α and solve the instance of the the t-SDH problem in $E(\mathbb{F}_q)$ given by pp by choosing $c \in \mathbb{Z}_p \setminus \{-\alpha\}$ and outputting $(c, \frac{1}{\alpha + c}P)$.

Case M2. This case covers the infeasability of computing a valid signature $\sigma_{\mathcal{M}^*}$ for some $\mathcal{M}^*$, which differs from all previously queried signatures. A needs to find a pair $(\mathcal{M}^*, \sigma_{\mathcal{M}^*}) \neq (\mathcal{M}_{i_j}, \sigma_{\mathcal{M}_{i_j}})$ for all $1 \leq i \leq q$ and $1 \leq j \leq q_i$ such that

$$\mathsf{Verify}_{\mathcal{M}}(\mathcal{M}^*, \sigma_{\mathcal{M}^*}, \mathsf{pp}, \mathsf{dpk_P}, \mathsf{dpk_O}) = \mathtt{true}.$$

In both cases **M2a** and **M2b**, this implies that A is able to forge DSS signatures under $\mathsf{dsk_P}$. □

A.3 Proof of Theorem 3

Case T1. The proof for this case is analogous to the proof in Appendix A.2 with the exception that A knows the proxy private keys corresponding to all templates. Therefore, the reduction is identical

to the one in the proof of signatures soundness in Appendix A.1.

Case T2. The proof for this case is analogous to the one in Appendix A.2, since A has exactly the same knowledge as in the previous proof.

Case M1. The proof for this case is analogous to the proof in Appendix A.2.

Case M2b. This case covers the infeasability of existentially forging signatures for messages, which are not instantiations of a template.

Firstly, we show that if A is able to find an $\mathcal{M}^*$ such that $\mathcal{M}^* \not\preceq \mathcal{T}_i$ for all $1 \leq i \leq q$ and $\sigma_{\mathcal{M}^*} \neq \sigma_{\mathcal{M}_{i_j}}$ for all $1 \leq j \leq q_i$ and

$$\mathsf{Verify}_{\mathcal{M}^*}(\mathcal{M}^*, \sigma_{\mathcal{M}^*}, \mathsf{pp}, \mathsf{dpk_P}, \mathsf{dpk_O}) = \mathtt{true},$$

then we can construct an efficient adversary B against the t-SDH assumption in $E(\mathbb{F}_q)$. Finding such an $\mathcal{M}^*$ implies that A has found a polynomial $m_{\mathcal{M}^*}$ such that $m_{\mathcal{M}^*}$ does not perfectly divide $t_{\mathcal{T}_i}$, i.e., $t_{\mathcal{T}_i} = \overline{m}_{\mathcal{M}^*} \cdot m_{\mathcal{M}^*} + \xi$ with $\xi \neq 0$.

Adversary B works as follows. B obtains an instance $(P, \alpha P, \ldots, \alpha^t P)$ to the t-SDH problem in $E(\mathbb{F}_q)$, sets $\mathsf{pp} = (H, E(\mathbb{F}_q), e, p, P, \alpha P)$ and runs A on pp and the respective DSS keys. If A returns $(\mathcal{M}^* = M_0, \sigma_{\mathcal{M}^*} = (\mu^*, \mathcal{C}_{\overline{\mathcal{M}}^*}, \mathcal{I}^* = (|M_i^*|)_{i=1}^n, \sigma_{\mathcal{T}_i}), \mathcal{T}_i)$, then B computes $m_{\mathcal{M}^*} = m_{\mathcal{T}_i}(\mathcal{M}^*)$ and $t_{\mathcal{T}_i} = t(\mathcal{T}_i)$. As we have

$$e(\mathcal{C}_{\mathcal{M}^*}, \mathcal{C}_{\overline{\mathcal{M}}^*}) = \mathcal{C},$$

$\mathcal{C}_{\overline{\mathcal{M}}^*}$ must have the form $\mathcal{C}_{\overline{\mathcal{M}}^*} = \rho_i(\overline{m}_{\mathcal{M}^*}(\alpha) + \frac{\xi(\alpha)}{m_{\mathcal{M}^*}(\alpha)})P$. By dividing $t_{\mathcal{T}_i}(X)$ through $m_{\mathcal{M}^*}(X)$, B obtains $\overline{m}_{\mathcal{M}^*}(X)$ and $\xi(X)$. From this B can compute $\overline{m}_{\mathcal{M}^*}(\alpha)P$. Since B knows ρ_i from the query phase, it can now compute

$$\rho_i^{-1}\mathcal{C}_{\overline{\mathcal{M}}^*} - \overline{m}_{\mathcal{M}^*}(\alpha)P = \frac{\xi(\alpha)}{m_{\mathcal{M}^*}(\alpha)}P.$$

As $\deg(t_{\mathcal{T}_i}) = 1$ and $\deg(m_{\mathcal{M}^*}) = 1$, we have $\deg(\xi) = 0$, i.e., $\xi(X) = \omega \in \mathbb{Z}_p^*$. Therefore, we obtain

$$\frac{\xi(\alpha)}{m_{\mathcal{M}^*}(\alpha)}P = \frac{\omega}{\alpha - H(M_0 \| id_{\mathcal{T}} \| 1)}P$$

and B can now compute

$$\frac{1}{\omega}\frac{\omega}{\alpha - H(M_0 \| id_{\mathcal{T}} \| 1)}P = \frac{1}{\alpha - H(M_0 \| id_{\mathcal{T}} \| 1)}P$$

which gives a solution $(-H(M_0 \| id_{\mathcal{T}} \| 1), \frac{1}{\alpha - H(M_0 \| id_{\mathcal{T}} \| 1)}P)$ to the t-SDH problem in $E(\mathbb{F}_q)$. It is immediate that the success probability of B is only negligibly smaller than that of A, due to the possibility that it could happen that $H(M_0 \| id_{\mathcal{T}} \| 1)$ is α, and the time required is only a small constant larger than that of B.

Secondly, another strategy, A can follow is to use non-intended perfect divisors of $t_{\mathcal{T}_i}$, i.e., constructing an $\mathcal{M}^*$ such that $\deg(m_{\mathcal{M}^*}) \neq n$ and/or the elements of $\mathcal{M}^*$ are not consecutive with respect to the order defined by template $\mathcal{T}_i$. However, the degree n of all valid message polynomials as well as the index values i in the message encoding are fixed by the originator. Thus, the verification can never be satisfied. In the latter case, A would additionally be required to compute second preimages for at least one of the unintended factors. □

A.4 Proof of Theorem 4

The strategy for this proof is as follows. When receiving the template signatures for the challenged templates $\mathcal{T}_0'$ and $\mathcal{T}_1'$, there is no a priori link between the templates and their signatures as they are returned in a randomly permuted order. Since the template ids are randomly chosen, both templates are of the same length n and the proxy public keys are equal, all boils down to deciding correspondence between templates and signatures by inspecting the commitments $\mathcal{C}_0'$ and $\mathcal{C}_1'$ contained in the

template signatures $\sigma_{\mathcal{T}'_0}$ and $\sigma_{\mathcal{T}'_1}$. Analogously, all values in the message signatures of an instantiation $\mathcal{M}'_l$, except for the commitments to the complementary message polynomials and the comprised template signatures, are identical. Latter allows the linking of queried message and template signatures and constructing two lists of commitments $(\mathcal{C}'_0, \mathcal{C}'_{\overline{\mathcal{M}'_{0_1}}}, \ldots, \mathcal{C}'_{\overline{\mathcal{M}'_{0_k}}})$ and $(\mathcal{C}'_1, \mathcal{C}'_{\overline{\mathcal{M}'_{1_1}}}, \ldots, \mathcal{C}'_{\overline{\mathcal{M}'_{1_k}}})$, where we assume w.l.o.g that A has queried all k instantiations. Consequently, the only strategy A can follow to gain an advantage over guessing, is to base its decision upon these two lists. What remains to show is that A has no advantage in winning the game, when using this information, which means that A's success probability for winning the game is exactly $1/2$. We do so by showing that all elements in the above two lists unconditionally hide the respective encoding polynomials.

Let $\mathcal{T}$ be a template allowing s instantiations and $\mathcal{C} = e(\rho \cdot t_{\mathcal{T}}(\alpha)P, P)$ be the commitment to its template encoding polynomial. W.l.o.g. we assume that let $s-1$ message signatures have been issued and the corresponding commitments to the complementary message encoding polynomials $\mathcal{C}_{\overline{\mathcal{M}_1}}, \ldots, \mathcal{C}_{\overline{\mathcal{M}_{s-1}}}$ are known. Nevertheless, this implies that one monomial factor $(X-\gamma)$ in the template encoding polynomial of $\mathcal{T}$ remains unknown. It also implies that $(X-\gamma)$ is an unknown factor of all complementary message encoding polynomials in the commitments $\mathcal{C}_{\overline{\mathcal{M}_i}} = \rho \cdot m_{\overline{\mathcal{M}_i}}(\alpha)P$ for $i = 1, \ldots, s-1$. The only unkown values in $\mathcal{C}$ (and in all other commitments) are ρ and γ. For all these commitments, there are, however, exactly p valid pairs $(\gamma, \rho) \in \mathbb{Z}_p^2$. Consequently, all these commitments unconditionally hide the encoding polynomials. (Note that also in case of a constrained set of $a < p$ possible values for γ - which would for instance be the case if we are dealing with a constrained message space - the hiding is still unconditional, as there are a equally likely pairs of possible values.) $\square$