Blank Digital Signatures*

Christian Hanser
Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology (TUG)
Inffeldgasse 16a, 8010 Graz, Austria
christian.hanser@iaik.tugraz.at

Daniel Slamanig
Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology (TUG)
Inffeldgasse 16a, 8010 Graz, Austria
daniel.slamanig@iaik.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. We also provide a detailed efficiency analysis of our proposed construction supporting its practicality. Finally, we outline several open issues and extensions for future work.

Categories and Subject Descriptors

[**Security and privacy**]: Digital signatures

General Terms

Algorithms, Design, Security, Theory

*This is a revised version of the AsiaCCS13 paper incorporating some corrections.

ASIA CCS'13, May 8–10, 2013, Hangzhou, China.

Keywords

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

1. 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 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 [16] 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 [7]. Surprisingly, only quite recently a suitable security model for proxy signatures has been introduced [6], and been extended to multi-level and identity-based proxy signature schemes later on [21]. 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* [16] 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 [21, 7], 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 [6], 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 [23, 13] as well as sanitizable signature schemes [3] have been introduced. Signature schemes, which allow *removal* of content (replacement by some special symbol $\perp$) by *any* party are called redactable [23, 13], while signature schemes allowing (arbitrary) *replacements* of *admissible* parts by a *designated* party are called sanitizable signature schemes [3], cf. [20] 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 [3] 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 [15], yet, the Bloom filter [5] based construction does not meet cryptographic security requirements and the cryptographic accumulator [4] based approach [15, 9] 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.

1.1 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 [17] 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 5.4 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.

1.2 Outline

In Section 2, we sketch some application scenarios for blank digital signatures. Section 3 discusses the mathematical and cryptographic preliminaries. Then, in Section 4 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, an efficiency analysis and a comparison to the naive approaches are given in Section 5. Finally, Section 6 concludes the paper and lists open issues for future work.

2. 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 [8][1], which supports controlled replacements [9, 15]. 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 [9, 15]. 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.

3. PRELIMINARIES

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

3.1 Mathematical Background

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

$$E : Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6, \quad (1)$$

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 [22].

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 $\Psi : G_2 \to G_1$ [10] 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 $k = \min\{\ell \in \mathbb{N} : p \mid q^\ell - 1\}$.

3.2 Digital Signatures

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

Definition 2 (Digital Signature Scheme). *A digital signature scheme is a tuple* $(\mathsf{KeyGen}^{\mathsf{DSS}}, \mathsf{Sign}^{\mathsf{DSS}}, \mathsf{Verify}^{\mathsf{DSS}})$ *of polynomial-time algorithms:*

$\mathsf{KeyGen}^{\mathsf{DSS}}(\kappa)$ *: Is a key generation algorithm that takes as input a security parameter $\kappa \in \mathbb{N}$ and outputs a private (signing) key $\mathsf{sk}^{\mathsf{DSS}}$ and a public (verification) key $\mathsf{pk}^{\mathsf{DSS}}$.*

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

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

Furthermore, we require the digital signature scheme to be correct, i.e., for all $(\mathsf{sk}^{\mathsf{DSS}}, \mathsf{pk}^{\mathsf{DSS}}) \in \mathsf{KeyGen}(\kappa)$ and all $M \in \{0,1\}^*$, $\mathsf{Verify}^{\mathsf{DSS}}(\mathsf{Sign}^{\mathsf{DSS}}(\mathsf{M}, \mathsf{sk}^{\mathsf{DSS}}), \mathsf{M}, \mathsf{pk}^{\mathsf{DSS}}) = \mathtt{true}$ must hold. A digital signature scheme is secure, if it is existentially unforgeable under adaptively chosen-message attacks ($\mathsf{UF\text{-}CMA}$) [12]. 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.*

3.3 Polynomial Commitments

In [14], Kate et al. introduced the notion of constant-size polynomial commitments. The authors present two distinct commitment schemes, whereas one is unconditionally binding and computationally hiding ($\mathsf{PolyCommit}_{\mathsf{DL}}$) and the other is unconditionally hiding as well as computationally binding ($\mathsf{PolyCommit}_{\mathsf{Ped}}$). For our scheme, we are using $\mathsf{PolyCommit}_{\mathsf{Ped}}$, which is based on Pedersen commitments [19]. The constructions of [14] 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 $\mathsf{PolyCommit}_{\mathsf{Ped}}$ construction of [14].

$\mathsf{Setup}(\kappa, t)$: Pick two groups G, G_T of the same prime order p (with p being a prime of bitlength κ) having a symmetric pairing $e : G \times G \to G_T$. Choose two

[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.

generators $P, Q \in G$ and $\alpha \in_R \mathbb{Z}_p^*$ and output $\mathsf{pk} = (G, G_T, p, e, P, \alpha P, \ldots, \alpha^t P, Q, \alpha Q, \ldots, \alpha^t Q)$ as well as $\mathsf{sk} = \alpha$.

$\mathsf{Commit}(\mathsf{pk}, f(X))$: Given $f(X) \in \mathbb{Z}_p[X]$ with $\deg(f) \leq t$, choose a random polynomial $r(X) \in \mathbb{Z}_p[X]$ with $\deg(f) \leq \deg(r) \leq t$, compute the commitment $\mathcal{C} = f(\alpha)P + r(\alpha)Q \in G$ and output $\mathcal{C}$.

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

$\mathsf{VerifyPoly}(\mathsf{pk}, \mathcal{C}, f(X), r(X))$: Verify whether

$$\mathcal{C} = \sum_{i=0}^{\deg(f)} f^{(i)}(\alpha^i P) + \sum_{i=0}^{\deg(r)} r^{(i)}(\alpha^i Q)$$

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

$\mathsf{CreateWit}(\mathsf{pk}, f(X), r(X), \lambda)$: Compute $\phi(X) = \frac{f(X)-f(\lambda)}{X-\lambda}$, $\hat{\phi}(X) = \frac{r(X)-r(\lambda)}{X-\lambda}$ and $W_\lambda = \phi(\alpha)P + \hat{\phi}(\alpha)Q$ and output $(\lambda, f(\lambda), r(\lambda), W_\lambda)$.

$\mathsf{VerifyEval}(\mathsf{pk}, \mathcal{C}, \lambda, f(\lambda), r(\lambda), W_\lambda)$: Verify that $f(\lambda)$ is the evaluation of f at point λ. This is done by checking whether

$$e(\mathcal{C}, P) = e(W_\lambda, \alpha P - \lambda P) \cdot e(f(\lambda)P + r(\lambda)Q, P)$$

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

This 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 [14]. 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. For the same reason, we simply use a random point R instead of $\sum_{i=0}^{\deg(r)} r^{(i)}(\alpha^i Q)$, which obviously also yields an unconditionally hiding Pedersen commitment.

4. 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.

4.1 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 3 (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 $\mathcal{T}$*. Furthermore, with $|\mathcal{T}|$ we denote the *size of template $\mathcal{T}$*, that is $|\mathcal{T}| = \sum_{i=1}^n |T_i|$. Finally, we call $\max_{1 \leq i \leq n} |T_i|$ the *depth of template $\mathcal{T} = (T_i)_{i=1}^n$*.

Definition 4 (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 $\mathcal{M} =$ ("I, hereby, declare to pay ", "120$", " for this tablet device."). The complement of template instantiation $\mathcal{M}$ is then $\overline{\mathcal{M}} = (\emptyset, \{$"100$", "150$"$\}, \emptyset)$.

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 5 (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 6 (Message Encoding). *Similar to Definition 5, 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.

4.2 Blank Digital Signature Scheme

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

Definition 7 (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 private and public keys* $(\mathsf{pk}_{\mathsf{orig}}, \mathsf{sk}_{\mathsf{orig}})$ *for the originator and returns them.*

$\mathsf{Sign}(\mathcal{T}, \mathsf{sk}_{\mathsf{orig}}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{sk}^{\mathsf{DSS}}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}})$**:** *This probabilistic algorithm takes a message template* $\mathcal{T}$*, the originator's private key* $\mathsf{sk}_{\mathsf{orig}}$*, the originator's public key* $\mathsf{pk}_{\mathsf{orig}}$*, the originator's signing key* $\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{orig}}$*, the proxy's verification key* $\mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}$ *and outputs a template signature* $\sigma_{\mathcal{T}}$ *and a template dependent private key for the proxy* $\mathsf{sk}^{\mathcal{T}}_{\mathsf{proxy}}$*.*

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

$\mathsf{Inst}(\mathcal{T}, \mathcal{M}, \sigma_{\mathcal{T}}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{sk}^{\mathcal{T}}_{\mathsf{proxy}}, \mathsf{sk}^{\mathsf{DSS}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}})$**:** *This probabilistic algorithm takes a template* $\mathcal{T}$*, an instantiation* $\mathcal{M}$*, a template signature* $\sigma_{\mathcal{T}}$*, the public key of the originator* $\mathsf{pk}_{\mathsf{orig}}$*, the private key of the proxy* $\mathsf{sk}^{\mathcal{T}}_{\mathsf{proxy}}$*, the signing key pair* $(\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}})$ *of the proxy and outputs a message signature* $\sigma_{\mathcal{M}}$*.*

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

4.3 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 private key $\mathsf{sk}_{\mathsf{orig}}$ and the signing keys $\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{orig}}$ and $\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{proxy}}$ should be able to forge a valid template or message signature. This is analogous to the security of traditional digital signatures. In particular, this means that it is infeasible to produce forgeries of the following types:

- **T1** finding a template $\mathcal{T}^*$ yielding the same template signature $\sigma_{\mathcal{T}^*} = \sigma_{\mathcal{T}_i}$ for one queried template $\mathcal{T}_i$.
- **T2** forging a valid template signature $\sigma_{\mathcal{T}^*}$ for some unqueried template $\mathcal{T}^*$.
- **M1** finding another message $\mathcal{M}^* \neq \mathcal{M}_{i_j}$ yielding message signature $\sigma_{\mathcal{M}^*}$ such that $\sigma_{\mathcal{M}^*} = \sigma_{\mathcal{M}_{i_j}}$ for some previously queried message $\mathcal{M}_{i_j}$.
- **M2** forging a valid message signature $\sigma_{\mathcal{M}}$ for some non-queried $\mathcal{M}$ from template signature $\sigma_{\mathcal{T}_i}$ for some previously queried template $\mathcal{T}_i$. There are two cases:
  - (a) $\mathcal{M} \preceq \mathcal{T}_i$ for some queried template $\mathcal{T}_i$,
  - (b) $\mathcal{M} \not\preceq \mathcal{T}_i$ for some queried template $\mathcal{T}_i$.

Immutability: The proxy having access to $\mathsf{sk}^{\mathcal{T}}_{\mathsf{proxy}}$ and $\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{proxy}}$, when given a template signature $\sigma_{\mathcal{T}}$ for template $\mathcal{T}$ should not be able to forge signatures. Here, cases **T1**, **T2** and **M1**, **M2**(**b**) from above apply. Note that forgery type **M2**(**a**) is not relevant, since A holds the respective private key anyway.

Privacy: No entity should be able to restore template $\mathcal{T}$ from a message signature $\sigma_{\mathcal{M}}$ representing a signature for an instantiation $\mathcal{M}$ of template $\mathcal{T}$.

4.3.1 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 $(\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}}) \in \mathsf{KeyGen}^{\mathsf{DSS}}(\kappa)$ and $(\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}) \in \mathsf{KeyGen}^{\mathsf{DSS}}(\kappa)$, any BDSS key $(\mathsf{pk}_{\mathsf{orig}}, \mathsf{sk}_{\mathsf{orig}}) \in \mathsf{KeyGen}(\kappa, \mathsf{t})$, any template $\mathcal{T}$ and any honestly computed template signature

$$\sigma_{\mathcal{T}} = \mathsf{Sign}(\mathcal{T}, \mathsf{sk}_{\mathsf{orig}}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{sk}^{\mathsf{DSS}}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}),$$

we require that the verification

$$\mathsf{Verify}_{\mathcal{T}}(\mathcal{T}, \sigma_{\mathcal{T}}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}}, \mathsf{sk}^{\mathcal{T}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}) = \mathtt{true}$$

holds.

Signature soundness: For any key pairs $(\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}}) \in \mathsf{KeyGen}^{\mathsf{DSS}}(\kappa)$ and $(\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}) \in \mathsf{KeyGen}^{\mathsf{DSS}}(\kappa)$, any BDSS key $(\mathsf{pk}_{\mathsf{orig}}, \mathsf{sk}_{\mathsf{orig}}) \in \mathsf{KeyGen}(\kappa, \mathsf{t})$, any template $\mathcal{T}$ and any honestly computed template signature

$$\sigma_{\mathcal{T}} = \mathsf{Sign}(\mathcal{T}, \mathsf{sk}_{\mathsf{orig}}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{sk}^{\mathsf{DSS}}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}),$$

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

$$\mathsf{Verify}_{\mathcal{T}}(\mathcal{T}^*, \sigma_{\mathcal{T}}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}}, \mathsf{sk}^{\mathcal{T}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}) = \mathtt{true}$$

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

Instantiation correctness: For any key pairs $(\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}}) \in \mathsf{KeyGen}^{\mathsf{DSS}}(\kappa)$ and $(\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}) \in \mathsf{KeyGen}^{\mathsf{DSS}}(\kappa)$, any BDSS key $(\mathsf{pk}_{\mathsf{orig}}, \mathsf{sk}_{\mathsf{orig}}) \in \mathsf{KeyGen}(\kappa, t)$, any template $\mathcal{T}$, any honestly computed signature $\sigma_{\mathcal{T}}$ and corresponding $\mathsf{sk}^{\mathcal{T}}_{\mathsf{proxy}}$ such that

$$\mathsf{Verify}_{\mathcal{T}}(\mathcal{T}, \sigma_{\mathcal{T}}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}}, \mathsf{sk}^{\mathcal{T}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}) = \mathtt{true},$$

any honestly computed message signature

$$\sigma_{\mathcal{M}} = \mathsf{Inst}(\mathcal{T}, \mathcal{M}, \sigma_{\mathcal{T}}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{sk}^{\mathcal{T}}_{\mathsf{proxy}}, \mathsf{sk}^{\mathsf{DSS}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}),$$

we require that the verification

$$\mathsf{Verify}_{\mathcal{M}}(\mathcal{M}, \sigma_{\mathcal{M}}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}}) = \mathtt{true}$$

holds.

Definition 8. *A* BDSS *is correct if it satisfies signature correctness, signature soundness and instantiation correctness.*

4.3.2 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.

Definition 9 (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.*

4.3.3 Immutability

Immutability guarantees that no malicious proxy can compute message templates or template instantiations not intended by the signer. In contrast to unforgeability, immutability deals with malicious insiders.

The immutability game differs only slightly from the unforgeability game. Therefore, we only briefly discuss the differences. Here, the adversary additionally obtains $\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{proxy}}$ from the challenger in the setup phase. In the query phase, A has access to a template signing oracle, which additionally returns the corresponding private key $\mathsf{sk}^{\mathcal{T}_i}_{\mathsf{proxy}}$ and can query signatures for arbitrary templates. A can compute valid template instantiations locally. In this game, A wins if he outputs valid forgeries of type **T1**, **T2** or **M1**, **M2**(**b**).

Here, forgery type **M1** covers already issued message signatures and prevents a proxy from presenting another, not previously signed, template instantiation, which validates under an already issued signature.

Setup: The challenger B runs $\mathsf{KeyGen}(\kappa, t)$ to obtain $(\mathsf{pk}_{\mathsf{orig}}, \mathsf{sk}_{\mathsf{orig}})$. Furthermore, B runs $\mathsf{KeyGen}^{\mathsf{DSS}}(\kappa)$ of a secure digital signature scheme twice to generate $(\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}})$ and $(\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}})$. It gives the adversary A the resulting public keys $\mathsf{pk}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}}$ and $\mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}$ and keeps the private keys $\mathsf{sk}_{\mathsf{orig}}, \mathsf{sk}^{\mathsf{DSS}}_{\mathsf{orig}}$ and $\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{proxy}}$ to itself.

Query: The adversary A has access to an oracle $\mathcal{O}$, which is simulated by the challenger and answers to queries of the form $(\mathcal{T}_i, \mathcal{M}_{\mathcal{T}_i})$ as follows. For each template $\mathcal{T}_i$ the challenger chooses $id_{\mathcal{T}_i}$, computes and stores $(\sigma_{\mathcal{T}_i}, \mathsf{sk}^{\mathcal{T}_i}_{\mathsf{proxy}}) = \mathsf{Sign}(\mathcal{T}_i, \mathsf{sk}_{\mathsf{orig}}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{sk}^{\mathsf{DSS}}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}})$. Next, B computes $\sigma_{\mathcal{M}_{i_j}} = \mathsf{Inst}(\mathcal{T}_i, \mathcal{M}_{i_j}, \sigma_{\mathcal{T}_i}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{sk}^{\mathcal{T}_i}_{\mathsf{proxy}}, \mathsf{sk}^{\mathsf{DSS}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}})$ and returns $\sigma_{\mathcal{M}_{i_j}}$. A can issue these queries for different templates as well as template instantiations in an adaptively interleaved manner.

Output: The adversary A outputs either a triple $(\mathcal{T}^*, \sigma_{\mathcal{T}^*}, \mathsf{sk}^{\mathcal{T}^*}_{\mathsf{proxy}})$ or a pair $(\mathcal{M}^*, \sigma_{\mathcal{M}^*})$. A wins if either $\mathsf{Verify}_{\mathcal{T}}(\mathcal{T}^*, \sigma_{\mathcal{T}^*}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}}, \mathsf{sk}^{\mathcal{T}^*}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}})$ or $\mathsf{Verify}_{\mathcal{M}}(\mathcal{M}^*, \sigma_{\mathcal{M}^*}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}})$ accepts, whereas the forgeries must be of type **T1**, **T2** or **M1**, **M2**.

Game 1: Unforgeability Game

Definition 10 (Immutability). *A* BDSS *is called* immutable*, if for any polynomial-time algorithm A the probability of winning the immutability game is negligible as a function of security parameter κ.*

4.3.4 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 $\mathcal{T}$. More precisely, let k be the depth of template $\mathcal{T}$, which is the smallest number of instantiations necessary to reveal all choices of $\mathcal{T}$, then no verifier should be able to derive information about the remaining unseen choices inside $\mathcal{T}$ by knowing $k-1$ instantiations.

Definition 11 (Privacy). *A* BDSS *is called* private*, if for any polynomial-time algorithm A the probability of winning Game 2 is negligible as a function of security parameter κ.*

4.3.5 Security

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 $e : E(\mathbb{F}_q)[p] \times E(\mathbb{F}_q)[p] \to \mathbb{F}^*_{q^k}[p]$, a full-domain cryptographic hash function $H : \{0,1\}^* \to \mathbb{Z}_p$ for use with the encoding functions and a full-domain cryptographic hash function $H' : \mathbb{F}^*_{q^k}[p] \to \{0,1\}^\kappa$. Let $P \in E(\mathbb{F}_q)[p]$ be a generator and let $g = e(P,P)$ be a generator of $\mathbb{F}^*_{q^k}[p]$. Pick two elements $\alpha, \delta \in_R \mathbb{Z}^*_p$, compute $(\alpha P, \ldots, \alpha^t P)$ as well as $(\delta P, \alpha\delta P, \ldots, \alpha^t \delta P)$ and output $\mathsf{sk}_{\mathsf{orig}} = \delta$, $\mathsf{pk}_{\mathsf{orig}} = (H, H', E(\mathbb{F}_q), e, p, P, g, \alpha P, \ldots, \alpha^t P, \delta P, \alpha\delta P, \ldots, \alpha^t \delta P)$.

Sign: Given $\mathcal{T}, \mathsf{sk}_{\mathsf{orig}}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{sk}^{\mathsf{DSS}}_{\mathsf{orig}}$ and $\mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}$, 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 random point $R \in E(\mathbb{F}_q)[p]$, computes $R_0 = \delta R$,

$$\mathcal{C} = H'(e(\sum_{i=0}^{\ell} t^{(i)}_{\mathcal{T}}(\alpha^i P) + R, \delta P)) \quad \text{and} \quad \tau = \mathsf{Sign}^{\mathsf{DSS}}(id_{\mathcal{T}} \| \mathcal{C} \| R_0 \| R_2 \| n \| \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}, \mathsf{sk}^{\mathsf{DSS}}_{\mathsf{orig}}).$$

It picks a point $R_1 \in_R E(\mathbb{F}_q)[p]$, computes $R_2 = R - R_1$, and returns the template signature $\sigma_{\mathcal{T}} = (id_{\mathcal{T}}, \mathcal{C}, R_0, R_2, n, \tau)$ as well as $\mathsf{sk}^{\mathcal{T}}_{\mathsf{proxy}} = R_1$.

Verify$_{\mathcal{T}}$: Given $\mathcal{T}, \sigma_{\mathcal{T}}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}}, \mathsf{sk}^{\mathcal{T}}_{\mathsf{proxy}}$ and $\mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}$, 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

$$\mathsf{Verify}^{\mathsf{DSS}}(\tau, id_{\mathcal{T}} \| \mathcal{C} \| R_0 \| R_2 \| n \| \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}}) = \texttt{true} \ \wedge \ e(R_0, P) = e(R_1 + R_2, \delta P) \ \wedge \ H'(e(\sum_{i=0}^{\ell} t^{(i)}_{\mathcal{T}}(\alpha^i \delta P) + R_0, P)) = \mathcal{C}$$

where $t^{(i)}_{\mathcal{T}}$ is the i'th coefficient of $t_{\mathcal{T}}$. If so, return `true` and `false` otherwise.

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

$$\mathcal{C}_{\overline{\mathcal{M}}} = \sum_{i=0}^{\ell-n} m^{(i)}_{\overline{\mathcal{M}}}(\alpha^i P) + \gamma R_1 \quad \text{and} \quad \mu = \mathsf{Sign}^{\mathsf{DSS}}(\tau \| \mathcal{C}_{\overline{\mathcal{M}}} \| \gamma \| \mathcal{I}, \mathsf{sk}^{\mathsf{DSS}}_{\mathsf{proxy}}),$$

where $m^{(i)}_{\overline{\mathcal{M}}}$ is the i'th coefficient of $m_{\overline{\mathcal{M}}}$. It returns $\sigma_{\mathcal{M}} = (\mu, \mathcal{C}_{\overline{\mathcal{M}}}, \gamma, \mathcal{I}, \sigma_{\mathcal{T}})$.

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

$$\mathsf{Verify}^{\mathsf{DSS}}(\tau, id_{\mathcal{T}} \| \mathcal{C} \| R_0 \| R_2 \| n \| \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}}) = \texttt{true} \wedge \mathsf{Verify}^{\mathsf{DSS}}(\mu, \tau \| \mathcal{C}_{\overline{\mathcal{M}}} \| \gamma \| \mathcal{I}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}) = \texttt{true} \wedge |\mathcal{I}| = n \wedge \sum_{i=1}^{n} |M_i| = |\mathcal{M}|$$

On failure return `false`, otherwise evaluate $m_{\mathcal{M}} = m_{\mathcal{T}}(\mathcal{M})$, let $m^{(i)}_{\mathcal{M}}$ be the i'th coefficient of $m_{\mathcal{M}}$, and compute

$$\mathcal{C}_{\mathcal{M}} = \sum_{i=0}^{n} m^{(i)}_{\mathcal{M}}(\alpha^i P) \text{ and } \mathcal{C}'_{\mathcal{M}} = \sum_{i=0}^{n} m^{(i)}_{\mathcal{M}}(\alpha^i \delta P)$$

and check whether

$$H'(e(\mathcal{C}_{\overline{\mathcal{M}}} + \gamma R_2, \mathcal{C}'_{\mathcal{M}}) \cdot e((1-\gamma)R_0, \mathcal{C}_{\mathcal{M}}) \cdot e(P - \mathcal{C}_{\mathcal{M}}, R_0)) = \mathcal{C}$$

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

Scheme 1: Blank Digital Signature Scheme

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

DEFINITION 12. *We call a* BDSS secure*, if it is correct, unforgeable, immutable and private.*

5. CONSTRUCTION

In this section we detail our construction and present an analysis of its efficiency in terms of computational effort as well as key and signature sizes. Moreover, we prove its security, i.e., we show that our presented construction is correct, unforgeable, immutable and private.

5.1 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 [11], 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 4, 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

We assume that the originator as well as the proxy both own an authentic key pair for a secure digital signature scheme $(\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}})$ and $(\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}})$, respectively.

$\mathsf{Setup}_{\mathsf{T}}$**:** The trusted third party T chooses a suitable security parameter κ and a value $t \in \mathbb{N}$ representing the maximum template length, runs $\mathsf{KeyGen}(\kappa, t)$, sends $\mathsf{sk}_{\mathsf{orig}}$ to the originator O and publishes $\mathsf{pk}_{\mathsf{orig}}$ in an authentic fashion.

$\mathsf{Issue}_{\mathsf{O}}$**:** O defines a message template $\mathcal{T}$, runs $\mathsf{Sign}(\mathcal{T}, \mathsf{sk}_{\mathsf{orig}}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{sk}^{\mathsf{DSS}}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}})$ and gives $\sigma_{\mathcal{T}} = (id_{\mathcal{T}}, \mathcal{C}, n, \tau)$ as well as $\mathsf{sk}^{\mathcal{T}}_{\mathsf{proxy}}$ to P.

$\mathsf{Issue}_{\mathsf{P}}$**:** P runs $\mathsf{Verify}_{\mathcal{T}}(\mathcal{T}, \sigma_{\mathcal{T}}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}}, \mathsf{sk}^{\mathcal{T}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}})$ 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 $\mathsf{Inst}(\mathcal{T}, \mathcal{M}, \sigma_{\mathcal{T}}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{sk}^{\mathcal{T}}_{\mathsf{proxy}}, \mathsf{sk}^{\mathsf{DSS}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}})$ 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 $\mathsf{Verify}_{\mathcal{M}}(\mathcal{M}, \sigma_{\mathcal{M}}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}})$ to check whether $\sigma_{\mathcal{M}}$ is a valid signature for $\mathcal{M}$ issued by O and P.

Protocol 1: Blank Digital Signature Protocol

Setup: The challenger B runs $\mathsf{KeyGen}(\kappa, t)$ to obtain $(\mathsf{pk}_{\mathsf{orig}}, \mathsf{sk}_{\mathsf{orig}})$. Furthermore, B runs $\mathsf{KeyGen}^{\mathsf{DSS}}(\kappa)$ of a secure digital signature scheme twice to generate $(\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}})$ and $(\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}})$. It gives the adversary A the resulting public keys $\mathsf{pk}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}}$ and $\mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}$ and keeps the private keys $\mathsf{sk}_{\mathsf{orig}}$, $\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{orig}}$ and $\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{proxy}}$ to itself.

Query 1: The adversary A issues template signature queries $\mathcal{T}_1, \ldots, \mathcal{T}_q$. To each query $\mathcal{T}_i$ the challenger responds by running $\mathsf{Sign}(\mathcal{T}_i, \mathsf{sk}_{\mathsf{orig}}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{sk}^{\mathsf{DSS}}_{\mathsf{orig}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}})$ to generate a signature $\sigma_{\mathcal{T}_i}$ for $\mathcal{T}_i$ and sending it together with $\mathsf{sk}^{\mathcal{T}_i}_{\mathsf{proxy}}$ to the adversary. For each template signature query $\mathcal{T}_i$, A can issue an arbitrary number of signature instantiation queries of the form $(\mathcal{T}_i, \mathcal{M}_{i_j})$. To every such query B responds by running $\mathsf{Inst}(\mathcal{T}_i, \mathcal{M}_{i_j}, \sigma_{\mathcal{T}}, \mathsf{pk}_{\mathsf{orig}}, \mathsf{sk}^{\mathcal{T}_i}_{\mathsf{proxy}}, \mathsf{sk}^{\mathsf{DSS}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}})$ and returning $\sigma_{\mathcal{M}_{i_j}}$. All of these queries can be made adaptively interleaved.

Challenge: At some point A signals B that he is ready to be challenged. B constructs a non-queried random template $\mathcal{T}$ of depth k and produces a template signature for $\mathcal{T}$, but keeps $\mathcal{T}$ and $\mathsf{sk}^{\mathcal{T}}_{\mathsf{proxy}}$ to itself.

Query 2: The adversary A is allowed to issue queries as in query phase 1 and at most $k-1$ instantiation queries to the unknown template $\mathcal{T}$. All of these queries can be made adaptively interleaved.

Output: The adversary A outputs $\mathcal{T}^*$ and wins if $\mathcal{T}^* = \mathcal{T}$.

Game 2: Privacy Game

is represented by a message encoding polynomial that perfectly 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, whereas the originator's commitment is given as hash value in order to prevent arithmetics. 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 Pedersen commitments of the form $\mathcal{C} = f(\alpha)P + R$ to hide the committed polynomials. Pedersen commitments provide computational binding as long as the discrete logarithm between the values R and P is unknown. Typically, this is achieved by letting a third party compute these parameters. In our scenario, R is chosen by the originator, which requires us to additionally commit to a multiple of R and to use an additive decomposition of R in order to fix it.

5.2 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 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 $\mathsf{pk}_{\mathsf{orig}}$. This is because the values $\alpha P, \ldots, \alpha^t P, \delta P, \alpha\delta P, \ldots, \alpha^t \delta P \in G_1$ also need to be mapped to elements of group G_2, i.e., we need to put the additional points $\alpha P', \ldots, \alpha^t P', \delta P', \alpha\delta P', \ldots, \alpha^t \delta P' \in G_2$ into $\mathsf{pk}_{\mathsf{orig}}$, where P' is a generator of G_2.

5.3 Efficiency

In the following, we give a detailed analysis of the efficiency of our scheme in terms of computational effort as well as key and signature sizes. We use $\mathtt{P}, \mathtt{S}, \mathtt{A}$ and $\mathtt{m}$ to denote the costs of a pairing evaluation, the costs of an elliptic curve scalar multiplication, the costs of a point addition and the costs of one multiplication in $\mathbb{Z}_p$, respectively. Moreover, we use $\mathtt{Exp}(\mathtt{d}), \mathtt{s}$ and $\mathtt{v}$ for the costs of the expansion of a polynomial of degree d, the costs of a signing operation and the costs of a signature verification, respectively.

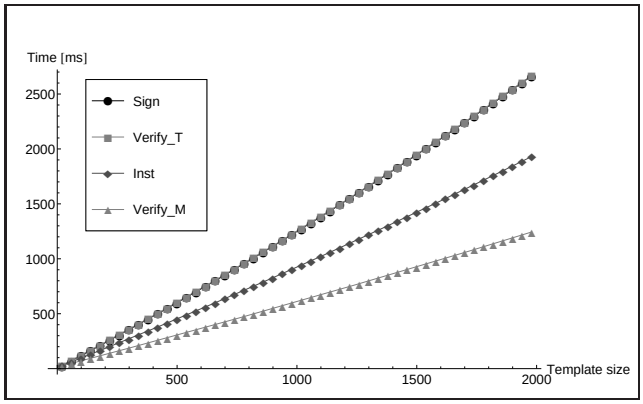

Figure 1: Performance Evaluation of the Algorithms in Scheme 1.

Note that $\texttt{Exp(d)}$ is $\texttt{d(d+1)/2m}$ when ignoring shifts and field additions, which are inexpensive. The costs of the algorithms of Scheme 1 are summarized in Table 1. Figure 1 illustrates the performance of the algorithms in Scheme 1 for varying template size $|\mathcal{T}| = \ell$ and template length n set to $\ell/4$, meaning that on average there are three choices per exchangeable element. We conducted the experiments on an Intel Core i5-2540M equipped with 8GB RAM running Ubuntu 12.10/amd64 and OpenJDK 6/amd64. For ECDSA and basic elliptic curve arithmetics we were using the Java™ library ECCelerate™ 2.0 [1], which is freely available for purposes of research, as well as the jPBC library [2] version 1.2.1 for the pairing evaluations. Thereby, we are using the NIST P-224 curve [11] for ECDSA and the Tate pairing on an MNT curve [18] with a group size of 224 bits and embedding degree $k = 6$ in order to perform our benchmarks, respectively. Note that we have chosen the asymmetric version of Scheme 1 to gain improved efficiency.

Algorithm	Costs
KeyGen	$\texttt{(2t+1)S}$
Sign	$\texttt{P+}(\ell+1)\texttt{S+}(\ell+2)\texttt{A+Exp}(\ell)\texttt{+s}$
$\mathsf{Verify}_{\mathcal{T}}$	$\texttt{3P+}(\ell+1)\texttt{S+}(\ell+2)\texttt{A+Exp}(\ell)\texttt{+v}$
Inst	$\texttt{3P+}(\ell-\texttt{n+2})\texttt{S+}(\ell-\texttt{n+1})\texttt{A+Exp}(\ell-\texttt{n})\texttt{+s+v}$
$\mathsf{Verify}_{\mathcal{M}}$	$\texttt{3P+(2n+4)S+(2n+2)A+Exp(n)+2v}$

Table 1: Costs of the Algorithms of Scheme 1 for templates of size ℓ and length n.

In Table 2, we analyze the sizes of keys and signatures of Scheme 1. We use a pairing-friendly curve of bitsize κ, assume all points to be compressed and for the DSS we use ECDSA with key sizes of κ bits. Consequently, the public and private keys of the DSS are of size κ bits and the signature is of size 2κ. Note that we only count values that are related to the security parameter κ and, consequently, no small values, such as integers and the like.

Component	Size
$\mathsf{sk}_{\mathsf{orig}}$	κ
$\mathsf{pk}_{\mathsf{orig}}$	$2(t+1)\kappa$
$\mathsf{sk}_{\mathsf{proxy}}$	κ
$\sigma_{\mathcal{T}}$	6κ
$\sigma_{\mathcal{M}}$	10κ

Table 2: Sizes of Keys and Signatures in Scheme 1.

Observe that only $\mathsf{pk}_{\mathsf{orig}}$ depends on the maximum degree ℓ of the template polynomial $t_{\mathcal{T}}$. For instance, given a template $\mathcal{T}$ with 20 fixed elements and 25 exchangeable elements with 5 choices each, we obtain a template of size $\ell = 145$ yielding a template polynomial $t_{\mathcal{T}}$ with $\deg(t_{\mathcal{T}}) = 145$. Consequently, we have $t \geq 145$ and the size of $\mathsf{pk}_{\mathsf{orig}}$ would be approximately 8.1kB for $\kappa = 224$, which is absolutely reasonable.

5.4 Comparison to the Naive Approaches

Recall that the first naive approach given in Section 1.1 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 [13]. 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 [13] 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.

5.5 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 hardness of the* ECDLP *as well as the existence of second preimage resistant hash functions, Scheme 1 is correct with respect to Definition 8.*

PROOF. See Appendix A.2. □

THEOREM 2. *Assuming the hardness of the* ECDLP *as well as the existence of second preimage resistant hash functions and secure digital signature schemes, Scheme 1 is unforgeable with respect to Definition 9.*

PROOF. See Appendix A.1. □

THEOREM 3. *Assuming the hardness of the* ECDLP *as well as the existence of second preimage resistant hash functions and secure digital signature schemes, Scheme 1 is immutable with respect to Definition 10.*

PROOF. See Appendix A.3. □

THEOREM 4. *Scheme 1 is private with respect to Definition 11.*

PROOF. See Appendix A.4. □

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

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

6. 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, have analyzed its efficiency supporting its practicality and have given several use cases, such as delegated contract signing.

6.1 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.

7. 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.

8. REFERENCES

[1] ECCelerate 2.0. `http://jce.iaik.tugraz.at/sic/Products/Core-Crypto-Toolkits/ECCelerate`, 2012.

[2] jPBC 1.2.1. `http://gas.dia.unisa.it/projects/jpbc`, 2012.

[3] G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik. Sanitizable Signatures. In *ESORICS*, volume 3679 of *LNCS*, pages 159–177. Springer, 2005.

[4] J. C. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In *Advances in Cryptology - EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 274–285, 1993.

[5] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[6] A. Boldyreva, A. Palacio, and B. Warinschi. Secure proxy signature schemes for delegation of signing rights. *IACR Cryptology ePrint Archive*, 2003:96, 2003.

[7] A. Boldyreva, A. Palacio, and B. Warinschi. Secure Proxy Signature Schemes for Delegation of Signing Rights. *J. Cryptology*, 25(1):57–115, 2012.

[8] C. Brzuska, H. C. Pöhls, and K. Samelin. Non-interactive public accountability for sanitizable signatures. In *Proc. of the 9th European PKI Workshop: Research and Applications (EuroPKI 2012)*, LNCS. Springer-Verlag, 2012.

[9] S. Canard and A. Jambert. On Extended Sanitizable Signature Schemes. In *Topics in Cryptology - CT-RSA 2010*, volume 5985 of *LNCS*, pages 179–194, 2010.

[10] S. Chatterjee and A. Menezes. On cryptographic protocols employing asymmetric pairings - the role of ψ revisited. *Discrete Applied Mathematics*, 159(13):1311–1322, 2011.

[11] P. Gallagher and C. Furlani. FIPS PUB 186-3 federal information processing standards publication digital signature standard (dss), 2009.

[12] S. Goldwasser, S. Micali, and R. L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[13] R. Johnson, D. Molnar, D. X. Song, and D. Wagner. Homomorphic Signature Schemes. In *Topics in Cryptology - CT-RSA 2002*, volume 2271 of *LNCS*, pages 244–262, 2002.

[14] A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology - ASIACRYPT 2010*, pages 177–194, 2010.

[15] M. Klonowski and A. Lauks. Extended sanitizable signatures. In *ICISC*, pages 343–355, 2006.

[16] M. Mambo, K. Usuda, and E. Okamoto. Proxy

Signatures for Delegating Signing Operation. In *ACM Conference on Computer and Communications Security (CCS 1996)*, pages 48–57, 1996.

[17] R. C. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO*, pages 369–378, 1987.

[18] Miyaji, Nakabayashi, and Takano. New Explicit Conditions of Elliptic Curve Traces for FR-Reduction. *TIEICE: IEICE Transactions on Communications/Electronics/Information and Systems*, 2001.

[19] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, pages 129–140, 1991.

[20] K. Samelin, H. C. Poehls, J. Posegga, and H. de Meer. Redactable vs. Sanitizable Signatures. Technical Report Number MIP-1208, Department of Informatics and Mathematics, University of Passau, 2012.

[21] J. C. N. Schuldt, K. Matsuura, and K. G. Paterson. Proxy Signatures Secure Against Proxy Key Exposure. In *11th International Workshop on Practice and Theory in Public-Key (PKC 2008)*, volume 4939 of *LNCS*, pages 141–161, 2008.

[22] J. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *Graduate Texts in Mathematics*. Springer, 1986.

[23] R. Steinfeld, L. Bull, and Y. Zheng. Content Extraction Signatures. In *ICISC 2001*, volume 2288 of *LNCS*, pages 285–304. Springer, 2001.

APPENDIX

A. PROOFS

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

A.1 Proof of Theorem 2 (Sketch)

The proof consists of two parts. The first part covers unforgeability of template signatures, whereas the second part covers unforgeability of message signatures. Both parts consist of two cases covering the reuse of queried signatures and existential forging of signatures as detailed in Section 4.3.

During the query phase, A is allowed to issue an arbitrary number of template instantiation queries for every previously queried template $\mathcal{T}_i$.

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

$$\mathcal{C}_i = H'(e(\sum_{i=0}^{\ell} t^{(i)}_{\mathcal{T}^*}(\alpha^i P) + R_i), \delta P)),$$

for some $1 \leq i \leq q$, then A has either

1. found a template $\mathcal{T}^* \neq \mathcal{T}_i$ such that $t_{\mathcal{T}_i}(X) = t_{\mathcal{T}^*}(X)$, or
2. found a second polynomial $t_{\mathcal{T}^*}$ with $\deg(t_{\mathcal{T}^*}) \leq t$ with $t_{\mathcal{T}_i}(\alpha) = t_{\mathcal{T}^*}(\alpha)$ and the corresponding template $\mathcal{T}^*$.

In case one, A has found second preimages in H with non-negligible probability. More precisely, $\mathsf{sk}^{\mathcal{T}}_{\mathsf{proxy}}$ 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 he 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 a polynomial $t_{\mathcal{T}^*}$ with $t_{\mathcal{T}_i}(\alpha) = t_{\mathcal{T}^*}(\alpha)$, i.e., having the same evaluation at the unknown point α as $t_{\mathcal{T}_i}$ for some $1 \leq i \leq q$. If A is able to extract the value α from αP by breaking the ECDLP, for which the probability is negligibly small, then he could construct a polynomial $t_{\mathcal{T}^*}$ with $t_{\mathcal{T}_i}(\alpha) = t_{\mathcal{T}^*}(\alpha)$. However, then A would still need to find second preimages in H as in case 1 to construct a corresponding template $\mathcal{T}^*$.

Taking all together, gives us the desired contradiction.

Case T2. This case covers the infeasability of computing a valid signature $\sigma_{\mathcal{T}^*}$ for some $\mathcal{T}^*$ giving $\mathcal{C}^*$ (which may be computed by A), which differs from all previously queried 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

$$\mathsf{Verify}^{\mathsf{DSS}}(\tau, id_{\mathcal{T}^*} \| \mathcal{C}^* \| R_0^* \| R_2^* \| n^* \| \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{proxy}}, \mathsf{pk}^{\mathsf{DSS}}_{\mathsf{orig}}) = \mathtt{true}$$

then A must be able to forge signatures of the digital signature scheme DSS under $\mathsf{sk}^{\mathsf{DSS}}_{\mathsf{orig}}$. This gives us the desired contradiction.

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 $\sigma_{\mathcal{M}_{i_j}}$. If A was able 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

$$|\mathcal{M}^*| = |\mathcal{M}_{i_j}| \quad \wedge \quad \deg(m_{\mathcal{M}^*}) = n \quad \wedge$$
$$H'(e(\mathcal{C}_{\overline{\mathcal{M}}_{i_j}} + \gamma_{i_j} R_{2,i}, \mathcal{C}'_{\mathcal{M}^*}) \cdot e((1-\gamma_{i_j}) R_{0,i}, \mathcal{C}_{\mathcal{M}^*}) \cdot$$
$$\cdot e(P - \mathcal{C}_{\mathcal{M}^*}, R_{0,i})) = \mathcal{C}_i$$

for some $1 \leq i \leq q$ and $1 \leq j \leq q_i$ then A has found second preimages in H with non-negligible probability. More precisely, since due to μ all values in the verification relation are fixed, 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 $m(\mathcal{M}^*) = m(\mathcal{M}_{i_j})$ and $|\mathcal{M}^*| = |\mathcal{M}_{i_j}|$, 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.

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 $1 \leq j \leq q_i$ such that $\deg(m_{\mathcal{M}^*}) = n$ and

$$H'(e(\mathcal{C}_{\overline{\mathcal{M}}^*} + \gamma^* R_2^*, \mathcal{C}'_{\mathcal{M}^*}) \cdot e((1-\gamma^*) R_0^*, \mathcal{C}_{\mathcal{M}^*}) \cdot$$
$$\cdot e(P - \mathcal{C}_{\mathcal{M}^*}, R_0^*)) = \mathcal{C}_i$$

First we consider finding $\mathcal{M}^* \preceq \mathcal{T}_i$ for some i, for which $\sigma_{\mathcal{M}^*} \neq \sigma_{\mathcal{M}_{i_j}}$ for all $1 \leq j \leq q_i$. In order for this to hold for a given $\mathcal{C}_i$, A must construct $\mathcal{C}_{\overline{\mathcal{M}}^*}$. Since A knows all templates, A can choose $\mathcal{M}^* \preceq \mathcal{T}_i$ and compute $\mathcal{C}_{\overline{\mathcal{M}}^*} = \sum_{l=0}^{\ell_i - n_i} m^{(l)}_{\overline{\mathcal{M}}^*}(\alpha^l P) + \gamma^* R_1$, whereas R_1 can be extracted from previous queries as γ_{i_j} and $m_{\overline{\mathcal{M}}_{i_j}}$ are known. Still,

A must be able to forge signatures of the digital signature scheme DSS under $\mathsf{sk}_{\mathsf{proxy}}^{\mathsf{DSS}}$.

Secondly, we consider finding $\mathcal{M}^* \not\subseteq \mathcal{T}_i$, for which $\sigma_{\mathcal{M}^*} \neq \sigma_{\mathcal{M}_{i_j}}$ for all $1 \leq j \leq q_i$. Finding such an $\mathcal{M}^*$ implies that A has found a polynomial $m_{\mathcal{M}^*}$ such that $m_{\mathcal{M}^*}$ does not perfectly divide $t_{\mathcal{T}_i}$. This means that $t_{\mathcal{T}_i} = m_{\overline{\mathcal{M}}^*} \cdot m_{\mathcal{M}^*} + \xi$ with $\xi \neq 0$. Note that the only way for A to include the remainder ξ for the verification relation is to put $\xi(\alpha)$ into $\mathcal{C}_{\overline{\mathcal{M}}^*}$. However, since the only place where $\mathcal{C}_{\overline{\mathcal{M}}^*}$ is input to the verification is the first pairing $e(\mathcal{C}_{\overline{\mathcal{M}}^*} + \gamma R_2, \mathcal{C}'_{\mathcal{M}^*})$, this will result in $\mathcal{C}_{\overline{\mathcal{M}}^*} = \sum_{l=0}^{\ell_i - n_i} m_{\overline{\mathcal{M}}^*}^{(l)}(\alpha^l P) + \frac{\xi(\alpha)}{m_{\mathcal{M}^*}(\alpha)} R_1$. Furthermore, A then needs to compute γ as $\frac{\xi(\alpha)}{m_{\mathcal{M}^*}(\alpha)}$, as these are the only values A can influence. However, this requires A to have knowledge of α, which requires A to break the ECDLP. 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. 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, unless A has the power to compute second preimages.

In all cases, A must be able to forge signatures of the digital signature scheme DSS under $\mathsf{sk}_{\mathsf{proxy}}^{\mathsf{DSS}}$. This gives us the desired contradictions.

Since all cases gave us the desired contradictions, the proof is complete. □

A.2 Proof of Theorem 1 (Sketch)

We show the signature and instantiation correctness, where we omit showing that the DSS verification works, since it is clear from the construction. The signature correctness is obvious, and we now show the signature soundness. This means that we have to show that the originator commits to exactly one template polynomial by issuing a template signature. Similarly to case **T1** in the proof of unforgeability, the originator would need to find second preimages or another polynomial having the same evaluation at the unknown point α. Moreover, since he commits to R by signing R_0 along with R_2 and gives R_1 to the proxy, the proxy can check this commitment by verifying whether $e(R_0, P) = e(R_1 + R_2, \delta P)$ holds, which prevents the originator from changing R after signature generation.

It remains to show that the message signature verification works, i.e., we have to prove the correctness of

$$H'(e(\mathcal{C}_{\overline{\mathcal{M}}} + \gamma R_2, \mathcal{C}'_{\mathcal{M}}) \cdot e((1-\gamma)R_0, \mathcal{C}_{\mathcal{M}}) \cdot e(P - \mathcal{C}_{\mathcal{M}}, R_0)) = \mathcal{C},$$

whereas $\mathcal{C} = H'(e(t_{\mathcal{T}}(\alpha)P + R, \delta P))$. The left-hand-side yields

$$g^{(\overline{m}+\gamma r_1+\gamma r_2)\cdot\delta m+((1-\gamma)\cdot\delta r)\cdot m+(1-m)\cdot\delta r} =$$
$$g^{(\delta m\overline{m}+\gamma\delta rm)+(\delta rm-\gamma\delta rm)+(\delta r-\delta rm)} =$$
$$g^{\delta t+\delta r} = e(t_{\mathcal{T}}(\alpha)P + R, \delta P),$$

whereas we assume R_1, R_2 and R to be $r_1 P, r_2 P$ and rP for some unknown values r_1, r_2 and r, respectively. Furthermore, we write t, m and $\overline{m}$ instead of $t_{\mathcal{T}}(\alpha), m_{\mathcal{M}}(\alpha)$, and $m_{\overline{\mathcal{M}}}(\alpha)$, respectively. This completes the proof. □

A.3 Proof of Theorem 3 (Sketch)

Case T1 & T2. The proofs for these two cases are analogous to the proof in Appendix A.1, since A has exactly the same knowledge as in the previous proof. More precisely, this is due to the knowledge of all templates in the unforgeability game.

Case M1 & M2(b). The only difference to the power of the adversary A in the proof in Appendix A.1 is that A knows R_1 and $\mathsf{sk}_{\mathsf{proxy}}^{\mathsf{DSS}}$ beforehand. As we have shown in the proof in Appendix A.1, due to our strong adversary model, A can obtain R_1 anyway, which makes the situation identical to the unforgeability proof, with the exception that A has no longer to break the DSS under $\mathsf{sk}_{\mathsf{proxy}}^{\mathsf{DSS}}$. Therefore, the proofs are almost identical to the proofs in the unforgeability case. □

A.4 Proof of Theorem 4 (Sketch)

Let $\mathcal{T}$ be a template and let k be the depth of $\mathcal{T}$. Now, we assume that A only knows at most $k-1$ instantiations and, thus, A has not full knowledge of $\mathcal{T}$. What A can try to do is to guess the remaining choices based on $\mathcal{C}$ and the instantiation dependent values $\mathcal{C}_{\overline{\mathcal{M}}}$. However, note that γR_1 as well as R are random values with unknown discrete logarithms, which can not be computed using any values from signatures or public keys. Thus, the values $t_{\mathcal{T}}(\alpha)P + R$ and δP are uniformly random and, consequently, $e(t_{\mathcal{T}}(\alpha)P + R, \delta P)$ as well as $\mathcal{C} = H'(e(t_{\mathcal{T}}(\alpha)P + R, \delta P))$ are also uniformly random. Moreover, in the same fashion also $\mathcal{C}_{\overline{\mathcal{M}}}$ is a uniformly random value. As a consequence, $\mathcal{C}$ and $\mathcal{C}_{\overline{\mathcal{M}}}$ constitute unconditionally hiding Pedersen commitments. Note that the instantiation dependent value γ guarantees that the values γR_1 in the commitments $\mathcal{C}_{\overline{\mathcal{M}}}$ of different template instantiations differ. Thus, an adversary can gain no knowledge by using differences of distinct values $\mathcal{C}_{\overline{\mathcal{M}}}$. Note also that since $\mathcal{C}$ is a hash value this prevents A from performing arithmetic with values $\mathcal{C}_{\overline{\mathcal{M}}}$ and thereby gaining knowledge of $t_{\mathcal{T}}$. From this follows the privacy of Scheme 1. □