Secure Proxy Signature Schemes for Delegation of Signing Rights

Alexandra Boldyreva Adriana Palacio Bogdan Warinschi

Dept. of Computer Science & Engineering, University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92093, USA.
Email: {aboldyre,apalacio,bogdan}@cs.ucsd.edu
URL: http://www-cse.ucsd.edu/users/{aboldyre,apalacio,bogdan}

Abstract

A proxy signature scheme permits an entity to delegate its signing rights to another entity. These schemes have been suggested for use in numerous applications, particularly in distributed computing. But to date, no proxy signature schemes with guaranteed security have been proposed; no precise definitions or proofs of security have been provided for such schemes. In this paper, we formalize a notion of security for proxy signature schemes and present provably-secure schemes. We analyze the security of the well-known delegation-by-certificate scheme and show that after some slight but important modifications, the resulting scheme is secure, assuming the underlying standard signature scheme is secure. We then show that employment of the recently introduced aggregate signature schemes permits bandwidth and computational savings. Finally, we analyze the proxy signature scheme of Kim, Park and Won, which offers important performance benefits. We propose modifications to this scheme that preserve its efficiency, and yield a proxy signature scheme that is provably secure in the random-oracle model, under the discrete-logarithm assumption.

Keywords: Applied cryptography, digital signature schemes, proxy signature schemes, aggregate signature schemes, provable security.

Contents

1 Introduction

A proxy signature protocol allows an entity, called the *designator* or *original signer*, to delegate another entity, called a *proxy signer*, to sign messages on its behalf, in case of say, temporal absence, lack of time or computational power, etc. The delegated proxy signer can compute a *proxy signature* that can be verified by anyone with access to the original signer's certified public key.

Applications and background. Proxy signatures have found numerous practical applications, particularly in distributed computing where delegation of rights is quite common. Examples discussed in the literature include distributed systems [22, 33], Grid computing [7], mobile agent applications [11, 15], distributed shared object systems [18], global distribution networks [2], and mobile communications [24]. The proxy signature primitive and the first efficient solution were introduced by Mambo, Usuda and Okamoto [19]. Since then proxy signature schemes have enjoyed a considerable amount of interest from the cryptographic research community. New security considerations and constructions have been proposed, old schemes have been broken, followed by more constructions (e.g., [12, 34, 23, 31, 30, 35, 15, 8, 16, 32]). Furthermore, various extensions of the basic proxy signature primitive have been considered. These include threshold proxy signatures [12, 37, 28, 29, 10], blind proxy signatures [13], proxy signatures with warrant recovery [14], nominative proxy signatures [24], one-time proxy signatures [11], and proxy-anonymous proxy signatures [27].

Unfortunately, the extensive cryptographic research on the topic has brought developers more confusion than guidance because almost every other paper breaks some previously proposed construction, and proposes a new one. See [36, 17, 15, 16, 32] for some illustrative examples of this trial and error approach. Very few schemes were left unbroken, and none of them has provable-security guarantees. Typically, security of these schemes is argued by presenting attacks that fail, which provides only very weak guarantees. What is clearly desirable but has not been provided until now, is a proxy signature scheme with *guaranteed* security. In order to achieve this goal, it is necessary to first formalize a notion of security for proxy signature schemes, since the current security requirements are vague and ill-defined. This problem was recognized and left open in [16].

Our current work is aimed at filling this void. This is the first work on proxy signatures in the provable-security direction. We define a formal model for the security of proxy signature schemes, which enables the cryptographic analysis of such schemes. Then we present several examples of proxy signature schemes that *provably* satisfy this notion of security, under widely-believed computational-complexity assumptions.

Functionality of proxy signature schemes. As in previous works, we assume a Public Key Infrastructure (PKI) setting, where each entity holds a public and secret key pair. As usual, each user can sign messages using the signing algorithm of a standard digital signature scheme, and his or her secret key. When a user (the original signer) desires to delegate his or her signing ability to another user (the proxy signer), they run a possibly interactive *proxy-designation* protocol. Through a successful execution of this protocol, the proxy signer obtains a proxy signing key. It can then sign messages on behalf of the original signer using a *proxy signing* algorithm and the proxy signing key. Anyone can verify the validity of such signatures using a *proxy verification* algorithm and the original signer's public key.

Natural constructions. Before discussing the security of these schemes, we sketch a couple of natural constructions based on any standard digital signature scheme.

The most straightforward solution is for the designator to give its secret key to the proxy signer, who can then use it to sign messages. In this case proxy signatures are just standard signatures, and can be verified the usual way. This scheme, called *full delegation* in the literature, has several

shortcomings. Its security relies on the honesty of the proxy signer in a completely unrealistic manner. It provides no way to restrict signing rights to particular types of messages or a certain time period. Even if the proxy signer is fully trusted, this scheme increases the vulnerability of the designator's secret key. Additionally, it requires the establishment of a secure channel between the original signer and the proxy signer. Although most previous works assume a secure channel for the proxy-designation protocol, we find this requirement unnecessary and undesirable.

A second approach is known as *delegation by certificate* or *delegation by warrant*. Here the designator uses the signing algorithm of a standard signature scheme and its secret key to sign a *warrant*, which contains information regarding the particular proxy signer. For instance, the warrant may contain the proxy signer's public key, a period of validity, and restrictions on the class of messages for which the warrant is valid. We call the signature computed by the designator a *certificate*. The designator sends the proxy signer the warrant and the certificate. The proxy signer then uses its own secret key as the proxy signing key to sign messages on behalf of the designator. A proxy signature contains the warrant, the certificate and the proxy signer's signature. A verifier needs to ensure that the certificate is valid with respect to the public key of the designator, verify the second signature with respect to the public key of the proxy signer specified in the warrant, and make sure that the message signed conforms to the restrictions in the warrant. We show that after some slight but important modifications, the delegation-by-certificate scheme is in fact a secure proxy signature scheme, assuming the underlying standard signature scheme is secure.

A delegation-by-certificate proxy signature can be computed in roughly the same amount of time required for standard signing, but verification of such proxy signatures requires twice the time to verify a standard signature. Consider, for example, an RSA-signature-based delegation-by-certificate proxy signature scheme. Verification requires two modular exponentiations (i.e., two RSA-signature verifications). If a discrete-logarithm-based scheme such as Schnorr's signature scheme is used, verification of a proxy signature requires four modular exponentiations (i.e., two Schnorr-signature verifications). Most of the works on basic proxy signature schemes mentioned above focused on constructing a more efficient scheme, where verification of a proxy signature requires less time than verification of two standard signatures. Several such constructions were proposed, but they lack provable-security guarantees. In order to provide an efficient, provably-secure proxy signature scheme, we first pin down an appropriate formal notion of security.

SECURITY OF PROXY SIGNATURE SCHEMES. The security properties for proxy signature schemes introduced in [19] were somewhat enhanced by [15], and did not evolve much since then. The properties stated in [15] are the following.

Verifiability: From a proxy signature, a verifier can be convinced of the original signer's agreement on the signed message.

Strong unforgeability: The original signer and third parties who are not designated as proxy signers cannot create a valid proxy signature.

Strong identifiability: Anyone can determine the identity of the corresponding proxy signer from a proxy signature.

Strong undeniability: A proxy signer cannot repudiate a proxy signature it created.

Prevention of misuse: A proxy signing key cannot be used for purposes other than generating valid proxy signatures. In case of misuse, the responsibility of the proxy signer should be determined explicitly.[1]

[1]This property is not only ill-defined, but too strong. In particular, it seems to imply that delegation-by-certificate schemes are inherently insecure because in these schemes the proxy signing key is the proxy signer's secret key, used for standard signing. The attacks discussed in [15] that lead to the formulation of this property can be prevented without going to this extreme. Our model implicitly takes them into account.

While these informal requirements provide some intuition about the goals that a notion of security for proxy signature schemes should capture, their precise meaning is unclear. They do not specify what an attack is, leaving important questions unanswered. For instance, what are an adversary's capabilities? In particular, can malicious parties collude? Are attackers allowed to see or request signatures? Are they allowed to register keys? What exactly is the adversary's goal? When is an attacker considered successful?

We clarify these issues by designing a formal model for the security of proxy signature schemes. It involves a rather powerful adversary who is allowed to corrupt an arbitrary number of users and learn their secret keys. Moreover, the adversary can register public keys on behalf of new users, possibly obtained otherwise than running the key-generation algorithm, and possibly depending on the public keys of already registered users.[2] Furthermore, we allow the adversary to interact with honest users playing the role of a designator or a proxy signer. We also allow it to see transcripts of all executions of the proxy-designation protocol between honest users, i.e., we do not assume the existence of a secure channel between a designator and a proxy signer. The adversary can ask to see both standard signatures and proxy signatures generated by honest users on messages of the adversary's choice. We say that the adversary wins if it manages to create a standard signature or a proxy signature for a new message, i.e., a message that was not signed by an honest user. As is standard in modern cryptography, we characterize security of proxy signature schemes in terms of their resistance against attacks by probabilistic polynomial-time adversaries. We say that a proxy signature scheme is secure if no such adversary can win with probability non-negligible in the security parameter of the scheme. This security model, which is one of our main contributions, is detailed in Section 3.

Analysis of delegation-by-certificate proxy signature schemes. Having a well-defined notion of security in hand allowed us to identify some simple, previously overlooked attacks on delegation-by-certificate proxy signature schemes. We discuss these attacks in Section 4. We also suggest fixes that guarantee the security of delegation-by-certificate schemes based on secure standard signature schemes. Namely, it is crucial for security of these constructs that users prepend distinct symbols, say 11, 00, 01 to each message before signing it with the standard signing algorithm, depending on whether the user is generating a standard signature, signing a warrant for proxy designation, or generating a proxy signature on behalf of a designator. We also require that a proxy signer prepend the public key of the designator to a message before signing it on the designator's behalf. We prove that the resulting proxy signature scheme is secure according to our notion of security, assuming that the standard signature scheme is secure (i.e., existentially unforgeable under adaptive chosen message attacks [9]).

Aggregate-signature-based proxy signature schemes. Quite recently, Boneh et al. [5] introduced the concept of an aggregate signature scheme, which allows composition of a *single* signature out of signatures generated by several users for distinct messages. Using aggregate signatures it is possible to obtain an improvement over delegation-by-certificate proxy signature schemes in terms of both bandwidth and efficiency. In Section 5, we discuss a simple construction of a secure proxy signature scheme, given any secure aggregate signature scheme, such that a proxy signature consists of a warrant and a single aggregated signature.

The aggregate signature scheme presented in [5] is constructed from the co-GDH signature scheme based on bilinear maps [6], a scheme that produces short digital signatures. This bilinear aggregate signature scheme was proved secure in the random-oracle model [4], under the Computational co-Diffie-Hellman assumption. In the case of a proxy signature scheme based on the bilinear

[2] We assume, however, that during registration of public keys, all users prove knowledge [3] of the corresponding secret key, which is the recommended practice for certification authorities (e.g., [1, 21]).

aggregate signature scheme, the length of a proxy signature is the length of a warrant plus the length of a single short bilinear co-GDH signature. Verification of a proxy signature requires three bilinear map computations, whereas verification of a delegation-by-certificate proxy signature based on the bilinear co-GDH signature scheme requires four bilinear map computations.

The Triple Schnorr proxy signature scheme. Kim, Park, and Won [12] presented a proxy signature scheme (KPW) based on the Schnorr signature scheme [26] that is more efficient than the delegation-by-certificate scheme based on Schnorr signatures. The latter requires four modular exponentiations per proxy signature verification, while verification of a KPW proxy signature requires only three exponentiations. This improvement can be furthered by employing an algorithm for simultaneous multiple exponentiation. The cost of verification of a KPW proxy signature can be reduced to about 1.25 exponentiations, versus roughly 2.35 exponentiations for verification of a Schnorr-based delegation-by-certificate proxy signature. To the best of our knowledge, the KPW scheme is the only proxy signature scheme that remains unbroken, does not require a secure channel for proxy designation, and has this advantage in efficiency over the corresponding delegation-by-certificate scheme.

We discuss the KPW scheme in Section 6. We modify the scheme, preserving its efficiency, and prove that the resulting scheme is secure in the random-oracle model, assuming hardness of computation of discrete logarithms in the underlying group. This is our second main result. We call this scheme the Triple Schnorr proxy signature scheme since it uses Schnorr signatures for standard signing, proxy designation, and proxy signing. It is the first provably-secure proxy signature scheme with an advantage in efficiency of verification of more than 45% over the corresponding delegation-by-certificate scheme. Standard signing and proxy signing require approximately the same amount of time in both of these schemes.

2 Preliminaries

Notation. We denote by $\{0,1\}^*$ the set of all binary strings of finite length. For $N \in \mathbb{N}$, we let $[N] = \{1, \ldots, N\}$. If A is a possibly randomized algorithm, then the notation $x \xleftarrow{\$} A(x_1, x_2, \ldots)$ denotes that x is assigned the outcome of the experiment of running A on inputs $x_1, x_2, \ldots$. For strings $a_1, \ldots, a_n$, $a_1 || \cdots || a_n$ denotes an encoding such that the constituent strings are uniquely recoverable from the final one. Recall that a function $f : \mathbb{N} \to \mathbb{R}$ is called *negligible* if it approaches zero faster than the reciprocal of any polynomial, i.e., for any polynomial p, there exists $n_p \in \mathbb{N}$ such that for all $n \geq n_p$, $f(n) \leq 1/p(n)$.

Signature schemes. We recall the components of a digital signature scheme and the standard notion of security for such schemes, namely, existential unforgeability under adaptive chosen message attacks [9].

Definition 2.1 [Digital signature scheme] A digital signature scheme $\mathsf{DS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ is specified by four polynomial-time algorithms with the following functionalities.

- The randomized *parameter-generation* algorithm $\mathcal{G}$ takes input 1^k, where k is the security parameter, and outputs some global parameters params. These may contain, for example, a security parameter, the description of a cyclic group and a generator, and the description of a hash function. We assume that these parameters become publicly available.
- The randomized *key-generation* algorithm $\mathcal{K}$ takes input global parameters params and outputs a pair (pk, sk) consisting of a public key and a matching secret key, respectively.

- The (possibly) randomized *signing* algorithm $\mathcal{S}$ takes input a secret key sk and a message $M \in \{0,1\}^*$, and outputs a signature σ.
- The deterministic *verification* algorithm $\mathcal{V}$ takes input a public key pk, a message M and a candidate signature σ for M, and outputs a bit. We say that σ is a *valid* signature for M relative to pk if $\mathcal{V}(pk, M, \sigma) = 1$.

For any pair of keys (pk, sk) that might be output by $\mathcal{K}$ and any $M \in \{0,1\}^*$, it is required that $\mathcal{V}(pk, M, \mathcal{S}(sk, M)) = 1$.

Definition 2.2 [Security of a digital signature scheme] Let $\mathsf{DS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ be a digital signature scheme. Consider an adversary $\mathcal{A}$ that is given input a public key pk and access to a signing oracle $\mathcal{O}_{\mathcal{S}}(sk, \cdot)$, where pk and sk are matching keys generated via $\mathsf{params} \xleftarrow{\$} \mathcal{G}(1^k)$; $(pk, sk) \xleftarrow{\$} \mathcal{K}(\mathsf{params})$. The oracle takes input a message M and returns a signature $\sigma \xleftarrow{\$} \mathcal{S}(sk, M)$. $\mathcal{A}$ queries this oracle on messages of its choice, and eventually outputs a forgery (M, σ). The adversary's advantage in attacking the scheme is the probability that it outputs a pair (M, σ) such that σ is a valid signature for message M and this message was not queried to the signing oracle. DS is said to be *secure against existential forgery under adaptive chosen-message attacks (or simply, secure)* if the advantage of any poly(k)-time adversary $\mathcal{A}$ is negligible in the security parameter k.

3 Proxy Signature Schemes

THE SETTING. As discussed in the introduction, we consider a PKI-like setting: users are identified by natural numbers, and we let pk_i denote the public key of user $i \in \mathbb{N}$, and sk_i denote the corresponding secret key.

3.1 Syntax of Proxy Signature Schemes

A proxy signature scheme involves a digital signature scheme for standard signing, a protocol that users can run in order for one of them to designate the other as a proxy signer, a signing algorithm to be used by proxy signers (which, in general, can differ from the one used for standard signing), and a corresponding verification algorithm for proxy signatures. Additionally, the strong identifiability property mentioned in Section 1 suggests that there should be an algorithm that extracts the identity of the proxy signer from a proxy signature. This identity can be the natural number identifying the user, or equivalently, the user's public key. The following definition details the components of a proxy signature scheme.

Definition 3.1 [Proxy signature scheme] A *proxy signature scheme* is a tuple $\mathsf{PS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V}, (\mathcal{D}, \mathcal{P}), \mathcal{PS}, \mathcal{PV}, \mathcal{ID})$, where the constituent algorithms run in polynomial time, $\mathsf{DS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ is a digital signature scheme, and the other components are defined as follows.

- $(\mathcal{D}, \mathcal{P})$ is a pair of interactive randomized algorithms forming the (two-party) *proxy-designation protocol*. The input to each algorithm includes two public keys pk_i, pk_j for the *designator* i and the *proxy signer* j, respectively. $\mathcal{D}$ also takes as input the secret key sk_i of the designator, and $\mathcal{P}$ also takes as input the secret key sk_j of the proxy signer. As result of the interaction, the expected local output of $\mathcal{P}$ is skp, a *proxy signing key* that user j uses to produce proxy signatures on behalf of user i. $\mathcal{D}$ has no local output. We write $skp \xleftarrow{\$} [\mathcal{D}(pk_i, pk_j, sk_i), \mathcal{P}(pk_i, pk_j, sk_j)]$ for the result of this interaction.
- $\mathcal{PS}$ is the (possibly) randomized *proxy signing* algorithm. It takes input a proxy signing key skp and a message $M \in \{0,1\}^*$, and outputs a proxy signature $p\sigma$.

- $\mathcal{PV}$ is the deterministic *proxy verification* algorithm. It takes input a public key pk, a message M and a proxy signature $p\sigma$, and outputs 0 or 1. In the latter case, we say that $p\sigma$ is a *valid* proxy signature for M relative to pk.
- $\mathcal{ID}$ is the *proxy identification* algorithm. It takes input a valid proxy signature $p\sigma$, and outputs an identity, i.e., a public key, or $\perp$ in case of an error.

CORRECTNESS. We require that for all messages $M \in \{0,1\}^*$ and all users $i, j \in \mathbb{N}$, if skp is a proxy signing key for user j on behalf of user i, i.e., $skp \xleftarrow{\$} [\mathcal{D}(pk_i, pk_j, sk_i), \mathcal{P}(pk_i, pk_j, sk_j)]$, then $\mathcal{PV}(pk_i, M, \mathcal{PS}(skp, M)) = 1$ and $\mathcal{ID}(\mathcal{PS}(skp, M)) = pk_j$. Informally, this means that signatures produced with proxy signing keys are valid relative to the public key of the designator, and that the identity of the proxy signer can be extracted from proxy signatures that it produces. Notice that no secret keys are involved in the identification process, i.e., we model the case when the identification algorithm can be run by anyone.

3.2 A Notion of Security for Proxy Signature Schemes

SOME INTUITION. We first informally describe some of the features of our adversarial model. We allow the adversary to corrupt users and learn their secret keys. It can also add new users and register public keys for them. These keys do not have to have the distribution defined by the key-generation algorithm. In principle, they can even depend on the public keys of honest users. In order to prevent these rogue-key attacks, we require all users to prove knowledge of the associated secret key during key registration, as is commonly recommended to be done in practice [1, 21]. To capture this, the adversary is required to output *both* the public key and the corresponding secret key for a new user. Alternatively, we could explicitly consider the key-registration process in the model, and, in proofs of security, use the extractors guaranteed by the proof of knowledge property [3] to extract the secret keys from the adversary. We use the former approach for simplicity.

We model a seemingly extreme case in which the adversary is working against a *single* honest user, say user 1, and can select and register keys for all other users. Notice that this is without loss of generality since any attack that can be carried out in the presence of more honest users, can be performed by having some of the users under the control of the adversary behave honestly. The adversary can play the role of user $i \neq 1$ in executions of the proxy-designation protocol with user 1, as designator or as proxy signer. In both cases, the adversary may behave dishonestly in an attempt to obtain information from the honest user. We comment that there is no restriction on the number of executions of the proxy-designation protocol between the same two users. We do not rule out the possibility that a user designate himself (which may be useful, for example, to create a temporary key for use in a hostile environment), so we let the adversary request user 1 to run the proxy-designation protocol with himself, and see the transcript of the execution. We emphasize that we do not assume the existence of a secure channel between a designator and a proxy signer.

We model chosen-message attack capabilities by providing the adversary access to two oracles: a standard signing oracle and a proxy signing oracle. The first oracle takes input a message M, and returns a standard signature for M by user 1. The second oracle takes input a tuple (i, l, M), and, if user 1 was designated by user i at least l times, returns a proxy signature for M created by user 1 on behalf of user i, using the l-th proxy signing key.

The goal of the adversary is to produce one of the following forgeries: (1) a standard signature by user 1 for a message that was not submitted to the standard signing oracle, (2) a proxy signature for a message M by user 1 on behalf of some user i such that either user i never designated user 1, or M was not in a query (i, l, M) made to the proxy signing oracle, or (3) a proxy signature for a message M by some user i on behalf of user 1, such that user i was never designated by user 1.

Our notion of security for proxy signature schemes is formally defined as follows.

Definition 3.2 [Security of a proxy signature scheme] Let $\mathsf{PS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V}, (\mathcal{D}, \mathcal{P}), \mathcal{PS}, \mathcal{PV}, \mathcal{ID})$ be a proxy signature scheme. Consider an experiment $\mathbf{Exp}^{\text{ps-uf}}_{\mathsf{PS},\mathcal{A}}(k)$ related to scheme PS, adversary $\mathcal{A}$, and security parameter k. First, system parameters params are generated by running $\mathcal{G}$ on input 1^k. Then a public and secret key pair for user 1 is generated via $(pk_1, sk_1) \xleftarrow{\$} \mathcal{K}(\mathsf{params})$. A counter n for the number of users is initialized to 1, and an empty array $\mathbf{skp}_1$ and an empty set D are created. Adversary $\mathcal{A}$ is given input pk_1, and it can make the following requests or queries, in any order and any number of times.

- (i registers pk_i) $\mathcal{A}$ can request to register a public key pk_i for user $i = n+1$ by outputting a pair (pk_i, sk_i) of matching public and secret keys. These keys are stored, counter n is incremented, and an empty array $\mathbf{skp}_i$ is created.
- (1 designates i) $\mathcal{A}$ can request to interact with user 1 running $\mathcal{D}(pk_1, pk_i, sk_1)$, for some $i \in \{2, \ldots, n\}$, and play the role of user i running $\mathcal{P}(pk_1, pk_i, sk_i)$; after a successful run, D is set to $D \cup \{pk_i\}$.
- (i designates 1) $\mathcal{A}$ can request to interact with user 1 running $\mathcal{P}(pk_i, pk_1, sk_1)$, for some $i \in \{2, \ldots, n\}$, and play the role of user i running $\mathcal{D}(pk_i, pk_1, sk_i)$. The private output skp of $\mathcal{P}$ is stored in the last unoccupied position of $\mathbf{skp}_i$. We emphasize that $\mathcal{A}$ does not have access to the elements of $\mathbf{skp}_i$.
- (1 designates 1) $\mathcal{A}$ can request that user 1 run the designation protocol with itself, and see the transcript of the interaction. The private output skp of user 1 is stored in the next available position in $\mathbf{skp}_1$. $\mathcal{A}$ does not have access to the elements of $\mathbf{skp}_1$.
- (standard signature by 1) $\mathcal{A}$ can query oracle $\mathcal{O}_{\mathcal{S}}(sk_1, \cdot)$ with a message M and obtain a standard signature for M by user 1, $\sigma \xleftarrow{\$} \mathcal{S}(sk_1, M)$.
- (proxy signature by 1 on behalf of i using the l-th proxy signing key) $\mathcal{A}$ can make a query (i, l, M), where $i \in [n]$, $l \in \mathbb{N}$, and $M \in \{0,1\}^*$, to oracle $\mathcal{O}_{\mathcal{PS}}((\mathbf{skp}_u)_{u\in[n]}, \cdot, \cdot, \cdot)$. If key $\mathbf{skp}_i[l]$ has already been defined, we say the query is *valid* and the oracle returns $\mathcal{PS}(\mathbf{skp}_i[l], M)$; if $\mathbf{skp}_i[l]$ has not been defined, the query is said to be *invalid* and the oracle returns $\perp$.

Eventually, $\mathcal{A}$ outputs a forgery (M, σ) or $(M, p\sigma, pk)$. The output of the experiment is determined as follows:

1. If the forgery is of the form (M, σ), where $\mathcal{V}(pk_1, M, \sigma) = 1$, and M was not queried to oracle $\mathcal{O}_{\mathcal{S}}(sk_1, \cdot)$, then return 1. [forgery of a standard signature]
2. If the forgery is of the form $(M, p\sigma, pk)$, where $pk = pk_i$ for some $i \in [n]$, $\mathcal{PV}(pk_i, M, p\sigma) = 1$, $\mathcal{ID}(p\sigma) = pk_1$, and no valid query (i, l, M), for $l \in \mathbb{N}$, was made to $\mathcal{O}_{\mathcal{PS}}((\mathbf{skp}_u)_{u\in[n]}, \cdot, \cdot, \cdot)$, then return 1. [forgery of a proxy signature by user 1 on behalf of user i]
3. If the forgery is of the form $(M, p\sigma, pk_1)$, where $\mathcal{PV}(pk_1, M, p\sigma) = 1$, and $\mathcal{ID}(p\sigma) \notin D \cup \{pk_1\} \cup \{\perp\}$, then return 1. [forgery of a proxy signature by user $i \neq 1$ on behalf of user 1; user i was not designated by user 1]
4. Otherwise, return 0

We define the *advantage* of adversary $\mathcal{A}$ as

$$\mathbf{Adv}^{\text{ps-uf}}_{\mathsf{PS},\mathcal{A}}(k) \;=\; \Pr\left[\, \mathbf{Exp}^{\text{ps-uf}}_{\mathsf{PS},\mathcal{A}}(k) = 1 \,\right].$$

We adopt the convention that the *time complexity* of adversary $\mathcal{A}$ is the execution time of the entire experiment, including the time taken for parameter and key generation, and computation of answers

to oracle queries. We say that PS is a *secure proxy signature scheme* if the function $\mathbf{Adv}^{\text{ps-uf}}_{\mathsf{PS},\mathcal{A}}(\cdot)$ is negligible for all adversaries $\mathcal{A}$ of time complexity polynomial in the security parameter k. ∎

4 Delegation-by-certificate proxy signature schemes

Since the introduction of the proxy signature primitive [19], it has been believed that proxy signature schemes can be securely constructed from any digital signature scheme, in a very simple way. Informally, a user i (with public and secret key pair (pk_i, sk_i)) can delegate its signing capability to user j (with keys pk_j, sk_j) by sending it an appropriate *warrant* and a *certificate*, which is a signature for the warrant under key sk_i. A warrant is a message containing the public key pk_j of the designated proxy signer j and, possibly, restrictions on the messages the proxy signer is allowed to sign. These restrictions depend on the application and may include periods of validity, specifications about the types of messages allowed, etc. Once designated, user j can create a proxy signature for a message M by computing a signature for M under its secret key sk_j and concatenating this signature with the warrant and the certificate. Verification of a proxy signature involves verifying the validity of the certificate relative to pk_i, verifying the validity of the signature for M relative to pk_j, and checking that M conforms to the restrictions, if any, specified in the warrant. We will not mention this final check explicitly again, since its details are specific to each application, but is important to keep in mind that verification includes this step.

As we mentioned in Section 1, the "raison d'être" of most of the previous work on proxy signature schemes was to improve on the efficiency of the delegation-by-certificate solution. As a consequence, its details have never been properly pinned down. Nevertheless, we believe discussing this scheme in some detail is important since its conceptual simplicity and generality makes it convenient for implementation in applications requiring the functionality of proxy signatures. We first note various weaknesses of a naive implementation of the scheme. Then, we propose appropriate fixes and show that the resulting scheme is indeed secure.

FLAWS AND FIXES. Given a standard digital signature created by user j for message M, a malicious user i can transform this signature into a proxy signature by user j for message M on behalf of user i. For this it is sufficient for i to sign a warrant certifying that j can produce signatures on its behalf. These two signatures together with the warrant comprise a proxy signature valid relative to the public key of user i. This and similar problems can be fixed by introducing a way to differentiate between signatures created for standard signing, proxy designation, and proxy signing. For instance, in order to sign a message M, the user actually signs message $11||M$, whereas in order to sign a warrant ω during the designation process, the user actually signs $00||\omega$, and for proxy signing of a message M, the proxy signer signs $01||M$. Verification of standard (resp., proxy) signatures is then performed by first prepending 11 (resp., 01) to the message.

This fix is insufficient. The resulting delegation-by-certificate proxy signatures still have some undesirable malleability properties. Indeed, given a single proxy signature created by user j on behalf of user i for a message M, it is possible to obtain a proxy signature for the same message by user j on behalf of some malicious user k. For this, k removes the warrant (and corresponding certificate) stating that j can sign on behalf of i, and replaces it with one stating that j is allowed to sign on behalf of k. The result is a proxy signature by user j on behalf of user k, which user j did not produce. Again, there is a simple fix: to sign on behalf of user i, instead of signing message $01||M$, user j signs message $01||pk_i||M$. This ties the part of the proxy signature created by the proxy signer to the designator.

We are now ready to summarize the above discussion by presenting the construction of a delegation-by-certificate proxy signature scheme from any digital signature scheme.

Construction 4.1 Let $\mathsf{DS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ be a signature scheme. The algorithms of the corresponding delegation-by-certificate proxy signature scheme $\mathsf{PS}[\mathsf{DS}] = (\mathcal{G}_1, \mathcal{K}_1, \mathcal{S}_1, \mathcal{V}_1, (\mathcal{D}, \mathcal{P}), \mathcal{PS}, \mathcal{PV}, \mathcal{ID})$ are defined as follows:

- The parameter and key generation algorithms stay the same: $\mathcal{G}_1 = \mathcal{G}$, $\mathcal{K}_1 = \mathcal{K}$.
- A standard signature for message M is obtained by prepending 11 to the message, and signing the result using $\mathcal{S}$, i.e., $\mathcal{S}_1(sk, M) = \mathcal{S}(sk, 11||M)$.
- The verification of a signature σ for message M is done by computing $\mathcal{V}_1(pk, M, \sigma) = \mathcal{V}(pk, 11||M, \sigma)$.
- User i, in order to designate user j as a proxy signer, simply sends to j an appropriate warrant ω together with a signature cert for message $00||\omega$, under the secret key of user i. Since the warrant includes the public key of the designated proxy signer, for clarity, we make it explicit and require cert to be a signature for message $00||pk_j||\omega$. The corresponding proxy signing key of j is $skp = (sk_j, pk_i, pk_j||\omega, \mathsf{cert})$.
- A proxy signature for message M produced by user j on behalf of user i, contains a warrant ω, the public key of the proxy signer pk_j, the certificate cert (a signature for $00||pk_j||\omega$ under sk_i) and a signature for $01||pk_i||M$ under sk_j. Formally,
$$\mathcal{PS}((sk_j, pk_i, pk_j||\omega, \mathsf{cert}), M) \;=\; (\omega, pk_j, \mathsf{cert}, \mathcal{S}(sk_j, 01||pk_i||M)).$$
- Proxy signature verification is defined via
$$\mathcal{PV}(pk, M, (\omega, pk', \mathsf{cert}, \sigma)) \;=\; \mathcal{V}(pk, 00||pk'||\omega, \mathsf{cert}) \wedge \mathcal{V}(pk', 01||pk||M, \sigma).$$
- The identification algorithm is simply defined as $\mathcal{ID}((\omega, pk', \sigma)) = pk'$.

The following theorem states our result about the security of proxy signature scheme $\mathsf{PS}[\mathsf{DS}]$.

Theorem 4.2 Let DS be a secure digital signature scheme. Then the scheme $\mathsf{PS}[\mathsf{DS}]$ specified in Construction 4.1 is a secure proxy signature scheme.

This result follows from a theorem stated in Section 5.2 and is proved there.

5 Aggregate-signature-based proxy signature schemes

As we mentioned in Section 1, aggregate signatures can be used to optimize the length, and possibly the verification time of delegation-by-certificate proxy signatures. In this section we treat this matter in more detail. We first briefly review the aggregate signature primitive, its security and existing schemes. We refer the reader to [5] for details. We then show how aggregate signatures can be used to construct proxy signature schemes.

5.1 Aggregate signatures, their security and constructions

Aggregate signature schemes [5], allow construction of a single signature for a sequence of messages, out of signatures generated by distinct users for each of these messages. Formally, an aggregate signature scheme is given by a tuple of algorithms $\mathsf{AS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V}, \mathcal{A}, \mathcal{AV})$, where $\mathsf{DS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ is a standard digital signature scheme, called the *base* signature scheme, $\mathcal{A}$ is the *aggregation* algorithm, and $\mathcal{AV}$ is the *aggregate verification* algorithm. The aggregation algorithm takes input a sequence of public keys, messages and signatures $pk_1, \ldots, pk_n, M_1, \ldots, M_n, \sigma_1, \ldots, \sigma_n$, where each signature σ_i is valid for M_i relative to public key pk_i, and outputs a *single* aggregate signature $a\sigma$. The aggregate

verification algorithm takes input a sequence of public keys and messages $pk_1, \ldots, pk_n, M_1, \ldots, M_n$ and an aggregate signature $a\sigma$ and outputs a bit. It is required that

$$\mathcal{AV}(pk_1, \ldots, pk_n, M_1, \ldots, M_n, \mathcal{A}(pk_1, \ldots, pk_n, M_1, \ldots, M_n, \mathcal{S}(sk_1, M_1), \ldots, \mathcal{S}(sk_n, M_n))) = 1.$$

Security of aggregate signature schemes. The security of aggregate signatures is defined via an experiment $\mathbf{Exp}^{\text{ag-uf}}_{\mathsf{AS},\mathcal{B}}(k)$ related to scheme AS, adversary $\mathcal{B}$ and security parameter k. First, system parameters params are generated by running $\mathcal{G}$ on input 1^k. Then, a public and secret key pair (pk_1, sk_1) is selected by running the key-generation algorithm $\mathcal{K}$ on input params, and pk_1 is given as input to $\mathcal{B}$. Furthermore, $\mathcal{B}$ is provided with access to the signing oracle $\mathcal{S}(sk_1, \cdot)$. The goal of the adversary is to output a forgery, i.e., $n-1$ additional public keys $pk_2, \ldots, pk_n$, for some $n \geq 1$, a sequence of messages $M_1, \ldots, M_n$, and an aggregate signature $a\sigma$ for these messages. The experiment returns 1 (in which case we say that $\mathcal{B}$ wins) if $\mathcal{AV}(pk_1, \ldots, pk_n, M_1, \ldots, M_n, a\sigma) = 1$ and $\mathcal{B}$ did not submit M_1 to the signing oracle. The advantage of adversary $\mathcal{B}$ is defined by

$$\mathbf{Adv}^{\text{ag-uf}}_{\mathsf{AS},\mathcal{B}}(k) = \Pr\left[\, \mathbf{Exp}^{\text{ag-uf}}_{\mathsf{AS},\mathcal{B}}(k) = 1 \,\right],$$

and aggregate signature scheme AS is said to be secure if the function $\mathbf{Adv}^{\text{ag-uf}}_{\mathsf{AS},\mathcal{B}}(\cdot)$ is negligible for any poly(k)-time adversary $\mathcal{B}$.

Constructions of aggregate signature schemes. Note that any secure standard signature scheme DS yields a secure trivial aggregate signature scheme, in which the aggregation algorithm simply concatenates all signatures, and the aggregate verification algorithm simply verifies all signatures. We will refer to the aggregate signature scheme constructed this way as $\mathsf{TAS}[\mathsf{DS}]$. It is easy to see that $\mathsf{TAS}[\mathsf{DS}]$ is secure given that the underlying signature scheme DS is secure.

Boneh et al. [5] introduced a bilinear aggregate signature scheme which works in Gap Diffie-Hellman (GDH) groups (where the Decisional Diffie-Hellman problem is believed to be hard but the Computational Diffie-Hellman problem is easy) and which are also equipped with an additional bilinear map. The construction is based on the co-GDH signature scheme [6], which allows to produce short signatures. The security of the bilinear aggregate signature scheme is proved in the random-oracle model, assuming hardness of the Computational co-Diffie-Hellman assumption.

5.2 Aggregate-signature-based proxy signature schemes

We sketch the construction of a proxy signature scheme from any aggregate signature scheme.

Construction 5.1 Let $\mathsf{AS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V}, \mathcal{A}, \mathcal{AV})$ be an aggregate signature scheme. The algorithms of the corresponding proxy signature scheme $\mathsf{PS}[\mathsf{AS}] = (\mathcal{G}_1, \mathcal{K}_1, \mathcal{S}_1, \mathcal{V}_1, (\mathcal{D}, \mathcal{P}), \mathcal{PS}, \mathcal{PV}, \mathcal{ID})$ are defined as follows.

- The algorithms $\mathcal{G}_1, \mathcal{K}_1, \mathcal{S}_1, \mathcal{V}_1, (\mathcal{D}, \mathcal{P})$ use the algorithms $\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V}$ in the same way as those in Construction 4.1.
- The proxy signing algorithm $\mathcal{PS}$ uses $\mathcal{A}$ to aggregate the certificate and a proxy signature as follows:

$$\begin{aligned}\mathcal{PS}((sk_j, pk_i, pk_j||\omega, \mathsf{cert}), M) \;=\; \\ (\omega, pk_j, \mathcal{A}(pk_i, pk_j, 00||pk_j||\omega, 01||pk_i||M, \mathsf{cert}, \mathcal{S}(sk_j, 01||pk_i||M))).\end{aligned}$$

- The proxy verification algorithm $\mathcal{PV}$ is defined by

$$\mathcal{PV}(pk, M, (\omega, pk', a\sigma)) \;=\; \mathcal{AV}(pk, pk', 00||pk'||\omega, 01||pk||M, a\sigma).$$

- The identification algorithm is defined by $\mathcal{ID}((\omega, pk', a\sigma)) = pk'$.

The following theorem formally relates the security of the above construction to the security of the base aggregate signature scheme.

Theorem 5.2 Let AS be a secure aggregate signature scheme. Then the scheme PS[AS] defined above is a secure proxy signature scheme.

The proof of Theorem 5.2 is in Appendix A. We now use this result to prove Theorem 4.2.

Proof of Theorem 4.2: Let TAS[DS] be the trivial aggregate signature scheme defined in Section 5.1. As we mentioned there, it is secure if DS is secure. PS[TAS[DS]] as defined by Construction 5.1 is exactly the delegation-by-certificate scheme PS[DS] as per Construction 4.1. Therefore, Theorem 5.2 implies that PS[DS] is secure. ▮

A PROXY SIGNATURE SCHEME WITH SHORT SIGNATURES. As we mentioned before, the bilinear aggregate signature scheme was proved secure in the random-oracle model, assuming hardness of the Computational co-Diffie-Hellman assumption. This together with Theorem 5.2 imply that under the same assumptions, the proxy signature scheme obtained from the bilinear aggregate signature as per Construction 5.1 is a secure proxy signature scheme.

The length of the corresponding proxy signature is the length of the warrant plus the length of one short co-GDH signature. The use of the bilinear aggregate signature scheme also permits computational savings since the verification of a proxy signature requires 3 bilinear map computations versus 4 in the verification of a proxy signature in the co-GDH-signature-based delegation-by-certificate solution.

6 The Triple Schnorr scheme and its security

The proxy signature scheme proposed by Kim, Park, and Won [12] (KPW) employs Schnorr's signature scheme [26] for standard signing and for delegation, and allows the use of any signature scheme based on the hardness of the DLP for generation of proxy signatures. We make slight modifications to the version of the KPW scheme in which Schnorr's signature scheme is also used for proxy signing, and prove that the resulting proxy signature scheme is secure in the random-oracle model, under the assumption of hardness of the DLP. We call this provably-secure scheme Triple Schnorr. We remark that our modifications do not affect the length of the signatures produces nor do they have a significant impact on performance. We first recall the Schnorr digital signature scheme.

6.1 The Schnorr signature scheme.

On input 1^k, the parameter-generation algorithm $\mathcal{G}$ outputs primes p, q such that $2^{k-1} \leq p < 2^k$ (p is k bits long) and q divides $p-1$, an element $g \in \mathbb{Z}_p^*$ of order q, and a hash function $G : \{0,1\}^* \rightarrow \mathbb{Z}_q$. The key-generation algorithm $\mathcal{K}$, on input (p, q, g, G), selects a random $x \in \mathbb{Z}_q$, computes $X \leftarrow g^x \bmod p$, and outputs the pair $((p, q, g, G, X), (p, q, g, G, x))$ of public and secret keys. To simplify the notation, we will assume that the values p, q, g, G are available to all parties and we will not include them explicitly in the public and secret keys (i.e., the public and secret keys will simply be X and x, respectively).

To sign a message M, the signing algorithm $\mathcal{S}$ performs the following operations.

- Pick a random $y \in \mathbb{Z}_q$
- Compute a *commitment* $Y \leftarrow g^y \bmod p$

- Compute a *challenge* $c \leftarrow G(M, Y)$
- Compute $s \leftarrow y + c \cdot x \bmod q$
- Output (Y, s) as the signature of M

To verify a signature $(\overline{Y}, \bar{s})$ for message M, the verification algorithm $\mathcal{V}$ performs the following operations.

- Compute the challenge $c \leftarrow G(M, \overline{Y})$
- If $g^{\bar{s}} \equiv \overline{Y} \cdot X^c \pmod p$ then output 1 else output 0

The Schnorr signature scheme is known to be provably-secure in the random-oracle model, assuming hardness of the DLP [25]. In the sequel, $\mathsf{S} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ denotes the Schnorr scheme. We use the notation $\mathcal{S}^G$, $\mathcal{V}^G$ to emphasize that the hash function used in the scheme is G.

6.2 The Triple Schnorr proxy signature scheme.

In the following description, additions to the KPW scheme are underlined. Other modifications are specified below. The Triple Schnorr scheme is the proxy signature scheme $\mathsf{TS} = (\mathcal{G}_\mathsf{T}, \mathcal{K}_\mathsf{T}, \mathcal{S}_\mathsf{T}, \mathcal{V}_\mathsf{T}, (\mathcal{D}, \mathcal{P}), \mathcal{PS}, \mathcal{PV}, \mathcal{ID})$ whose constituent algorithms are defined as follows.

- The parameter-generation algorithm $\mathcal{G}_\mathsf{T}$ runs the Schnorr scheme parameter-generation algorithm $\mathcal{G}$ to get (p, q, g, G). It generates a hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$, and outputs $(p, q, g, G, \underline{H})$.
- On input (p, q, g, G, H), the key-generation algorithm $\mathcal{K}_\mathsf{T}$ runs the Schnorr scheme key-generation algorithm $\mathcal{K}$ on (p, q, g, G) to get $((p, q, g, G, X), (p, q, g, G, x))$, and outputs $((p, q, g, G, \underline{H}, X), (p, q, g, G, \underline{H}, x))$. Again, we will assume that the values p, q, g, G, H are available to all parties, and the public and secret keys will simply be X and x, respectively.
- To sign a message M, the signing algorithm first prepends a 1 to the message, and then runs the Schnorr signing algorithm with hash function G, on the result, i.e., $\mathcal{S}_\mathsf{T}(sk, M) = \mathcal{S}^G(sk, \underline{1}||M)$.
- To verify a signature σ for message M, the verification algorithm first prepends a 1 to the message, and then runs the Schnorr verification algorithm with hash function G, on the result, i.e., $\mathcal{V}_\mathsf{T}(pk, M, \sigma) = \mathcal{V}^G(pk, \underline{1}||M, \sigma)$.
- In order to designate user j as a proxy signer, user i sends user j an appropriate warrant ω and a certificate cert that is a Schnorr signature with hash function G for message $\underline{0}||\omega$ under the secret key sk_i of user i. We require that the warrant include the public keys of the designator and the proxy signer. For clarity, we make these keys explicit and require user i to sign the message $0||pk_i||pk_j||\omega$, i.e., $\mathsf{cert} = \mathcal{S}^G(sk_i, \underline{0||pk_i||pk_j}||\omega) = (Y, s)$. User j verifies this signature, and if it is valid, he computes a proxy signing key as $skp = (\underline{pk_i||pk_j}||\omega, Y, t)$, where $t = G(\underline{0||pk_i||pk_j}||\omega, Y) \cdot sk_j + s \bmod q$. See Figure 1.
- A proxy signature for message M, on behalf of user i, produced by user j (with proxy signing key $(pk_i||pk_j||\omega, Y, t)$), contains the warrant ω, the delegation commitment Y, the public key of the proxy signer pk_j, and a Schnorr signature with hash function H for message $0||M||pk_i||pk_j||\omega||Y$ under key t. Formally,

$$\mathcal{PS}((pk_i||pk_j||\omega, Y, t), M) \;=\; (\omega, Y, pk_j, \mathcal{S}^H(t, \underline{0}||M||\underline{pk_i||pk_j}||\omega||Y))$$

  In the version of the KPW scheme in which Schnorr's signature scheme is also used for proxy signing, the proxy signing (resp., verification) algorithm uses the Schnorr signing (resp., verification) algorithm with hash function G. The scheme is also provably secure in this case, but in order to simplify the proof, we chose to use an independent hash function H for proxy signing.

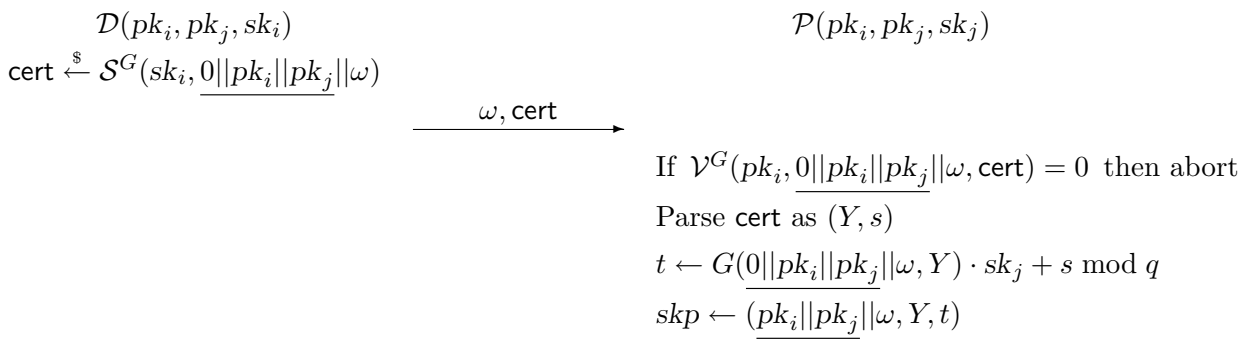

Figure 1: The Triple Schnorr designation protocol run by user i as designator, and user j as proxy signer.

- To verify a proxy signature (ω, Y, pk', σ) for message M with public key pk, the proxy verification algorithm computes a proxy public key as $pkp = (pk \cdot pk')^{G(\underline{0||pk||pk'}||\omega,Y)} \cdot Y \bmod p$, and then runs the Schnorr verification algorithm with hash function H, on the computed key pkp, message $0||M||pk||pk'||\omega||Y$, and signature σ, i.e.,
$$\mathcal{PV}(pk, M, (\omega, Y, pk', \sigma)) \;=\; \mathcal{V}^H(pkp, \underline{0}||M||\underline{pk||pk'||\omega||Y}, \sigma)$$
- The proxy identification algorithm is defined as $\mathcal{ID}((\omega, Y, pk', \sigma)) = pk'$.

We observe that verification of a Triple Schnorr proxy signature requires three exponentiations modulo p. Using a simultaneous multiple exponentiation algorithm such as Algorithm 14.88 in [20], the three exponentiations can be computed at a cost of about 1.25 exponentiations. This is a significant improvement over the Schnorr-based delegation-by-certificate scheme, for which verification (using simultaneous multiple exponentiation) requires roughly 2.35 exponentiations. Standard signing and proxy signing require approximately the same amount of time in the Triple Schnorr scheme as in the Schnorr-based delegation-by-certificate scheme. Proxy designation requires one additional modular multiplication to compute the proxy signing key in the Triple Schnorr scheme.

Security of Triple Schnorr. The following theorem states our result about the security of the Triple Schnorr proxy signature scheme in the random-oracle model.

Theorem 6.1 Let $\mathsf{TS} = (\mathcal{G}_\mathsf{T}, \mathcal{K}_\mathsf{T}, \mathcal{S}_\mathsf{T}, \mathcal{V}_\mathsf{T}, (\mathcal{D}, \mathcal{P}), \mathcal{PS}, \mathcal{PV}, \mathcal{ID})$ be the Triple Schnorr scheme defined above. If the DLP is hard then TS is a secure proxy signature scheme in the random-oracle model.

The proof of Theorem 6.1 is in Appendix B.

7 Acknowledgements

The authors are grateful to Mihir Bellare for valuable discussions. They also thank David Pointcheval for clarifications on [25].

References

[1] C. Adams and S. Farrell. Internet x.509 public key infrastructure: Certificate management protocols. RFC 2510, March 1999.

[2] A. Bakker, M. Steen, and A. S. Tanenbaum. A law-abiding peer-to-peer network for free-software distribution. In *IEEE International Symposium on Network Computing and Applications (NCA'01)*, 2001.

[3] M. Bellare and O. Goldreich. On defining proofs of knowledge. In E. Brickell, editor, *Proceedings of Advances in Cryptology – Crypto'92*, volume 740 of *LNCS*, pages 390–420. Springer-Verlag, 1992.

[4] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, LNCS. Springer-Verlag, 1993.

[5] D. Boneh, C. Gentry, H. Shacham, and B. Lynn. Aggregate and verifiably encrypted signatures from bilinear maps. In E. Biham, editor, *Proceedings of Advances is Cryptology – Eurocrypt'03*, LNCS. Springer-Verlag, 2003. To appear.

[6] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In C. Boyd, editor, *Asiacrypt '01*, volume 2248 of *LNCS*. Springer Verlag, 2001.

[7] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. In *Fifth ACM Conference on Computers and Communications Security*, 1998.

[8] H. Ghodosi and J. Pieprzyk. Repudiation of cheating and non-repudiation of Zhang's proxy signature schemes. In *LNCS*, volume 1587. Springer-Verlag, 2001.

[9] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17(2):281–308, April 1988.

[10] J. Herranz and G. Sez. Verifiable secret sharing for general access structures, with application to fully distributed proxy signatures. In *Proceedings of Financial Cryptography 2003*, LNCS. Springer-Verlag, 2003.

[11] H. Kim, J. Baek, B. Lee, and K. Kim. Secret computation with secrets for mobile agent using one-time proxy signature. In *Cryptography and Information Security 2001*, 2001.

[12] S. Kim, S. Park, and D. Won. Proxy signatures, revisited. In Y. Han, T. Okamoto, and S. Quing, editors, *Proceedings of International Conference on Information and Communications Security (ICICS)'97*, volume 1334 of *LNCS*, pages 223–232. Springer-Verlag, 1997.

[13] S. Lal and A. K. Awasthi. Proxy blind signature scheme. *Cryptology ePrint Archive, Report 2003/072. Available at* `http://eprint.iacr.org/`, 2003.

[14] S. Lal and A. K. Awasthi. A scheme for obtaining a warrant message from the digital proxy signatures. *Cryptology ePrint Archive, Report 2003/073. Available at* `http://eprint.iacr.org/`, 2003.

[15] B. Lee, H. Kim, and K. Kim. Strong proxy signature and its applications. In *Proceedings of SCIS*, 2001.

[16] J. Lee, J. Cheon, and S. Kim. An analysis of proxy signatures: Is a secure channel necessary? In M. Joye, editor, *Topics in Cryptology–CT-RSA'03*, volume 2612 of *LNCS*, pages 68–79. Spinger-Verlag, 2003.

[17] N.-Y. Lee, T. Hwang, and C.-H. Wang. On zhang's nonrepudiable proxy signature schemes. In *Information Security and Privacy, Third Australasian Conference, ACISP'98*, LNCS, pages 415–422, 1999.

[18] J. Leiwo, C. Hanle, P. Homburg, and A. S. Tanenbaum. Disallowing unauthorized state changes of distributed shared objects. In *SEC*, pages 381–390, 2000.

[19] M. Mambo, K. Usuda, and E. Okamoto. Proxy signatures for delegating signing operation. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security (CCS)*, pages 48–57. ACM, 1996.

[20] A. Menezes, P. C. van Oorschot, and S. Vanstone. *Handbook of Apllied Cryptography.* CRC Press, 1997.

[21] M. Meyers, C. Adams, D. Solo, and D. Kemp. Internet x.509 certificate request message format. RFC 2511, March 1999.

[22] B. C. Neuman. Proxy based authorization and accounting for distributed systems. In *Proceedings of the 13th International Conference on Distributed Computing Systems*, pages 283–291, 1993.

[23] T. Okamoto, M. Tada, and E. Okamoto. Extended proxy signatures for smart cards. In *LNCS*, volume 1729 of *LNCS*. Springer-Verlag, 1999.

[24] H.-U. Park and L.-Y. Lee. A digital nominative proxy signature scheme for mobile communications. In *ICICS 2001*, volume 2229 of *LNCS*, pages 451–455. Springer-Verlag, 2001.

[25] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–369, 2000.

[26] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[27] K. Shum and V.-K. Wei. A strong proxy signature scheme with proxy signer privacy protection. In *Eleventh IEEE International Workshop on Enabling Technologies: Infrastucture for Collaborative Enterprises (WET ICE '02)*, 2002.

[28] H. Sun, N.-Y. Lee, and T. Hwang. Threshold proxy signatures. In *IEEE Proceedings - Computers and Digital Techniques*, volume 146, pages 259–263. IEEE Press, 1999.

[29] H. M. Sun. An efficient nonrepudiable threshold proxy signature scheme with known signers. *Computer Communications*, 22(8):717–722, 1999.

[30] H.-M. Sun. On the design of time-stamped proxy signatures with traceable receivers. In *IEEE Proceedings - Computers and Digital Techniques*. IEEE Press, 2000.

[31] H.-M. Sun and B.-T. Hsieh. Remarks on two nonrepudiable proxy signature schemes. In *Ninth National Conference on Information Security*, volume 241-246, 1999.

[32] H.-M. Sun and B.-T. Hsieh. On the security of some proxy signature schemes. *Cryptology ePrint Archive, Report 2003/068. Available at* `http://eprint.iacr.org/`, 2003.

[33] V. Varadharajan, P. Allen, and S. Black. An analysis of the proxy problem in distributed systems. In *Proceedings of 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 255–275, 1991.

[34] C.-K. Wu and V. Varadharajan. Modified Chinese Remainder Theorem and its application to proxy signatures. In *ICPP Workshop*, pages 146–, 1999.

[35] S.-M. Yen, C.-P. Hung, and Y.-Y. Lee. Remarks on some proxy signature schemes. In *Workshop on Cryptology and Information Security, 2000 ICS*, pages 54–59, 2000.

[36] K. Zhang. Nonrepudiable proxy signature schemes. *Manuscript, Available at* `http://citeseer.nj.nec.com/360090.html`, 1997.

[37] K. Zhang. Threshold proxy signature schemes. In *Proceedings of 1st International Information Security Workshop*, pages 282–290, 1997.

A Proof of Theorem 5.2

The proof is by reduction; we show that for every adversary $\mathcal{A}$ with non-negligible advantage $\mathbf{Adv}^{\text{ps-uf}}_{\mathsf{PS}[\mathsf{AS}],\mathcal{A}}$ we can construct an adversary $\mathcal{B}$ with non-negligible advantage $\mathbf{Adv}^{\text{ag-uf}}_{\mathsf{AS},\mathcal{B}}$, so if AS is a secure aggregate signature scheme, $\mathsf{PS}[\mathsf{AS}]$ is a secure proxy signature scheme. The details are as follows.

Description of the adversary. $\mathcal{B}$ is given pk_1 and access to the signing oracle $\mathcal{O}_{\mathcal{S}}(sk_1, \cdot)$. $\mathcal{B}$ initializes a counter $n = 1$ for the number of users and creates an empty array $\mathbf{wskp}_1$. $\mathcal{B}$ runs $\mathcal{A}$ on input pk_1 handling all of $\mathcal{A}$'s requests and answering all $\mathcal{A}$'s queries as follows:

- If $\mathcal{A}$ requests to register a new user user $i = n + 1$ by outputting (pk_i, sk_i), then $\mathcal{B}$ stores these keys, increments n and creates an empty array $\mathbf{wskp}_i$.

- If $\mathcal{A}$ requests to interact with $\mathcal{D}(pk_1, pk_i, sk_1)$, where $i \in \{2, \ldots, n\}$, playing the role of $\mathcal{P}(pk_1, pk_i, sk_i)$, $\mathcal{B}$ creates an appropriate warrant[3] ω and makes query $00||pk_i||\omega$ to its signing oracle $\mathcal{O}_{\mathcal{S}}(sk_1, \cdot)$. Upon receiving an answer cert, it forwards ω, cert to $\mathcal{A}$.
- If $\mathcal{A}$ requests to interact with $\mathcal{P}(pk_i, pk_1, sk_1)$, where $i \in \{2, \ldots, n\}$, playing the role of $\mathcal{D}(pk_i, pk_1, sk_i)$, when $\mathcal{A}$ outputs ω, cert, B verifies that cert is a valid signature for message $00||pk_1||\omega$ (i.e., it checks if $\mathcal{V}(pk_i, 00||pk_1||\omega, \mathsf{cert}) = 1$). If so, the adversary stores ω, cert in the last unoccupied position of $\mathbf{wskp}_i$.
- If $\mathcal{A}$ requests that user 1 run the designation protocol with itself, B creates an appropriate warrant ω and makes query $00||pk_1||\omega$ to its signing oracle $\mathcal{O}_{\mathcal{S}}(sk_1, \cdot)$. Upon receiving an answer cert, it stores ω, cert in the last unoccupied position of $\mathbf{wskp}_1$.
- If $\mathcal{A}$ queries its oracle $\mathcal{O}_{\mathcal{S}_1}(sk_1, \cdot)$ with a message M, B makes query $11||M$ to its own signing oracle $\mathcal{O}_{\mathcal{S}}(sk_1, \cdot)$ and forwards the response to $\mathcal{A}$.
- If $\mathcal{A}$ makes a query (i, l, M), where $i \in [n]$, $l \in \mathbb{N}$, and $M \in \{0,1\}^*$, to its oracle $\mathcal{O}_{\mathcal{PS}}((\mathbf{skp}_u)_{u\in[n]}, \cdot, \cdot, \cdot)$, B responds as follows. If $\mathbf{wskp}_i[l]$ is not defined, it returns $\perp$ to $\mathcal{A}$. Otherwise, let ω, cert be the content of this position. $\mathcal{B}$ submits to the signing oracle $(01||pk_i||M)$ and in return obtains σ. It then computes $a\sigma$ as: $a\sigma = \mathcal{A}(pk_i, pk_1, 00||pk_1||\omega, 01||pk_i||M, \mathsf{cert}, \sigma)$ and returns $\omega, pk_1, a\sigma$ to $\mathcal{A}$.

Eventually $\mathcal{A}$ outputs a forgery. If $\mathcal{A}$ outputs a forgery of the form (M, σ) then the forgery output by $\mathcal{B}$ is $(11||M, \sigma)$. If $\mathcal{A}$ outputs $(M, p\sigma, pk_i)$ with $\mathcal{ID}(p\sigma) = pk_1$ then adversary $\mathcal{B}$ outputs $(pk_i, 01||pk_i||M, 00||pk_1||\omega, p\sigma)$. The last case is when $\mathcal{A}$ outputs a forgery $(M, p\sigma, pk_1)$ such that $\mathcal{ID}(p\sigma) = pk \notin D \cup \{pk_1\} \cup \{\perp\}$; in this case $\mathcal{B}$ outputs $(pk, 00||pk||\omega, 01||pk_1||M, p\sigma)$.

Analysis. We first claim that the view of A in the simulated experiment above is indistinguishable from the one in the real experiment $\mathbf{Exp}^{\text{ps-uf}}_{\mathsf{PS[AS]},\mathcal{A}}(k)$. Since the simulation is perfect and $\mathcal{A}$ has non-negligible advantage, at least one of the following events occurs with non-negligible probability. (See Definition 3.2.)

E_1 : $\mathcal{A}$ outputs a forgery of the form (M, σ), where $\mathcal{V}_1(pk_1, M, \sigma) = 1$, and M was not queried to oracle $\mathcal{O}_{\mathcal{S}_1}(sk_1, \cdot)$.

E_2 : $\mathcal{A}$ outputs a forgery of the form $(M, p\sigma, pk_i)$, for some $i \in \mathbb{N}$ where $\mathcal{PV}(pk_i, M, p\sigma) = 1$, $\mathcal{ID}(p\sigma) = pk_1$, and no query (i, j, M) was made to oracle $\mathcal{O}_{\mathcal{PS}}((\mathbf{skp}_u)_{u\in[n]}, \cdot, \cdot, \cdot)$, for $j \in \mathbb{N}$.

E_3 : $\mathcal{A}$ outputs a forgery of the form $(M, p\sigma, pk_1)$, where $\mathcal{PV}(pk_1, M, p\sigma) = 1$, and $\mathcal{ID}(p\sigma) = pk \notin D \cup \{pk_1\} \cup \{\perp\}$.

If event E_1 occurs, then M, σ are such that $\mathcal{V}_1(pk, 11||M, \sigma) = 1$. Since $\mathcal{A}$ did not query M to its signing oracle, $\mathcal{B}$ did not query $11||M$ to its signing oracle. Therefore, $\mathcal{B}$'s output $(11||M, \sigma)$ is a valid forgery.

If event E_2 occurs, $p\sigma$ is a valid proxy signature of user 1 on behalf of user i, hence, $p\sigma$ must be a valid aggregate signature for messages $00||pk_1||\omega$ and $01||pk_i||M$, relative to public keys pk_i and pk_1 respectively. It's a valid forgery for $\mathcal{A}$, i.e. $\mathcal{A}$ did not make a query i, l, M for any $l \in \mathbb{N}$ to its oracle $\mathcal{O}_{\mathcal{PS}}((\mathbf{skp}_u)_{u\in[n]}, \cdot, \cdot, \cdot)$. Hence $\mathcal{B}$ did not query message $01||pk_i||M$ to its signing oracle. Therefore, $(pk_i, 01||pk_i||M, 00||pk_1||\omega, p\sigma)$ is a valid forgery for $\mathcal{B}$.

Finally, if event E_3 occurs, $p\sigma$ is a valid aggregate signature on messages $00||pk||\omega$ and $01||pk_1||M$ relative to pk_1 and pk respectively. So the forgery output by $\mathcal{B}$ in this case, i.e. $(pk, 00||pk||\omega, 01||pk_1||M, p\sigma)$ is valid, as long as this message was not queried by $\mathcal{B}$ to its signing oracle. It is

[3]As we mentioned before, we do not specify the information contained in the warrant, since it is public and it depends on the application.

immediate that $\mathcal{B}$ makes this query only in the case $\mathcal{A}$ requests that user 1 designates user $\mathcal{ID}(p\sigma)$ as a proxy signer, which is not the case (since $\mathcal{ID}(p\sigma) = pk \notin D \cup \{pk_1\} \cup \{\perp\}$).

We showed that whenever $\mathcal{A}$ outputs a valid forgery, $\mathcal{B}$ outputs a valid forgery, therefore

$$\mathbf{Adv}^{\text{ps-uf}}_{\mathsf{PS[AS]},\mathcal{A}} = \mathbf{Adv}^{\text{ag-uf}}_{\mathsf{AS},\mathcal{B}}.$$

Since $\mathcal{A}$ runs poly(k)-time, it is easy to see that $\mathcal{B}$ runs poly(k)-time.

B Proof of Theorem 6.1

Since the Schnorr signature scheme $\mathsf{S} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ is secure in the random-oracle model, assuming hardness of the DLP [25], it is sufficient to prove that if S is secure then TS is secure. To this end, for any polynomial-time adversary $\mathcal{A}$ against TS we construct polynomial-time adversaries $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ against S such that if $\mathcal{A}$ has a non-negligible advantage in creating a forgery for TS, then at least one of the three adversaries has a non-negligible advantage in creating a forgery for S.

Fix $k \in \mathbb{N}$ and let $\mathcal{A}$ be a poly(k)-time adversary against TS. As is usual in the random-oracle model, the hash functions G and H used in the scheme are assumed to behave as random oracles, i.e., they are assumed to be chosen independently at random from all functions $f : \{0,1\}^* \to \mathbb{Z}_q$, and the adversary is given access to these oracles. Without loss of generality, we assume that the adversary does not repeat any random-oracle queries (it can just store the responses in a table).

Description of the adversaries. Since adversaries $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ have many similarities, we describe them simultaneously, pointing out their differences. The adversaries are each given input a public key pk_1, and access to a random oracle G and a Schnorr signing oracle $\mathcal{O}_{\mathcal{S}^G}(sk_1, \cdot)$, where sk_1 is a secret key matching pk_1. The adversaries use their oracles G and $\mathcal{O}_{\mathcal{S}^G}(sk_1, \cdot)$ to answer $\mathcal{A}$'s requests and queries as described in detail below. They simulate random oracle H. Some queries to H are answered with a freshly chosen random value, but for certain inputs which we specify shortly, the value of random oracle H is previously set to a uniformly distributed value.

Each adversary sets $n = 1$, creates empty arrays $\mathbf{wskp}_1$ and H, chooses some randomness for $\mathcal{A}$, and runs $\mathcal{A}$ on input pk_1 with this randomness. It then answers the requests and queries made by $\mathcal{A}$ as follows.

- If $\mathcal{A}$ requests to register a public key pk_i for user $i = n + 1$ by outputting a pair (pk_i, sk_i) of matching public and secret keys, the adversary stores these keys, increments n, and creates an empty array $\mathbf{wskp}_i$.
- If $\mathcal{A}$ requests to interact with user 1 running $\mathcal{D}(pk_1, pk_i, sk_1)$, for some $i \in \{2, \ldots, n\}$, and play the role of user i running $\mathcal{P}(pk_1, pk_i, sk_i)$, the adversary creates an appropriate warrant ω and makes query $0||pk_1||pk_i||\omega$ to its signing oracle $\mathcal{O}_{\mathcal{S}^G}(sk_1, \cdot)$. Upon receiving an answer cert, it forwards ω, cert to $\mathcal{A}$.
- If $\mathcal{A}$ requests to interact with user 1 running $\mathcal{P}(pk_i, pk_1, sk_1)$, for some $i \in \{2, \ldots, n\}$, and play the role of user i running $\mathcal{D}(pk_i, pk_1, sk_i)$, when $\mathcal{A}$ outputs ω, cert, the adversary verifies that $\mathsf{cert} = (Y, s)$ is a valid signature for message $0||pk_i||pk_1||\omega$ (i.e., it checks that $\mathcal{V}^G(pk_i, 0||pk_i||pk_1||\omega, \mathsf{cert}) = 1$). If so, the adversary stores ω, Y, s in the last unoccupied position of $\mathbf{wskp}_i$.
- If $\mathcal{A}$ requests that user 1 run the designation protocol with itself, the adversary creates an appropriate warrant ω and makes query $0||pk_1||pk_1||\omega$ to its signing oracle $\mathcal{O}_{\mathcal{S}^G}(sk_1, \cdot)$. Upon receiving an answer $\mathsf{cert} = (Y, s)$, adversaries $\mathcal{B}$ and $\mathcal{D}$ store ω, Y, s in the last unoccupied position of $\mathbf{wskp}_1$ and forward ω, cert to $\mathcal{A}$. Adversary $\mathcal{C}$ checks if $\mathcal{A}$ has made the query $(0||pk_1||pk_1||\omega, Y)$ to random oracle G. If so, it aborts. Otherwise, it stores ω, Y, s in the last unoccupied position of $\mathbf{wskp}_1$ and forwards ω, cert to $\mathcal{A}$. Note that since oracle $\mathcal{O}_{\mathcal{S}^G}(sk_1, \cdot)$

chooses commitment Y uniformly at random, the probability that $\mathcal{A}$ previously made query $(0||pk_1||pk_1||\omega, Y)$ to random oracle G is negligible.

- If $\mathcal{A}$ queries its oracle $\mathcal{O}_{\mathcal{S}_\mathsf{T}}(sk_1, \cdot)$ with a message M, the adversary makes query $1||M$ to its signing oracle $\mathcal{O}_{\mathcal{S}^G}(sk_1, \cdot)$ and forwards the response to $\mathcal{A}$.
- If $\mathcal{A}$ makes a query (i, l, M), where $i \in [n]$, $l \in \mathbb{N}$, and $M \in \{0,1\}^*$, to its oracle $\mathcal{O}_{\mathcal{PS}}((\mathbf{skp}_u)_{u\in[n]}, \cdot, \cdot, \cdot)$, the adversary responds as follows. If $\mathbf{wskp}_i[l]$ is not defined, it returns $\perp$ to $\mathcal{A}$. Otherwise, it parses $\mathbf{wskp}_i[l]$ as ω_l, Y_l, s_l, and performs the following operations.
  - Pick a random $c \in \mathbb{Z}_q$
  - Pick a random $s \in \mathbb{Z}_q$
  - Make query $(0||pk_i||pk_1||\omega_l, Y_l)$ to oracle G and let e be the response
  - Compute commitment $Y \leftarrow g^s \cdot ((pk_i \cdot pk_1)^e \cdot Y_l)^{-c} \bmod p$
  - If $\mathcal{A}$ has made the query $(0||M||pk_i||pk_1||\omega_l||Y_l, Y)$ to random oracle H, then abort. Otherwise, set $H[0||M||pk_i||pk_1||\omega_l||Y_l, Y] \leftarrow c$
  - Return $(\omega_l, Y_l, pk_1, (Y, s))$ to $\mathcal{A}$

  Thus, the adversary simulates proxy signing by user 1 on behalf of user i using the l-th proxy signing key. It is well known and easy to see that the simulated signature (Y, s) has the same distribution as a real Schnorr signature. Therefore, the signature returned to adversary $\mathcal{A}$ has the same distribution as a signature returned by oracle $\mathcal{O}_{\mathcal{PS}}((\mathbf{skp}_u)_{u\in[n]}, \cdot, \cdot, \cdot)$. Note that since s is uniformly distributed and independent from $\mathcal{A}$, Y is uniformly distributed and independent from $\mathcal{A}$. Hence the probability that $\mathcal{A}$ made query $(0||M||pk_i||pk_1||\omega_l||Y_l, Y)$ to random oracle H is negligible.
- If $\mathcal{A}$ makes a query x to random oracle G, the adversary makes the same query to its oracle G, obtaining a response e. Adversaries $\mathcal{B}$ and $\mathcal{D}$ return e to $\mathcal{A}$. Adversary $\mathcal{C}$ checks if x is of the form $(0||pk_1||pk_1||\omega, Y)$, where ω, Y does not appear in $\mathbf{wskp}_1$ (i.e., for all l, if $\mathbf{wskp}_1[l] = \omega_l, Y_l, s_l$, then $\omega \neq \omega_l$ or $Y \neq Y_l$). If so, it returns $e \cdot 2^{-1} \bmod q$ to $\mathcal{A}$. Otherwise, it returns e to $\mathcal{A}$. In either case, the response is uniformly distributed.
- If $\mathcal{A}$ makes a query (M, Z) to random oracle H, the adversary checks if $H[M, Z]$ is defined. If not, it picks a random $c \in \mathbb{Z}_q$ and sets $H[M, Z] \leftarrow c$. Then it returns $H[M, Z]$ to $\mathcal{A}$.

The behaviors of adversaries $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ differ once $\mathcal{A}$ outputs its forgery (M, σ) or $(M, p\sigma, pk)$.

Adversary $\mathcal{B}$ aborts if $\mathcal{A}$'s forgery is not of the form (M, σ). Otherwise, it outputs the forgery $(1||M, \sigma)$.

Adversary $\mathcal{C}$ aborts if $\mathcal{A}$'s forgery is not of the form $(M, p\sigma, pk_i)$, for some $i \in [n]$, where $\mathcal{ID}(p\sigma) = pk_1$. Otherwise, $\mathcal{C}$ parses $p\sigma$ as $(\omega, Y, pk_1, (\overline{Y}, \bar{s}))$. If $\mathcal{A}$ did not make query $(0||M||pk_i||pk_1||\omega||Y, \overline{Y})$ to random oracle H, then $\mathcal{C}$ aborts. Otherwise, let c be the response adversary $\mathcal{C}$ gave $\mathcal{A}$ when it made this query. $\mathcal{C}$ rewinds $\mathcal{A}$ to the point where it makes this query and gives it a randomly chosen response $c' \neq c$. It continues the execution of $\mathcal{A}$ (with the same randomness), responding to its requests and queries as before (with fresh coins), until $\mathcal{A}$ outputs a forgery. If this forgery is not of the form $(M, (\omega, Y, pk_1, (\overline{Y}, \bar{s}')), pk_i)$, then $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ computes the proxy secret corresponding to $\mathcal{A}$'s forgeries as $t \leftarrow (\bar{s} - \bar{s}') \cdot (c - c')^{-1} \bmod q$.

If $pk_i \neq pk_1$, then $\mathcal{C}$ makes the query $(0||pk_i||pk_1||\omega, Y)$ to random oracle G, obtains a response e, computes $s' \leftarrow t - e \cdot sk_i \bmod q$, and outputs the forgery $(0||pk_i||pk_1||\omega, (Y, s'))$.

If $pk_i = pk_1$ and there exists $l \in \mathbb{N}$ such that $\mathbf{wskp}_1[l] = \omega, Y, s_l$ for some s_l, then $\mathcal{C}$ makes the query $(0||pk_1||pk_1||\omega, Y)$ to random oracle G, obtains a response e, and computes user 1's secret

key as $sk_1 \leftarrow (t - s_l) \cdot e^{-1} \bmod q$. It then selects a message M' that was not queried to oracle $\mathcal{O}_{\mathcal{S}^G}(sk_1, \cdot)$, computes a Schnorr signature (Y', s') for this message under key sk_1 and outputs the forgery $(M', (Y', s'))$.

If $pk_i = pk_1$ and no such l exists (i.e., ω, Y does not appear in $\mathbf{wskp}_1$), then $\mathcal{C}$ outputs the forgery $(0||pk_1||pk_1||\omega, (Y, t))$.

Adversary $\mathcal{D}$ aborts if $\mathcal{A}$'s forgery is not of the form $(M, p\sigma, pk_1)$. Otherwise, $\mathcal{D}$ parses $p\sigma$ as $(\omega, Y, pk_i, (\overline{Y}, \bar{s}))$. If $pk_i = pk_1$ or $\mathcal{A}$ did not make query $(0||M||pk_1||pk_i||\omega||Y, \overline{Y})$ to random oracle H, then $\mathcal{D}$ aborts. Otherwise, let c be the response adversary $\mathcal{D}$ gave $\mathcal{A}$ when it made this query. $\mathcal{D}$ rewinds $\mathcal{A}$ to the point where it makes this query and gives it a randomly chosen response $c' \neq c$. It continues the execution of $\mathcal{A}$ (with the same randomness), responding to its requests and queries as before (with fresh coins), until $\mathcal{A}$ outputs a forgery. If this forgery is not of the form $(M, (\omega, Y, pk_i, (\overline{Y}, \bar{s}')), pk_1)$, then $\mathcal{D}$ aborts. Otherwise, $\mathcal{D}$ computes the proxy secret corresponding to $\mathcal{A}$'s forgeries as $t \leftarrow (\bar{s} - \bar{s}') \cdot (c - c')^{-1} \bmod q$. $\mathcal{D}$ makes the query $(0||pk_1||pk_i||\omega, Y)$ to random oracle G, obtains a response e, computes $s' \leftarrow t - e \cdot sk_i \bmod q$, and outputs the forgery $(0||pk_1||pk_i||\omega, (Y, s'))$.

Clearly, the running time of adversaries $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ is polynomial in k.

Analysis. Consider experiment $\mathbf{Exp}^{\text{ps-uf}}_{\mathsf{TS},\mathcal{A}}(k)$. Since $\mathcal{A}$ has a non-negligible advantage, at least one of the following events occurs with non-negligible probability. (See Definition 3.2.)

- E_1 : $\mathcal{A}$ outputs a forgery of the form (M, σ), where $\mathcal{V}_{\mathsf{T}}(pk_1, M, \sigma) = 1$, and M was not queried to oracle $\mathcal{O}_{\mathcal{S}_{\mathsf{T}}}(sk_1, \cdot)$
- E_2 : $\mathcal{A}$ outputs a forgery of the form $(M, p\sigma, pk_i)$, for some $i \in [n]$, where $\mathcal{PV}(pk_i, M, p\sigma) = 1$, $\mathcal{ID}(p\sigma) = pk_1$, and no valid query (i, l, M), for $l \in \mathbb{N}$, was made to oracle $\mathcal{O}_{\mathcal{PS}}((\mathbf{skp}_u)_{u\in[n]}, \cdot, \cdot, \cdot)$
- E_3 : $\mathcal{A}$'s forgery is of the form $(M, p\sigma, pk_1)$, where $\mathcal{PV}(pk_1, M, p\sigma) = 1$ and $\mathcal{ID}(p\sigma) \notin D \cup \{pk_1\} \cup \{\perp\}$

We observe that if adversaries $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ do not abort before $\mathcal{A}$ outputs its forgery, then they perfectly simulate the environment provided to $\mathcal{A}$ in experiment $\mathbf{Exp}^{\text{ps-uf}}_{\mathsf{TS},\mathcal{A}}(k)$. Also, the probability that $\mathcal{B}$, $\mathcal{C}$, or $\mathcal{D}$ abort before $\mathcal{A}$ outputs its forgery is negligible. We will show that if event E_1 occurs, then with non-negligible probability adversary $\mathcal{B}$ is successful, if E_2 occurs, then with non-negligible probability $\mathcal{C}$ is successful, and if E_3 occurs, then with non-negligible probability $\mathcal{D}$ is successful. This implies that the advantage of at least one of these adversaries is non-negligible, as desired.

Assume event E_1 occurs and consider the execution of $\mathcal{A}$ by adversary $\mathcal{B}$. Assume $\mathcal{B}$ does not abort. Then $\mathcal{B}$ outputs the forgery $(1||M, \sigma)$. Since $\mathcal{V}_{\mathsf{T}}(pk_1, M, \sigma) = 1$, $\mathcal{V}^G(pk_1, 1||M, \sigma) = 1$. Since M was not queried to oracle $\mathcal{O}_{\mathcal{S}_{\mathsf{T}}}(sk_1, \cdot)$, $\mathcal{B}$ did not make query $1||M$ to oracle $\mathcal{O}_{\mathcal{S}^G}(sk_1, \cdot)$. Thus $\mathcal{B}$'s output is a successful forgery of a Schnorr signature.

Assume event E_2 occurs. Consider the execution of $\mathcal{A}$ by adversary $\mathcal{C}$ and assume $\mathcal{C}$ does not abort. Notice that $\mathcal{C}$ simulates random oracle G with a function G' that is equal to G except possibly on some points of the form $(0||pk_1||pk_1||\omega, Y)$, where $G'(0||pk_1||pk_1||\omega, Y) = G(0||pk_1||pk_1||\omega, Y) \cdot 2^{-1} \bmod q$. Since E_2 occurs, $\mathcal{PV}(pk_i, M, p\sigma) = 1$ and $\mathcal{ID}(p\sigma) = pk_1$. Therefore, $p\sigma$ must be of the form $(\omega, Y, pk_1, (\overline{Y}, \bar{s}))$, where $(\overline{Y}, \bar{s})$ is a valid Schnorr signature with random oracle H for message $0||M||pk_i||pk_1||\omega||Y$ relative to proxy public key $pkp = (pk_i \cdot pk_1)^{G'(0||pk_i||pk_1||\omega, Y)} \cdot Y \bmod p$. With non-negligible probability, $\mathcal{A}$ made query $(0||M||pk_i||pk_1||\omega||Y, \overline{Y})$ to random oracle H (otherwise, it would have only a negligible probability of outputting such a valid Schnorr signature $(\overline{Y}, \bar{s})$.) When $\mathcal{A}$ made this query, $H[0||M||pk_i||pk_1||\omega||Y, \overline{Y}]$ must not have been defined because $\mathcal{A}$ did not make a valid query (i, l, M), for $l \in \mathbb{N}$, to oracle $\mathcal{O}_{\mathcal{PS}}((\mathbf{skp}_u)_{u\in[n]}, \cdot, \cdot, \cdot)$. (Only such a query could

cause $H[0||M||pk_i||pk_1||\omega||Y,\overline{Y}]$ to be defined.) Hence $\mathcal{C}$ must have chosen a fresh random value $c \in \mathbb{Z}_q$ as a response to this query. When $\mathcal{C}$ rewinds $\mathcal{A}$ to this point and gives it a random response $c' \neq c$, with non-negligible probability, $\mathcal{A}$ produces a forgery with respect to the same query. (The exact probabilistic analysis is based on the forking lemma of [25] and is similar to the analysis presented there for the proof of security of the Schnorr signature scheme.) Since the query contains M, pk_i, pk_1, ω, Y, and $\overline{Y}$, with non-negligible probability, $\mathcal{A}$ forges a proxy signature for the same message M, relative to the same public key pkp, using the same commitment $\overline{Y}$, i.e., it produces a forgery of the form $(M,(\omega,Y,pk_1,(\overline{Y},\bar{s}')),pk_i)$. From the two Schnorr signatures $(\overline{Y},\bar{s})$ and $(\overline{Y},\bar{s}')$, $\mathcal{C}$ extracts the discrete logarithm t of the proxy public key pkp. To see that the value computed is indeed the discrete logarithm of pkp, observe that since these signatures are valid relative to pkp, $g^{\bar{s}} \equiv \overline{Y} \cdot pkp^{c} \pmod{p}$ and $g^{\bar{s}'} \equiv \overline{Y} \cdot pkp^{c'} \pmod{p}$. Hence $g^{(\bar{s}-\bar{s}') \bmod q} \equiv pkp^{(c-c') \bmod q} \pmod{p}$, and therefore, $g^{(\bar{s}-\bar{s}')\cdot(c-c')^{-1} \bmod q} \equiv pkp \pmod{p}$. Thus $t = (\bar{s}-\bar{s}')\cdot(c-c')^{-1} \bmod q$ is the discrete logarithm of pkp. Since $pkp = (pk_i \cdot pk_1)^{G'(0||pk_i||pk_1||\omega,Y)} \cdot Y \bmod p$, $t = G'(0||pk_i||pk_1||\omega,Y)\cdot sk_i + G'(0||pk_i||pk_1||\omega,Y)\cdot sk_1 + y \bmod q$, where sk_i is the discrete logarithm of pk_i and y is the discrete logarithm of Y.

If $pk_i \neq pk_1$, then adversary $\mathcal{C}$ stored the key pair (pk_i, sk_i) when $\mathcal{A}$ made the request to register public key pk_i for user i. Also, note that $G'(0||pk_i||pk_1||\omega,Y) = G(0||pk_i||pk_1||\omega,Y)$. Therefore, when $\mathcal{C}$ subtracts the term $G(0||pk_i||pk_1||\omega,Y)\cdot sk_i \bmod q$ from t, it obtains $s' = G(0||pk_i||pk_1||\omega,Y)\cdot sk_1 + y \bmod q$. We observe that (Y,s') is a valid Schnorr signature with random oracle G for message $0||pk_i||pk_1||\omega$ relative to public key pk_1. Since $\mathcal{C}$ did not make the query $0||pk_i||pk_1||\omega$ to oracle $\mathcal{O}_{\mathcal{S}^G}(sk_1,\cdot)$, its forgery $(0||pk_i||pk_1||\omega,(Y,s'))$ is successful.

If $pk_i = pk_1$ and there exists $l \in \mathbb{N}$ such that $\mathbf{wskp}_1[l] = \omega, Y, s_l$ for some s_l, then $G'(0||pk_1||pk_1||\omega,Y) = G(0||pk_1||pk_1||\omega,Y)$. Also, since (Y,s_l) is a Schnorr signature with random oracle G for message $0||pk_1||pk_1||\omega$ under key sk_1, $s_l = G(0||pk_1||pk_1||\omega,Y)\cdot sk_1 + y \bmod q$. Therefore, when $\mathcal{C}$ subtracts the term s_l from t and divides by $G(0||pk_1||pk_1||\omega,Y)$, it obtains sk_1. Having user 1's secret key, it easily constructs a successful forgery.

If $pk_i = pk_1$ and no such l exists (i.e., ω, Y does not appear in $\mathbf{wskp}_1$), then with non-negligible probability, $\mathcal{A}$ made query $(0||pk_1||pk_1||\omega,Y)$ to random oracle G (otherwise, it would have only a negligible probability of outputting a Schnorr signature valid relative to pkp.) $\mathcal{C}$ gave $\mathcal{A}$ the value $G(0||pk_1||pk_1||\omega,Y)\cdot 2^{-1} \bmod q$ as a response to this query, i.e., $G'(0||pk_1||pk_1||\omega,Y) = G(0||pk_1||pk_1||\omega,Y)\cdot 2^{-1} \bmod q$. Therefore, $t = G(0||pk_1||pk_1||\omega,Y)\cdot 2^{-1}\cdot sk_1 + G(0||pk_1||pk_1||\omega,Y)\cdot 2^{-1}\cdot sk_1 + y \bmod q = G(0||pk_1||pk_1||\omega,Y)\cdot sk_1 + y \bmod q$. We observe that (Y,t) is a valid Schnorr signature with random oracle G for message $0||pk_1||pk_1||\omega$ relative to public key pk_1. Since ω, Y does not appear in $\mathbf{wskp}_1$ and $\mathcal{C}$ did not abort, it must be the case that $\mathcal{C}$ did not make the query $0||pk_1||pk_1||\omega$ to oracle $\mathcal{O}_{\mathcal{S}^G}(sk_1,\cdot)$. Thus its forgery $(0||pk_1||pk_1||\omega,(Y,t))$ is successful.

Assume event E_3 occurs. Consider the execution of $\mathcal{A}$ by adversary $\mathcal{D}$ and assume $\mathcal{D}$ does not abort. Since $\mathcal{PV}(pk_1,M,p\sigma) = 1$, $p\sigma$ must be of the form $(\omega,Y,pk_i,(\overline{Y},\bar{s}))$, where $(\overline{Y},\bar{s})$ is a valid Schnorr signature with random oracle H for message $0||M||pk_1||pk_i||\omega||Y$ relative to proxy public key $pkp = (pk_1 \cdot pk_i)^{G(0||pk_1||pk_i||\omega,Y)} \cdot Y \bmod p$. As in the previous case, $\mathcal{A}$ must have made query $(0||M||pk_1||pk_i||\omega||Y,\overline{Y})$ to random oracle H (otherwise, it would have only a negligible probability of outputting such a valid Schnorr signature $(\overline{Y},\bar{s})$.) When $\mathcal{A}$ made this query, $H[0||M||pk_1||pk_i||\omega||Y,\overline{Y}]$ must not have been defined because values for H are only set by $\mathcal{D}$ when answering queries to H or when simulating oracle $\mathcal{O}_{\mathcal{PS}}((\mathbf{skp}_u)_{u\in[n]},\cdot,\cdot,\cdot)$, and the points for which H is set during such a simulation are of the form $(0||M||pk_i||pk_1||\omega_j||Y_j,Y)$.) Hence $\mathcal{D}$ must have chosen a fresh random value $c \in \mathbb{Z}_q$ as a response to this query. When $\mathcal{D}$ rewinds $\mathcal{A}$ to this point and gives it a random response $c' \neq c$, with non-negligible probability, $\mathcal{A}$ produces a forgery with respect to the same query. Since the query contains M, pk_1, pk_i, ω, Y, and $\overline{Y}$, with non-negligible

probability, $\mathcal{A}$ forges a signature for the same message M, relative to the same public key pkp, using the same commitment $\overline{Y}$, i.e., it produces a forgery of the form $(M, (\omega, Y, pk_i, (\overline{Y}, \bar{s}')), pk_1)$. From the two Schnorr signatures $(\overline{Y}, \bar{s})$ and $(\overline{Y}, \bar{s}')$, $\mathcal{D}$ extracts the discrete logarithm t of the proxy public key pkp. As in the previous case, it is easy to see that the value computed is indeed the discrete logarithm of pkp, and that $t = G(0||pk_1||pk_i||\omega, Y) \cdot sk_1 + G(0||pk_1||pk_i||\omega, Y) \cdot sk_i + y \bmod q$, where sk_i is the discrete logarithm of pk_i and y is the discrete logarithm of Y. Adversary $\mathcal{D}$ knows the secret key sk_i. When it subtracts the term $G(0||pk_1||pk_i||\omega, Y) \cdot sk_i \bmod q$ from t, it obtains $s' = G(0||pk_1||pk_i||\omega, Y) \cdot sk_1 + y \bmod q$. We observe that (Y, s') is a valid Schnorr signature with random oracle G for message $0||pk_1||pk_i||\omega$ relative to public key pk_1. Since $\mathcal{ID}(p\sigma) \notin D$, $\mathcal{D}$ did not make the query $0||pk_1||pk_i||\omega$ to oracle $\mathcal{O}_{\mathcal{S}^G}(sk_1, \cdot)$. Therefore, its forgery $(0||pk_1||pk_i||\omega, (Y, s'))$ is successful.