Formal to Practical Security – LNCS 5458, pages 95–115. V. Cortier, C. Kirchner, M. Okada, and H. Sakurada Eds. Springer-Verlag.

Anonymous Consecutive Delegation of Signing Rights: Unifying Group and Proxy Signatures*

Georg Fuchsbauer and David Pointcheval

École normale supérieure, LIENS - CNRS - INRIA, Paris, France
`http://www.di.ens.fr/{~fuchsbau,~pointche}`

Abstract. We define a general model for consecutive delegations of signing rights with the following properties: The delegatee actually signing and all intermediate delegators remain anonymous. As for group signatures, in case of misuse, a special authority can *open* signatures to reveal all delegators' and the signer's identity. The scheme satisfies a strong notion of non-frameability generalizing the one for dynamic group signatures. We give formal definitions of security and show them to be satisfiable by constructing an instantiation proven secure under general assumptions in the standard model. Our primitive is a proper generalization of both group signatures and proxy signatures and can be regarded as non-frameable dynamic hierarchical group signatures.

1 Introduction

The concept of delegating signing rights for digital signatures is a well studied subject in cryptography. The most basic concept is that of proxy signatures, introduced by Mambo et al. [MUO96] and group signatures, introduced by Chaum and van Heyst [CvH91]. In the first, a *delegator* transfers the right to sign on his behalf to a *proxy signer* in a *delegation protocol*. Now the latter can produce *proxy signatures* that are verifiable under the delegator's public key. Security of such a scheme amounts to unforgeability of proxy signatures, in that an adversary can neither create a signature without having been delegated, nor impersonate an honest proxy signer.

On the other hand, in a group signature scheme, an authority called the *issuer* enrolls *group members*, who can then sign on behalf of the group, which has one single *group signature verification key*. Enrollment can be viewed as delegating the signing rights of the group—represented by the issuer—to its members. A crucial requirement is anonymity, meaning that from a signature one cannot tell which one of the group members actually signed. In contrast to ring signatures [RST01], to preclude misuse, there is another authority holding an *opening key* by which anonymity of the signer can be revoked. Generally, one distinguishes *static* and *dynamic* groups, depending on whether the system and the group are set up once and for all or whether members can join dynamically. For the dynamic case, a strong security notion called *non-frameability* is conceivable: nobody—not even the issuer nor the opener—is able to produce a signature that opens to a member who did not sign. The two standard security requirements are *traceability* (every valid signature can be traced to its signer), which together with non-frameability implies unforgeability, and *anonymity*, that is, no one except the opener can distinguish signatures of different users.

It is of central interest in cryptography to provide formal definitions of primitives and rigorously define the notions of security they should achieve. Only then can one *prove* instantiations of the primitive to be secure. Security of group signatures was first formalized by Bellare et al. [BMW03] and then extended to dynamic groups in [BSZ05]. The model of proxy signatures and their security were formalized by Boldyreva et al. [BPW03].[1]

* A short version of this work appeared as [FP08].

[1] Their scheme has later been attacked by [TL04]. Note, however, that our definition of non-frameability prevents this attack, since an adversary querying $\mathsf{PSig}(\cdot, \mathit{warr}, \cdot)$ and then creating a signature for task' is considered successful (cf. Sect. 3.3).

1.1 Our Results

The contribution of this paper is to unify the two above-mentioned seemingly rather different concepts, by establishing a general model which encompasses both proxy and group signatures, and which is of independent interest itself. We give security notions that imply the formal ones for both primitives. Moreover, we consider consecutive delegations where *all* intermediate delegators remain anonymous. As for dynamic group signatures, we define an opening authority separated from the issuer and which in addition might even be different for each user. (For proxy signatures, a plausible setting would be to enable the users to open signatures on their behalf.) We call our primitive *anonymous proxy signatures*, a term that already appeared in the literature (see e.g. [SK02]), however without providing a rigorous definition nor security proofs. As it is natural for proxy signatures, we consider a dynamic setting, which allows us to define an extension of non-frameability that additionally protects against wrongful accusation of delegation.

The most trivial instantiation of proxy signatures is "delegation-by-certificate": The delegator signs a document called the *warrant* containing the public key of the proxy and passes it to the latter. A proxy signature then consists of a regular signature by the proxy on the message and the signed warrant. Together they can by verified using the delegator's verification key only. Although hardly adaptable to the anonymous case—after all, the warrant contains the proxy's public key—, a virtue of the scheme is the fact that the delegator can restrict the delegated rights to specific *tasks* by specifying them in the warrant. Since our model supports re-delegation, a user might wish to re-delegate only a reduced subset of tasks she has been delegated for. We represent tasks by natural numbers and allow delegations for arbitrary sets of them, whereas re-delegation can be done for any subsets.

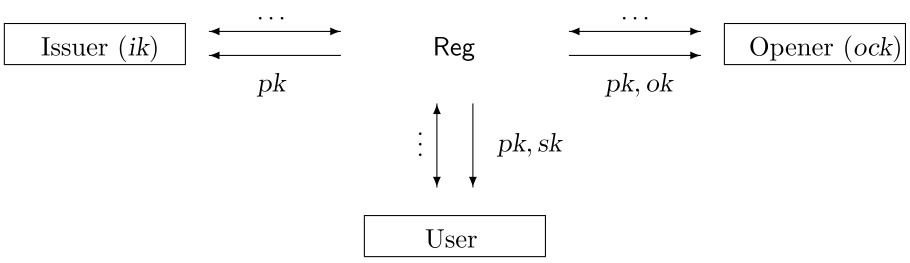
The primary practical motivation for the new primitive is GRID Computing, where Alice, after authenticating herself, starts a process. Once disconnected, the process may remain active, launch sub-processes and need access to additional resources that require further authentication. Alice thus delegates her rights to the process. On the one hand, not trusting the environment, she will not want to delegate all her rights, which can be realized by *delegation-by-certificate*. On the other hand, there is no need for the resources to know that it was not actually Alice who was authenticated, which is practically solely achieved by *full delegation*, i.e., giving the private key to the delegatee. While the first solution exposes the proxy's identity, the second approach does not allow for restriction of delegated rights nor provide any means to trace malicious signers. Anonymous proxy signatures incorporate both requirements at one blow.

Another feature of our primitive is that due to possible consecutiveness of delegations it can be regarded as *non-frameable, dynamic* hierarchical group signatures, a concept introduced by Trolin and Wikström [TW05] for the static setting.

After defining the new primitive and a corresponding security model, in order to show satisfiability of the definitions, we give an instantiation and prove it secure under the (standard) assumption that families of trapdoor permutations exist. The problem of devising a more efficient construction is left for future work. We emphasize furthermore that delegation in our scheme is non-interactive (the delegator simply sends a warrant she computed w.r.t. the delegatee's public key) and does not require a secure channel.

2 Algorithm Specification

We describe an anonymous proxy signature scheme by giving the algorithms it consists of. First of all, running algorithm Setup with the security parameter λ creates the public parameters of the scheme, as well as the *issuing key* ik given to the issuer in order to register users and the opener's certification key ock given to potential openers. When a user registers, she and her opening authority run the interactive protocol Reg with the issuer. In the end, all parties hold the user's public key pk, the user



$$
\begin{array}{rccccl}
\lambda & \rightarrow & \mathsf{Setup} & \rightarrow & pp, ik, ock \\
sk_x, [warr_{\rightarrow x},]\ TList, pk_y & \rightarrow & \mathsf{Del} & \rightarrow & warr_{[\rightarrow]x\rightarrow y} \\
sk_y, warr_{x\rightarrow\ldots\rightarrow y}, task, M & \rightarrow & \mathsf{PSig} & \rightarrow & \sigma \\
pk_x, task, M, \sigma & \rightarrow & \mathsf{PVer} & \rightarrow & b \in \{0,1\} \\
ok_x, \sigma, task, M \text{ and } \textit{registry}\text{-data} & \rightarrow & \mathsf{Open} & \rightarrow & \text{a list of users or } \perp \text{ (failure)}
\end{array}
$$

Fig. 1. Inputs and outputs of the algorithms

is the only one to know the corresponding signing key sk, and the opener possesses ok, the key to open signatures on the user's behalf.

Once a user U_1 is registered and holds her secret key sk_1, she can delegate her signing rights for a set of tasks $TList$ to user U_2 holding pk_2: U_1 runs $\mathsf{Del}(sk_1, TList, pk_2)$ to produce a warrant $warr_{1\rightarrow 2}$ that will enable U_2 to proxy sign on behalf of U_1. Now if U_2 wants to re-delegate the received signing rights for a possibly reduced set of tasks $TList' \subseteq TList$ to user U_3 holding pk_3, she runs $\mathsf{Del}(sk_2, warr_{1\rightarrow 2}, TList', pk_3)$, that is, with her warrant as additional argument, to produce $warr_{1\rightarrow 2\rightarrow 3}$. Every user in possession of a warrant valid for a task $task$ can produce proxy signatures σ for messages M corresponding to $task$ via $\mathsf{PSig}(sk, warr, task, M)$.[2] Anyone can then verify σ under the public key pk_1 of the first delegator (sometimes called "original signer" in the literature) by running $\mathsf{PVer}(pk_1, task, M, \sigma)$.

Finally, using the opening key ok_1 corresponding to pk_1, $\mathsf{Open}(ok_1, task, M, \sigma)$ opens a signature σ by returning the list of users that have re-delegated as well as the proxy signer.[3] Note that for simplicity, we identify users with their public keys, so Open returns a list of public keys. Figure 1 gives an overview of the algorithms constituting an anonymous proxy signature scheme.

Consider a warrant established by executions of Del with correctly registered keys. Then for any task and message we require that the signature produced on it pass verification.

Remark (Differences to the Model for Proxy Signatures). The specification deviates from the one in [BPW03] in the following points: First, dealing with anonymous proxy signatures, in our model there is no general *proxy identification* algorithm; instead, only authorized openers holding a special key may revoke anonymity. Second, in contrast to the above specifications, the *proxy-designation protocol* in [BPW03] is a pair of interactive algorithms and the *proxy signing* algorithm takes a single input, the *proxy signing key skp*. However, by simply defining the proxy part of the proxy-designation protocol as

$$skp := (sk, warr) \ ,$$

any scheme satisfying our specifications is easily adapted to theirs.

[2] Note that it depends on the concrete application to check whether M lies within the scope of $task$.
[3] We include $task$ and M in the parameters of Open so the opener can verify the signature before opening it.

```
$\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)$
    $(pp, ik, ock) \leftarrow \mathsf{Setup}(1^\lambda)$
    $(\text{ST}, pk, (sk^0, warr^0), (sk^1, warr^1), task, M) \leftarrow A_1(pp, ik : \mathsf{USndToO}, \mathsf{ISndToO}, \mathsf{OK}, \mathsf{Open})$
    if $pk \notin OReg$, return 0
    for $c = 0 \mathinner{..} 1$
        $\sigma^c \leftarrow \mathsf{PSig}(sk^c, warr^c, task, M)$
        if $\mathsf{PVer}(pk, task, M, \sigma^c) = 0$, return 0
        $(pk^c_2, \ldots, pk^c_{k_c}) \leftarrow \mathsf{Open}(\mathsf{OK}(pk), task, M, \sigma^c)$
    if opening succeeded and $k_0 \neq k_1$, return 0
    $d \leftarrow A_2(\text{ST}, \sigma^b : \mathsf{Open})$
    if $A_1$ did not query $\mathsf{OK}(pk)$ and $A_2$ did not query $\mathsf{Open}(pk, task, M, \sigma^b)$, return $d$,
    else return 0
```

Fig. 2. Experiment for ANONYMITY

3 Security Definitions

3.1 Anonymity

Anonymity ensures that signatures do not leak information on the identities of the intermediate delegators and the proxy signer, even in the presence of a corrupt issuer. However, the *number* of delegators involved may not remain hidden, as an openable signature must contain information about the delegators, whose number is not a priori bounded.

A quite "holistic" approach to define anonymity is the following experiment in the spirit of CCA2-indistinguishability: The adversary A, who controls the issuer and all users, is provided with an oracle to communicate with an honest opening authority. A may also query opening keys and the opening of signatures. Eventually, he outputs a public key, a message, a task and two secret-key/warrant pairs under one of which he is given a signature. Now A must decide which pair has been used to sign. Note that our definition implies all conceivable anonymity notions, such as proxy-signer anonymity, last-delegator anonymity, etc.

Figure 2 depicts the experiment, which might look more complex than expected, as there are several checks necessary to prevent the adversary from trivially winning the game by either

1. returning a public key he did not register with the opener,
2. returning an invalid warrant, that is, signatures created with it fail verification, or
3. having different lengths of delegation chains.[4]

The experiment simulates an honest opener as specified by Reg with whom the adversary communicates via the $\mathsf{USndToO}$ and $\mathsf{ISndToO}$ oracles, depending on whether he impersonates a user or the issuer. It also keeps a list $OReg$ of the opening keys it created and the corresponding public keys. Oracle OK, called with a public key, returns the related opening key from $OReg$ and when Open is called on $(pk', task', M', \sigma')$, the experiment looks up the corresponding opening key ok' and returns $\mathsf{Open}(ok', M', task', \sigma')$ if pk' has been registered and $\perp$ otherwise.

Definition 1 (Anonymity). A proxy signature scheme $\mathcal{PS}$ is ANONYMOUS if for any probabilistic polynomial-time (p.p.t.) adversary $A = (A_1, A_2)$, we have

$$\left| \Pr\left[\mathbf{Exp}^{\text{anon-1}}_{\mathcal{PS},A}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{anon-0}}_{\mathcal{PS},A}(\lambda) = 1\right] \right| \;=\; \text{negl}(\lambda) \ .$$

[4] The experiment checks 2. and 3. by producing a signature with each of the returned warrants and opening both to check if the number of delegators match. Note, that traceability (cf. Sect. 3.2) guarantees that valid signatures can be opened.

```
$\mathbf{Exp}^{\text{trace}}_{\mathcal{PS},A}(\lambda)$
    $(pp, ik, ock) \leftarrow \mathsf{Setup}(1^\lambda)$
    $(pk, task, M, \sigma) \leftarrow A(pp : \mathsf{SndToI}, \mathsf{SndToO})$
    if $\mathsf{PVer}(pk, task, M, \sigma) = 1$ and $\mathsf{Open}(\mathsf{OK}(pk), task, M, \sigma) = \perp$
        return 1, else return 0
```

Fig. 3. Experiment for TRACEABILITY

Remark (Hiding the Number of Delegations). A feature of our scheme is that users are able to delegate themselves. It is because of this fact—useful per se to create temporary keys for oneself to use in hostile environments—that one could define the following variant of the scheme:

Suppose there is a maximum number of possible delegations and that before signing, the proxy extends the actual delegation chain in her warrant to this maximum by consecutive self-delegations. The scheme would then satisfy a stronger notion of anonymity where even the number of delegations remains hidden. What is more, defining standard (non-proxy) signatures as self-delegated proxy signatures, even proxy and standard signatures become indistinguishable.

Since we also aim at constructing a generalization of group signatures in accordance with [BSZ05], we split the definition of what is called *security* in [BPW03] into two parts: traceability and non-frameability. We thereby achieve stronger security guarantees against malicious issuers.

3.2 Traceability

Consider a coalition of corrupt users and openers (the latter however following the protocol) trying to forge signatures. Then traceability guarantees that whenever a signature passes verification, it can be opened.[5]

In the game for traceability we let the adversary A register corrupt users and see the communication between issuer and opener. To win the game, A must output a signature and a public key under which it is valid such that opening of the signature fails.

Figure 3 shows the experiment for traceability, where the oracles SndToI and SndToO simulate issuer and opener respectively, according to the protocol Reg. In addition, they return a transcript of the communication between them. The experiment maintains a list of generated opening keys, so OK returns the opening key associated to the public key it is called with, or $\perp$ in case the key is not registered—in which case Open returns $\perp$, too.

Definition 2 (Traceability). A proxy signature scheme $\mathcal{PS}$ is TRACEABLE if for any p.p.t. adversary A, we have

$$\Pr\left[\mathbf{Exp}^{\text{trace}}_{\mathcal{PS},A}(\lambda) = 1\right] \ = \ \text{negl}(\lambda) \ .$$

3.3 Non-Frameability

Non-frameability ensures that no user is wrongfully accused of delegating or signing. In order to give a strong definition of non-frameability by according the adversary as much liberty as possible in his oracle queries, we require an additional functionality of the scheme: function OpenW applied to a warrant returns the list of delegators involved in creating it.

In the non-frameability game, the adversary can impersonate the issuer and the opener as well as corrupt users. He is given *all* keys created in the setup, and oracles to register honest users and query

[5] The issuer is assumed to behave honestly as he can easily create unopenable signatures by registering dummy users and sign in their name. The openers are *partially* corrupt, otherwise they could simply refuse to open or not correctly register the opening keys.

```
$\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}(\lambda)$
    $(pp, ik, ock) \leftarrow \mathsf{Setup}(1^\lambda)$
    $(ok, pk_1, task, M, \sigma) \leftarrow A(pp, ik, ock : \mathsf{ISndToU}, \mathsf{OSndToU}, \mathsf{SK}, \mathsf{Del}, \mathsf{PSig})$
    if $\mathsf{PVer}(pk_1, task, M, \sigma) = 0$ or $\mathsf{Open}(ok, task, M, \sigma) = \perp$, return 0
    $(pk_2, \ldots, pk_k) = \mathsf{Open}(ok, task, M, \sigma)$
    if $pk_1 \in HU$ and no queries $\mathsf{Del}(pk_1, TList, pk_2)$ with $TList \ni task$ made
            return 1                                                        (Case 1)
    if for some $i \geq 2$, $pk_i \in HU$ and no queries $\mathsf{Del}(pk_i, warr, TList, pk_{i+1})$ with
          $TList \ni task$ and $\mathsf{OpenW}(warr) = (pk_1, \ldots, pk_i)$ made, return 1      (Case 2)
    if $pk_k \in HU$ and no queries $\mathsf{PSig}(pk_k, warr, task, M)$ made
          with $\mathsf{OpenW}(warr) = (pk_1, \ldots, pk_k)$ made, return 1          (Case 3)
    return 0
```

Fig. 4. Experiment for Non-Frameability

delegations and proxy signatures from them. To win the game, the adversary must output a task, a message and a valid signature on it, such that the opening reveals either

1. a second delegator or proxy signer who was never delegated by an honest original delegator for the task,
2. an honest delegator who was not queried the respective delegation for the task, or
3. an honest proxy signer who did not sign the message for the task and the respective delegation chain.

We emphasize that impersonating U_1, U_1' and U_3, querying re-delegation from honest user U_2 to U_3 with a warrant from U_1 for U_2 and then producing a signature that opens to (U_1', U_2, U_3) is considered a successful attack. Note furthermore that it is the adversary that chooses the opening key to be used. See Fig. 4 for the experiment for non-frameability.

Oracles for non-frameability: ISndToU (OSndToU) enables the adversary impersonating a corrupt issuer (opener) to communicate with an honest user. When first called without arguments, the oracle simulates a new user starting the registration procedure and makes a new entry in HU, the list of honest users. Oracles Del and PSig are called with a user's public key, which the experiment replaces by the user's secret key from HU before executing the respective function; e.g., calling Del with parameters $(pk_1, TList, pk_2)$ returns $\mathsf{Del}(sk_1, TList, pk_2)$. Oracle SK takes a public key pk as argument and returns the corresponding private key after deleting pk from HU.

Definition 3 (Non-frameability). A proxy signature scheme $\mathcal{PS}$ is non-frameable if for any p.p.t. adversary A we have

$$\Pr\left[\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}(\lambda) = 1\right] \;=\; \text{negl}(\lambda) \enspace .$$

Remark. In the experiment $\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}$, the opening algorithm is run by the experiment, which by definition behaves honestly. To guard against corrupt openers, it suffices to add a (possibly interactive) zero-knowledge proof of correctness of opening.

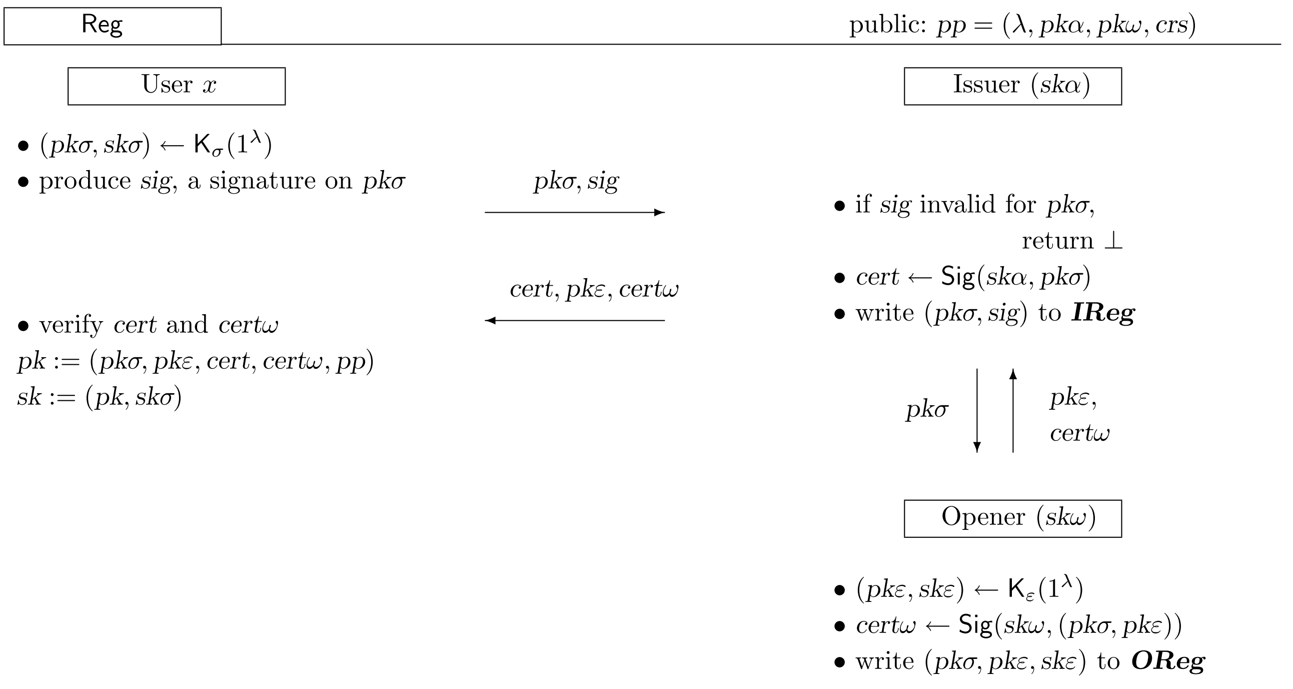

Reg
public: $pp = (\lambda, pk\alpha, pk\omega, crs)$
User x
Issuer $(sk\alpha)$
• $(pk\sigma, sk\sigma) \leftarrow \mathsf{K}_\sigma(1^\lambda)$
• produce sig, a signature on $pk\sigma$
$pk\sigma, sig$
• if sig invalid for $pk\sigma$, return $\perp$
$cert, pk\varepsilon, cert\omega$
• $cert \leftarrow \mathsf{Sig}(sk\alpha, pk\sigma)$
• write $(pk\sigma, sig)$ to ***IReg***
• verify $cert$ and $cert\omega$
$pk := (pk\sigma, pk\varepsilon, cert, cert\omega, pp)$
$sk := (pk, sk\sigma)$
$pk\sigma$
$pk\varepsilon$, $cert\omega$
Opener $(sk\omega)$
• $(pk\varepsilon, sk\varepsilon) \leftarrow \mathsf{K}_\varepsilon(1^\lambda)$
• $cert\omega \leftarrow \mathsf{Sig}(sk\omega, (pk\sigma, pk\varepsilon))$
• write $(pk\sigma, pk\varepsilon, sk\varepsilon)$ to ***OReg***

Fig. 5. Registration protocol

4 An Instantiation of the Scheme

4.1 Building Blocks

To construct the generic scheme $\mathcal{PS}$, we will use the following standard cryptographic primitives (formally defined in Appendix A) whose existence is implied by assuming trapdoor permutations [Rom90,DDN00,Sah99].

- $\mathcal{DS} = (\mathsf{K}_\sigma, \mathsf{Sig}, \mathsf{Ver})$, a digital signature scheme secure against existential forgeries under chosen-message attack [GMR88].
- $\mathcal{PKE} = (\mathsf{K}_\varepsilon, \mathsf{Enc}, \mathsf{Dec})$, a public-key encryption scheme with indistinguishable encryptions under adaptive chosen-ciphertext attack (CCA2) [RS92].
- $\Pi = (\mathsf{P}, \mathsf{V}, \mathsf{Sim})$, a non-interactive zero-knowledge (NIZK) proof system for an NP-language to be defined in the following that is simulation sound [BDMP91,Sah99].

4.2 Algorithms

The algorithm Setup establishes the public parameters and outputs the issuer's and the opener's certification key. The public parameters consist of the security parameter, a common random string for non-interactive zero-knowledge proofs and the two signature verification keys corresponding to the issuer's and the opener's key:

```
Setup
$1^\lambda \rightarrow$        | $(pk\alpha, sk\alpha) \leftarrow \mathsf{K}_\sigma(1^\lambda)$;  $(pk\omega, sk\omega) \leftarrow \mathsf{K}_\sigma(1^\lambda)$;  $crs \leftarrow \{0,1\}^{p(\lambda)}$
$pp, ik, ock \leftarrow$ | $pp := (\lambda, pk\alpha, pk\omega, crs)$;  $ik := sk\alpha$;  $ock := sk\omega$
```

The registration protocol is depicted in Fig. 5. When a user joins the system, she creates a pair of verification/signing keys $(pk\sigma, sk\sigma)$ and *signs* $pk\sigma$ (e.g. via an external PKI) in order to commit to it. She then sends $pk\sigma$ and the signature *sig* to the issuer. The latter, after checking *sig*, signs $pk\sigma$ with his *certificate issuing key* $sk\alpha$ and writes the user data to ***IReg***, the registration table.

In addition, the issuer sends $pk\sigma$ to the authority responsible for opening the user's signatures. The opener creates an encryption/decryption key pair $(pk\varepsilon, sk\varepsilon)$ and a certificate on $pk\varepsilon$ and $pk\sigma$, which together with $pk\varepsilon$ he sends to the issuer, who forwards it to the user.[6]

Remark (Attaining Non-Frameability). It is by having the users create their own signing keys $sk\sigma$ that a corrupt authority is prevented from framing them. The user is however required to commit to her verification key via *sig*, so that she cannot later repudiate signatures signed with the corresponding signing key. Now to frame a user by creating a public key and attributing it to her, the issuer would have to forge *sig*. Note that it is impossible to achieve non-frameability without assuming some sort of PKI prior to the scheme.

Algorithm Del enables user x to pass her signing rights to user y (if called with no optional argument $\boldsymbol{warr}_{\text{old}}$), or to re-delegate the rights represented by $\boldsymbol{warr}_{\text{old}}$ for the tasks in *TList*. A warrant is an array where $\boldsymbol{warr}[i]$ corresponds to the i^{th} delegation and $\boldsymbol{warr}[i][task]$ basically contains a signature by the i^{th} delegator on the next delegator's public key and *task*.

More specifically, consider user x being the k^{th} delegator. If $k > 1$, she first copies all entries for the tasks to re-delegate from $\boldsymbol{warr}_{\text{old}}$ to the new warrant $\boldsymbol{warr}$. She then writes her public key to $\boldsymbol{warr}[k][0]$, which will later be used by the delegatee, and finally produces a signature on the task, the public keys of the delegators, her and the delegatee's public key and writes it to $\boldsymbol{warr}[k][task]$.

```
Del
$sk_x$, [$\boldsymbol{warr}_{\text{old}}$,]   | parse $sk_x \rightsquigarrow (pk_x, sk\sigma)$; $k := |\boldsymbol{warr}_{\text{old}}| + 1$    // $k = 1$ if no $\boldsymbol{warr}_{\text{old}}$
       $TList, pk_y$ →  | for all $1 \le i < k$
                        |     $\boldsymbol{warr}[i][0] := \boldsymbol{warr}_{\text{old}}[i][0]$
                        |     for all $task \in TList$, $\boldsymbol{warr}[i][task] := \boldsymbol{warr}_{\text{old}}[i][task]$
                        | $\boldsymbol{warr}[k][0] := pk_x$
                        | for all $1 \le i \le k$, parse $\boldsymbol{warr}[i][0] \rightsquigarrow (pk\sigma_i, pk\varepsilon_i, cert_i, cert\omega_i, pp)$
                        | for all $task \in TList$
       $\boldsymbol{warr}$ ← |     $\boldsymbol{warr}[k][task] \leftarrow \mathsf{Sig}\big(sk\sigma, (task, pk\sigma_1, \ldots, pk\sigma_k, pk\sigma_y)\big)$
```

In order to prove correctness of an anonymous signature, we define a relation R_k, specifying an NP-language L_{R_k}. Basically, a theorem $(pk\alpha, pk\omega, pk\sigma_1, pk\varepsilon_1, cert\omega_1, task, M, C)$ is in L_{R_k} if and only if

(1) $pk\varepsilon_1$ is correctly certified w.r.t. $pk\omega$,

(2) there exist verification keys $pk\sigma_2, \ldots, pk\sigma_k$ that are correctly certified w.r.t. $pk\alpha$,

(3) there exist warrant entries $warr_i$ for $1 \le i < k$, s.t. $pk\sigma_i$ verifies the delegation chain $pk_1 \to \cdots \to pk_{i+1}$ for *task*,

(4) there exists a signature s on the delegation chain and M valid under $pk\sigma_k$,

(5) C is an encryption using some randomness ρ of all the verification keys, certificates, warrants and the signature s.

[6] In practice, our protocol would allow for the opener to communicate directly with the user without the detour via the issuer—for example in the case where each user is his own opener. We define the protocol this way to simplify exposition of the security proofs.

We define formally:

$$\begin{aligned}
&R_k\big[(pk\alpha, pk\omega, pk\sigma_1, pk\varepsilon_1, cert\omega_1, task, M, C),\\
&\qquad\qquad (pk\sigma_2, \ldots, pk\sigma_k, cert_2, \ldots, cert_k, warr_1, \ldots, warr_{k-1}, s, \rho)\big]\\
&:\Leftrightarrow\ \mathsf{Ver}\big(pk\omega, (pk\sigma_1, pk\varepsilon_1), cert\omega_1\big) = 1\ \wedge && (1)\\
&\quad \textstyle\bigwedge_{2\le i\le k} \mathsf{Ver}\big(pk\alpha, pk\sigma_i, cert_i\big) = 1\ \wedge && (2)\\
&\quad \textstyle\bigwedge_{1\le i\le k-1} \mathsf{Ver}\big(pk\sigma_i, (task, pk\sigma_1, \ldots, pk\sigma_{i+1}), warr_i\big) = 1\ \wedge && (3)\\
&\quad \mathsf{Ver}\big(pk\sigma_k, (task, pk\sigma_1, \ldots, pk\sigma_k, M), s\big) = 1\ \wedge && (4)\\
&\quad \mathsf{Enc}\big(pk\varepsilon_1, (pk\sigma_2, \ldots, pk\sigma_k, cert_2, \ldots, cert_k, warr_1, \ldots, warr_{k-1}, s), \rho\big) = C && (5)
\end{aligned}$$

Note that for every k, the above relation R_k defines an NP-language L_{R_k}, since given a witness, membership of a candidate theorem is efficiently verifiable and the length of a witness is polynomial in the length of the theorem. Let $\Pi_k := (\mathsf{P}_k, \mathsf{V}_k, \mathsf{Sim}_k)$ be a simulation-sound NIZK proof system for L_{R_k}.

Now to produce a proxy signature, it suffices to sign the delegation chain and the message, encrypt it together with all the signatures for the respective task from the warrant and prove that everything was done correctly, that is, prove that R_k is satisfied:

```
PSig
sk, warr,          | $k := |\mathbf{warr}| + 1$, parse $sk \rightsquigarrow (pk_k, sk\sigma)$
     task, M →     |         parse $pk_k \rightsquigarrow \big(pk\sigma_k, pk\varepsilon_k, cert_k, cert\omega_k, (\lambda, pk\alpha, pk\omega, crs)\big)$
                   | for $1 \le i < k$, parse $\mathbf{warr}[i][0] \rightsquigarrow (pk\sigma_i, pk\varepsilon_i, cert_i, cert\omega_i, pp)$
                   |         set $warr_i := \mathbf{warr}[i][task]$
                   | $s \leftarrow \mathsf{Sig}\big(sk\sigma, (task, pk\sigma_1, \ldots, pk\sigma_k, M)\big)$;  $\rho \leftarrow \{0,1\}^{p_\varepsilon(\lambda,k)}$
                   | $W := (pk\sigma_2, \ldots, pk\sigma_k, cert_2, \ldots, cert_k, warr_1, \ldots, warr_{k-1}, s)$
                   | $C \leftarrow \mathsf{Enc}(pk\varepsilon_1, W; \rho)$
                   | $\pi \leftarrow \mathsf{P}_k\big(1^\lambda, (pk\alpha, pk\omega, pk\sigma_1, pk\varepsilon_1, warr\omega_1, task, M, C), W \,\|\, \rho, crs\big)$
             σ ←   | $\sigma := (C, \pi)$
```

Verifying a proxy signature then amounts to verifying the proof it contains:

```
PVer
pk_x, task, M, σ → | parse $pk_x \rightsquigarrow \big(pk\sigma_x, pk\varepsilon_x, cert_x, cert\omega_x, (\lambda, pk\alpha, pk\omega, crs)\big)$;  $\sigma \rightsquigarrow (C, \pi)$
               b ← | $b := \mathsf{V}_k\big(1^\lambda, (pk\alpha, pk\omega, pk\sigma_x, pk\varepsilon_x, cert\omega_x, task, M, C), \pi, crs\big)$
```

To open a signature check its validity and decrypt the contained ciphertext:

```
Open
ok_x, task, M, σ →     | parse $ok_x \rightsquigarrow (pk_x, sk\varepsilon_x)$;  $\sigma \rightsquigarrow (C, \pi)$
                       | parse $pk_x \rightsquigarrow \big(pk\sigma_x, pk\varepsilon_x, cert_x, cert\omega_x, (\lambda, pk\alpha, pk\omega, crs)\big)$
                       | if $\mathsf{V}_k\big(1^\lambda, (pk\alpha, pk\omega, pk\sigma_x, pk\varepsilon_x, cert\omega_x, task, M, C), \pi, crs\big) = 0$
                       |     return $\perp$
                       | $(pk\sigma_2, \ldots, pk\sigma_k, cert_2, \ldots, cert_k, warr_1, \ldots, warr_{k-1}, s) := \mathsf{Dec}(sk\varepsilon_x, C)$
(pk_2, .., pk_k) ←     | if for some $i$, $pk_i$ is not in $\boldsymbol{IReg}$, return $\perp$
```

```
$\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)$
    $crs \leftarrow \{0,1\}^{p(\lambda)}$
    $(pk\alpha, sk\alpha) \leftarrow \mathsf{K}_\sigma(1^\lambda)$;  $(pk\omega, sk\omega) \leftarrow \mathsf{K}_\sigma(1^\lambda)$;  $pp := (\lambda, pk\alpha, pk\omega, crs)$
    $\big(\textsc{st}, pk, (warr^0, sk^0), (warr^1, sk^1), task, M\big) \leftarrow A_1(pp, sk\alpha \,:\, \mathsf{ISndToO}, \mathsf{OK}, \mathsf{Open})$
    if $pk \notin OReg$, return 0, else parse $pk \rightsquigarrow (pk\sigma^*, pk\varepsilon^*, cert^*, cert\omega^*, pp)$
    if $|warr^0| \neq |warr^1|$, return 0, else $k := |warr| + 1$
    for $c = 0 \,..\, 1$
        parse $sk^c \rightsquigarrow \big((pk\sigma_k^c, pk\varepsilon_k^c, cert_k^c, cert\omega_k^c, pp), sk\sigma^c\big)$
        for $i = 1 \,..\, k-1$:   $pk_i^c := warr^c[i][0] \rightsquigarrow (pk\sigma_i^c, pk\varepsilon_i^c, cert_i^c, cert\omega_i^c, pp)$
        $s^c \leftarrow \mathsf{Sig}(sk\sigma^c, (task, pk\sigma_1^c, \ldots, pk\sigma_k^c, M)$
        $m^c := (pk\sigma_2^c, \ldots, pk\sigma_k^c, cert_2^c, \ldots, cert_k^c, warr^c[1][task], \ldots, warr^c[k-1][task], s)$
        if $R_k^*(pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, task, M), m^c) = 0$, return 0
    $\rho \leftarrow \{0,1\}^{p_\varepsilon(\lambda,k)}$;  $C \leftarrow \mathsf{Enc}(pk\varepsilon^*, m^b \;;\, \rho)$
    $\pi \leftarrow \mathsf{P}_k\big(1^\lambda, (pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, task, M, C), \; m^b \,\|\, \rho, \; crs\big)$
    return $d \leftarrow A_2\big(\textsc{st}, (C, \pi) \,:\, \mathsf{Open}\big)$
```

```
Oracle $\mathcal{O}_{\mathsf{OK}}((pk\sigma^*, \cdot\cdot))$
    if $(pk\sigma^*, \cdot, \cdot, sk\varepsilon) \in OReg$
            for some $sk\varepsilon$
    return $sk\varepsilon$
```

```
Oracle $\mathcal{O}_{\mathsf{ISndToO}}(pk\sigma)$
    $(pk\varepsilon, sk\varepsilon) \leftarrow \mathsf{K}_\varepsilon(1^\lambda)$
    $cert\omega \leftarrow \mathsf{Sig}\big(sk\omega, (pk\sigma, pk\varepsilon)\big)$
    save $(pk\sigma, pk\varepsilon, cert\omega, sk\varepsilon)$ in $OReg$
    return $(pk\varepsilon, cert\omega)$
```

Fig. 6. Experiment for anonymity

4.3 Security Results

From the definition of the algorithms, it should be apparent that running PSig with a warrant correctly produced by registered users returns a signature which is accepted by PVer and correctly opened by Open. Moreover, the defined scheme satisfies all security notions from Sect. 3.

Lemma 4. *The proxy signature scheme $\mathcal{PS}$ is* ANONYMOUS *(Definition 1).*

Proof. The natural way to prove anonymity is by reduction to indistinguishability of the underlying encryption scheme: if the adversary can distinguish between two signatures (C_1, π_1) and (C_2, π_2), it must be by distinguishing C_1 from C_2, as the proofs π_i are zero-knowledge. (Simulating the proofs does not alter the experiments in any computationally distinguishable manner and could be performed by the adversary itself.) The only case that needs special treatment in the reduction is when the $\mathcal{PS}$ adversary, after being challenged on $\sigma = (C, \pi)$, queries (C, π')—which is perfectly legitimate, but poses a problem to the $\mathcal{PKE}$-adversary, which cannot forward C to its decryption oracle.

Without loss of generality, we assume that the adversary is *honest* in that it does not query $\mathsf{OK}(pk)$ or $\mathsf{Open}\big(pk, task, M, (C, \pi)\big)$. (Note that any adversary A can be transformed into an honest one having the same success probability by simulating A and outputting $d \leftarrow \{0,1\}$ if A makes an illegal query.)

Figure 6 shows the experiment for anonymity after plugging in the algorithm definitions and some simplifications. Relation R_k^* is defined as R_k restricted to the first 4 clauses, i.e., there is no check of encryption (which does not alter the experiment, since encryption is performed correctly by the experiment anyway). Note also that due to the communication between the parties defined in Reg, the $\mathsf{USndToO}$ oracle is obsolete, and due to honesty of A, we can omit the checks for illegal oracle queries at the end of the experiment.

We define a first variant of the original experiment by substituting the zero-knowledge proof π by a simulated one. Claim 1 then states that the variant is computationally indistinguishable from the original one.[7]

```
$\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)^{(1)}$
    1  $(crs, \textsc{st}_S) \leftarrow \mathsf{Sim}_1(1^\lambda)$
       $\vdots$
    13 $\pi \leftarrow \mathsf{Sim}_2\big(\textsc{st}_S, (pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert^*, task, M, C)\big)$
       $\vdots$
```

Claim 1. $\big|\Pr[\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)^{(1)} = 1]\big| \;\leq\; \mathbf{Adv}^{\text{zk}}_{\Pi,D}(\lambda)$, *where D is a p.p.t. algorithm that in the first stage, on input crs, runs $\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)$ from line 2 to 12 and outputs $(pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert^*, task, M, C), m^b \parallel \rho)$. After receiving π in the second stage, D continues simulating line 14.*[8]

Proof. The claim follows from equivalence of the following random variables:

$$\mathbf{Exp}^{\text{zk}}_{\Pi,D}(\lambda) \;=\; \mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda) \quad\text{and}\quad \mathbf{Exp}^{\text{zk-S}}_{\Pi,D}(\lambda) \;=\; \mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)^{(1)}\;. \qquad \square$$

Next, we define a second variant that can then be perfectly simulated by an adversary B against $\mathcal{PKE}$:

```
$\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)^{(2)}$
       $\vdots$
    14 $d \leftarrow A_2\big(\textsc{st}, (C, \pi) : \mathsf{Open}\big)$
    15 if A made a *valid* query $\mathsf{Open}\big(pk, task, M, (C, \pi')\big)$, return 0, else return $d$
```

Claim 2. $\big|\Pr[\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)^{(1)} = 1] - \Pr[\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)^{(2)} = 1]\big| \;=\; \text{negl}(\lambda)\;.$

(See below for the proof.) Due to the above claims, in order to proof Lemma 4, it suffices to relate $\Pr[\mathbf{Exp}^{\text{anon-b(2)}}_{\mathcal{PS},A} = 1]$ to $\Pr[\mathbf{Exp}^{\text{ind-cca\_b}}_{\mathcal{PKE},B} = 1]$. Let n be the maximal number of $\mathsf{ISndToO}$ queries performed by A. We construct an adversary against the encryption scheme that, on guessing the right user, perfectly simulates $\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)^{(2)}$:

```
Adversary $B_1(\overline{pk} : \mathsf{Dec})$
    1  $j^* \leftarrow \{1, \ldots, n\}$; $j := 0$; $(crs, \textsc{st}_S) \leftarrow \mathsf{Sim}_1(1^\lambda)$
       $\vdots$
    12 return $(m^0, m^1, \textsc{status})$

Adversary $B_2(\textsc{status}, C : \mathsf{Dec})$
       $\pi \leftarrow \mathsf{Sim}_2(\textsc{st}_S, (pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, task, M, C)$
       $d \leftarrow A_2\big(\textsc{st}, (C, \pi) : \mathsf{Open}\big)$
       if A made a *valid* query $\mathsf{Open}\big(pk, task, M, (C, \pi')\big)$, return 0, else return $d$
```

[7] For ease of presentation, we only give the lines of the experiment that changed.

[8] We use $\mathbf{Adv}^{\text{zk}}_{\bullet\bullet}(\cdot)$ as shortcut for $|\Pr[\mathbf{Exp}^{\text{zk}}_{\bullet\bullet}(\cdot) = 1] - \Pr[\mathbf{Exp}^{\text{zk-S}}_{\bullet\bullet}(\cdot) = 1]|$ and similarly for indistinguishability. For all other experiments, $\mathbf{Adv}^{\bullet}_{\bullet\bullet}(\cdot)$ denotes $\Pr[\mathbf{Exp}^{\bullet}_{\bullet\bullet}(\cdot) = 1]$.

```
Oracle $\mathcal{O}_{\mathsf{ISndToO}}(pk\sigma)$ by $B_1$
        $j := j + 1$
        if $j = j^*$ then $pk\varepsilon := \overline{pk}$;  else $(pk\varepsilon, sk\varepsilon) \leftarrow \mathsf{K}_\varepsilon(1^\lambda)$
        $cert\omega \leftarrow \mathsf{Sig}\big(sk\omega, (pk\sigma, pk\varepsilon)\big)$;  write $(pk\sigma, pk\varepsilon, cert\omega)$ to $OReg$
        return $(pk\varepsilon, cert\omega)$
```

When A calls its Open oracle for a public key containing $\overline{pk}$ and a valid signature (C', π'), B does the following: If $C' \neq C$, B uses its own Dec oracle to decrypt C'; if the signature contains the challenge C then B returns 0 anyway.

Consider the experiment when A returns pk containing $\overline{pk}$ (which happens with probability at least $\frac{1}{n(\lambda)}$). First, note that m^0 and m^1 are of equal length, for R^* guarantees that the warrants are formed correctly. Moreover, B no illegal queries C. We have thus

$$\Pr[\mathbf{Exp}^{\text{ind-cca-b}}_{\mathcal{PKE},B}(\lambda) = 1] \;\geq\; \tfrac{1}{n(\lambda)} \Pr[\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)^{(2)} = 1] \;. \tag{6}$$

On the other hand, by indistinguishability of $\mathcal{PKE}$, we have:

$$\big|\Pr[\mathbf{Exp}^{\text{ind-cca-1}}_{\mathcal{PKE},B}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\text{ind-cca-0}}_{\mathcal{PKE},B}(\lambda) = 1]\big| \;=\; \text{negl}(\lambda) \;,$$

which, because of (6) and Claims 1 and 2 yields:

$$\big|\Pr[\mathbf{Exp}^{\text{anon-1}}_{\mathcal{PS},A}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\text{anon-0}}_{\mathcal{PS},A}(\lambda) = 1]\big| \;=\; \text{negl}(\lambda) \;.$$

We conclude by proving the second claim.

Proof (of Claim 2). We show that after receiving (C, π), A is very unlikely to make a *valid* open query (C, π'), i.e., create a different proof π' for the statement

$$(pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, M, task, C) =: X \;.$$

If X was not in L_R, then due to simulation soundness of Π_k, such a query happens only with negligible probability. However, indistinguishability of ciphertexts implies that the same holds for $X \in L_R$, otherwise based on $\mathbf{Exp}^{\text{anon-b}(1)}_{\mathcal{PS},A}$ we could build a distinguisher B^b for $\mathcal{PKE}$ as follows:

```
Adversary $B^b_1(\overline{pk} : \mathsf{Dec})$
        ⋮
     12 return $(0^{|m^b|}, m^b, \textsc{status})$

Adversary $B^b_2(\textsc{status}, C : \mathsf{Dec})$
        $\pi \leftarrow \mathsf{Sim}_2\big(\textsc{st}_S, (pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, M, task, C)\big)$
        $d \leftarrow A_2(\textsc{st}, (C, \pi) : \mathsf{Open})$
        if at some point $A$ queries $(C, \pi')$ with $\pi' \neq \pi$ and
                $\mathsf{V}_k(1^\lambda, \big(pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, M, task, C), \pi', R\big) = 1$ then return 1
        else return 0
```

and a simulation-soundness adversary $S^{b,c}$ that runs $\mathbf{Exp}^{\text{ind-c}}_{\mathcal{PKE},B^b}$, except for having *crs* and π as input from its experiment instead of creating them itself. Now when when A first makes a valid query (C, π'), it outputs $\big(X := (pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, M, task, C), \pi'\big)$, and fails otherwise. We have

$$\big|\Pr[\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)^{(1)} = 1] - \Pr[\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)^{(2)} = 1]\big| \;\leq\; \Pr[E_b] \;,$$

where E_b denotes the event that in $\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}$, A makes a valid query (C,π'). It remains to bound the probability of event E_b. On the one hand, we have (note that $pk\varepsilon^* \neq \overline{pk}$ implies $X \notin L_R$, and thus $S^{b,1}$ succeeds in this case):

$$\begin{aligned}\Pr[\mathbf{Exp}^{\text{ind-1}}_{\mathcal{PKE},B^b}(\lambda)=1] &= \Pr[\mathbf{Exp}^{\text{ind-1}}_{\mathcal{PKE},B^b}(\lambda)=1 \;\wedge\; pk\varepsilon^*=\overline{pk}] + \\ &\qquad\qquad \Pr[\mathbf{Exp}^{\text{ind-1}}_{\mathcal{PKE},B^b}(\lambda)=1 \;\wedge\; pk\varepsilon^*\neq\overline{pk}] \\ &= \tfrac{1}{n(\lambda)}\Pr[E_b] + \big(1-\tfrac{1}{n(\lambda)}\big)\Pr[\mathbf{Exp}^{\text{ss}}_{\varPi,S^{b,1}}(\lambda)=1] \ .\end{aligned}$$

On the other hand, we have

$$\Pr[\mathbf{Exp}^{\text{ind-0}}_{\mathcal{PKE},B^b}(\lambda)=1] = \Pr[\mathbf{Exp}^{\text{ss}}_{\varPi,S^{b,0}}(\lambda)=1] \ ,$$

since $(X, 0^{|m^b|}) \notin R$. Combining the above, we get

$$\mathbf{Adv}^{\text{ind}}_{\mathcal{PKE},B^b}(\lambda) \;=\; \Big|\,\tfrac{1}{n(\lambda)}\Pr[E_b] \;-\; \big((1-\tfrac{1}{n(\lambda)})\mathbf{Adv}^{\text{ss}}_{\varPi,S^{b,1}}(\lambda) + \mathbf{Adv}^{\text{ss}}_{\varPi,S^{b,0}}(\lambda)\big)\,\Big| \ ,$$

and thus the following, which proves the claim:

$$\Pr[E_b] \;\leq\; n(\lambda)\big(\mathbf{Adv}^{\text{ind}}_{\mathcal{PKE},B^b}(\lambda) + \mathbf{Adv}^{\text{ss}}_{\varPi,S^{b,1}}(\lambda) + \mathbf{Adv}^{\text{ss}}_{\varPi,S^{b,0}}(\lambda)\big) \ . \qquad \square$$

Lemma 5. *The proxy signature scheme $\mathcal{PS}$ is* TRACEABLE *(Definition 2).*

Proof. First, note that the requirement to have $pk\varepsilon$ certified by the opener prevents the adversary from trivially winning the game as follows: return a public key containing a different $pk\varepsilon'$ and use it to encrypt when signing to get a valid signature that is not openable with the opener's key.

Figure 7 shows $\mathbf{Exp}^{\text{trace}}_{\mathcal{PS},A}$ including the SndToI oracle rewritten with the code of the respective algorithms. Note that due to our implementation of Reg, the SndToO oracle is obsolete and that the communication between issuer and opener (i.e., $pk\sigma$, $pk\varepsilon$, $cert\omega$) is known to the adversary.

We construct two adversaries B_ω, B_α against existential unforgeability of $\mathcal{DS}$ that simulate $\mathbf{Exp}^{\text{trace}}_{\mathcal{PS},A}$, while using their input $\overline{pk}$ as either the opener's certifying key (B_ω) or the issuer's signing key (B_α). When answering A's SndToI queries, B_ω and B_α use their oracle for the respective signature.

```
Adversary $B_\omega(\overline{pk} : \mathsf{Sig})$
1  $(pk\alpha, sk\alpha) \leftarrow \mathsf{K}_\sigma(1^\lambda)$; $pk\omega := \overline{pk}$
   ⋮
6  if no entry $pk$ in $\boldsymbol{OReg}$
       return $((pk\sigma^*, pk\varepsilon^*), cert\omega^*)$
7  return $\perp$
```

```
Adversary $B_\alpha(\overline{pk} : \mathsf{Sig})$
1  $pk\alpha := \overline{pk}$; $(pk\omega, sk\omega) \leftarrow \mathsf{K}_\sigma(1^\lambda)$
   ⋮
8  if for some $i$, $pk\sigma_i$ not in $\boldsymbol{IReg}$
       return $(pk\sigma_i, cert_i)$
9  return $\perp$
```

Let E_1, E_2 and S denote the following events:

E_1	...	$\mathbf{Exp}^{\text{trace}}_{\mathcal{PS},A}(\lambda)$ returns 1 in line 6
E_2	...	$\mathbf{Exp}^{\text{trace}}_{\mathcal{PS},A}(\lambda)$ returns 1 in line 8
S	...	$(pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, task, M, C) \in L_R$

We have $\mathbf{Adv}^{\text{trace}}_{\mathcal{PS},A}(\lambda) = \Pr[E_1 \wedge S] + \Pr[E_2 \wedge S] + \Pr[(E_1 \vee E_2) \wedge \bar{S}]$. Showing that the three summands are negligible completes thus the proof.

```
$\mathbf{Exp}^{\text{trace}}_{\mathcal{PS},A}(\lambda)$
  1  $(pk\alpha, sk\alpha) \leftarrow \mathsf{K}_\sigma(1^\lambda)$;  $(pk\omega, sk\omega) \leftarrow \mathsf{K}_\sigma(1^\lambda)$
  2  $crs \leftarrow \{0,1\}^{p(\lambda)}$;  $pp := (\lambda, pk\alpha, pk\omega, crs)$
  3  $(pk, task, M, \sigma) \leftarrow A(pp : \mathsf{SndToI})$
  4  parse $pk \rightsquigarrow (pk\sigma^*, pk\varepsilon^*, cert^*, cert\omega^*, pp)$;  $\sigma \rightsquigarrow (C, \pi)$
  5  if $\mathsf{V}_k(1^\lambda, (pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, task, M, C), \pi, crs) = 0$, return 0
  6  if no entry $pk$ in $\boldsymbol{OReg}$, return 1              // opening fails
             otherwise look up the corresponding $sk\varepsilon^*$.
  7  $(pk\sigma_2, \ldots, pk\sigma_k, cert_2, \ldots, cert_k, warr_1, \ldots, warr_{k-1}, s) := \mathsf{Dec}(sk\varepsilon^*, C)$
  8  if for some $i$, $pk\sigma_i$ not in $\boldsymbol{IReg}$, return 1
  9  return 0

$\mathcal{O}_{\mathsf{SndToI}}(pk\sigma, sig)$
  1  if verification of $sig$ on $pk\sigma$ fails then return $\perp$
  2  $cert \leftarrow \mathsf{Sig}(sk\alpha, pk\sigma)$; write $(pk\sigma, sig)$ to $\boldsymbol{IReg}$
  3  $(pk\varepsilon, sk\varepsilon) \leftarrow \mathsf{K}_\varepsilon(1^\lambda)$;  $cert\omega \leftarrow \mathsf{Sig}(sk\omega, (pk\sigma, pk\varepsilon))$
  4  write $(pk\sigma, pk\varepsilon, sk\varepsilon)$ to $\boldsymbol{OReg}$
  5  return $(cert, pk\varepsilon, cert\omega)$
```

Fig. 7. Experiment for traceability

$E_1 \wedge S$: S means $(pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, task, M, C) \in L_R$, and thus

$$\mathsf{Ver}\big(pk\omega, (pk\sigma^*, pk\varepsilon^*), cert\omega^*\big) = 1 \ .$$

On the other hand, E_1 implies that $(pk\sigma^*, pk\varepsilon^*)$ is not in $\boldsymbol{OReg}$, thus B_ω never asked a signature on it and therefore returns a valid forgery. We have thus

$$\Pr[E_1 \wedge S] \ \leq \ \Pr[\mathbf{Exp}^{\text{euf-cma}}_{\mathcal{DS},B_\omega}(\lambda) = 1] \ .$$

$E_2 \wedge S$: Now, S implies that for all $2 \leq j \leq k : \mathsf{Ver}(pk\alpha, pk\sigma_j, cert_j) = 1$, but $pk\sigma_i$ being not in $\boldsymbol{IReg}$ means B_α returns a valid forgery, and consequently

$$\Pr[E_2 \wedge S] \ \leq \ \Pr[\mathbf{Exp}^{\text{euf-cma}}_{\mathcal{DS},B_\alpha}(\lambda) = 1] \ .$$

$(E_1 \vee E_2) \wedge \bar{S}$: $(E_1 \vee E_2)$ implies $\mathsf{V}_k(1^\lambda, (pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, task, M, C), \pi, crs) = 1$, which, together with $\bar{S}$ contradicts soundness of Π_k: based on $\mathbf{Exp}^{\text{trace}}_{\mathcal{PS},A}$, we could construct an adversary B_{s} against soundness of Π_k which after receiving crs (rather than choosing it itself), runs along the lines of the experiment until line 4 and then outputs $\big((pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, task, M, C), \pi\big)$. We have thus

$$\Pr[(E_1 \vee E_2) \wedge \bar{S}] \ \leq \ \mathbf{Adv}^{\text{ss}}_{\Pi,B_{\text{s}}} \ . \qquad \square$$

Lemma 6. *The proxy signature scheme $\mathcal{PS}$ is* NON-FRAMEABLE *(Definition 3).*

Proof. Figure 8 shows experiment $\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}$ rewritten with the code of the respective algorithms. Note that we can dispense with the $\mathsf{OSndToU}$-oracle, because in our scheme the user communicates exclusively with the issuer.

We construct an adversary B against the signature scheme $\mathcal{DS}$ having input a verification key $\overline{pk}$ and access to a signing oracle $\mathcal{O}_{\mathsf{Sig}}$. B simulates $\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS}}$ for A, except that for one random user

```
$\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}(\lambda)$
1  $(pk\alpha, sk\alpha) \leftarrow \mathsf{K}_\sigma(1^\lambda)$;  $(pk\omega, sk\omega) \leftarrow \mathsf{K}_\sigma(1^\lambda)$;  $crs \leftarrow \{0,1\}^{p(\lambda)}$
2  $pp := (\lambda, pk\alpha, pk\omega, crs)$
3  $(ok, pk_1, task, M, \sigma) \leftarrow A(pp, sk\alpha, sk\omega : \mathsf{ISndToU}, \mathsf{SK}, \mathsf{Del}, \mathsf{PSig})$
4  parse $ok \rightsquigarrow ((pk\sigma_1, pk\varepsilon_1, cert_1, cert\omega_1, pp), sk\varepsilon_1)$;  $\sigma \rightsquigarrow (C, \pi)$
5  if $\mathsf{V}_k\big(1^\lambda, (pk\alpha, pk\omega, pk\sigma_1, pk\varepsilon_1, cert\omega_1, task, M, C), \pi, crs\big) = 0$ then return 0
6  $(pk\sigma_2, \ldots, pk\sigma_k, cert_2, \ldots, cert_k, warr_1, \ldots, warr_{k-1}, s) := \mathsf{Dec}(sk\varepsilon_1, C)$
7  if $pk_1 \in HU$ and no queries $\mathcal{O}_{\mathsf{Del}}(pk_1, \{\cdot\cdot, task, \cdot\cdot\}, pk_2)$ then return 1
8  if $\exists\, i : pk_i \in HU$ and no queries $\mathcal{O}_{\mathsf{Del}}(pk_i, warr, \{\cdot\cdot, task, \cdot\cdot\}, pk_{i+1})$
                          with $warr[j][0][1] = pk\sigma_j$ for $1 \le j \le i$ then return 1
9  if $pk_k \in HU$ and no queries $\mathcal{O}_{\mathsf{PSig}}(pk_k, warr, task, M)$
                          with $warr[j][0][1] = pk\sigma_j$ for $1 \le j \le k$ then return 1
10 return 0
```

```
$\mathcal{O}_{\mathsf{ISndToU}}(\emptyset)$
1  $(pk\sigma, sk\sigma) \leftarrow \mathsf{K}_\sigma(1^\lambda)$
2  $HU := HU \cup \{(pk\sigma, sk\sigma)\}$
3  return $pk\sigma$
```

```
$\mathcal{O}_{\mathsf{SK}}((pk\sigma, \cdot\cdot))$
1  if $\exists\, sk\sigma : (pk\sigma, sk\sigma) \in HU$,
2  delete the entry and return $sk\sigma$
3  otherwise, return $\bot$
```

Fig. 8. Instantiated experiment for non-frameability

registered by A via $\mathsf{ISndToU}$, B sets $pk\sigma$ to its input $\overline{pk}$, hoping that A will frame this very user. If B guesses correctly and A wins the game, a forgery under $\overline{pk}$ can be extracted from the untraceable proxy signature returned by A. Let $n(\lambda)$ be the maximal number of $\mathsf{ISndToU}$ queries performed by A.

Adversary B and its handling of A's $\mathsf{ISndToU}$ and SK oracle queries are detailed in Fig. 9. To answer oracle calls Del and PSig with argument $pk^* = (\overline{pk}, \cdot\cdot)$, B replaces the line with $\mathsf{Sig}(sk\sigma, (task, pk\sigma_1, \ldots))$ in the respective algorithms by a query to its own signing oracle. For all other public keys, B holds the secret keys and can thus answer all queries.

Let S denote the event $\big[(pk\alpha, pk\omega, pk\sigma_1, pk\varepsilon_1, cert\omega_1, task, M, C) \in L_R\big]$ and E_1, E_2, E_3 denote the union of S and the event that $\mathbf{Exp}^{\text{n-frame}}$ returns 1 in line 7, 8, 9, respectively. Then the following holds:

$$\mathbf{Adv}^{\text{n-frame}}_{\mathcal{PS},A}(\lambda) \;\le\; \Pr[E_1] \;+\; \Pr[E_2] \;+\; \Pr[E_3] \;+\; \Pr[\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}(\lambda) = 1 \,\wedge\, \bar{S}] \;\;.$$

We now show that the four summands are negligible:

1. Consider the event $E_1^* := [E_1 \,\wedge\, pk\sigma_1 = \overline{pk}]$. Then, since S is satisfied, we have

   $$\mathsf{Ver}\big(\overline{pk}, (task, pk\sigma_1, pk\sigma_2), warr_1\big) = 1 \;\;.$$

   So, B returns a valid message/signature pair.
   The forgery is valid, since B did not query its oracle for $(task, pk\sigma_1, pk\sigma_2)$, as this only happens when A queries $\mathcal{O}_{\mathsf{Del}}((pk\sigma_1, \cdot\cdot), \{\cdot\cdot, task, \cdot\cdot\}, (pk\sigma_2, \cdot\cdot))$, which by E_1 is not the case. Moreover, B simulates perfectly, for E_1 implies $\mathcal{O}_{\mathsf{SK}}((\overline{pk}, \cdot\cdot)$ was not queried. All in all, we have

   $$\mathbf{Adv}^{\text{euf-cma}}_{\mathcal{DS},B} \;\ge\; \Pr[E_1^*] \;=\; \Pr[pk^* = pk_1] \cdot \Pr[E_1] \;=\; \tfrac{1}{n(\lambda)} \Pr[E_1] \;\;.$$

2. Consider the event $[E_2 \,\wedge\, pk\sigma_i = \overline{pk}]$. Then S implies

   $$\mathsf{Ver}(\overline{pk}, \big((task, pk\sigma_1, \ldots, pk\sigma_{i+1}), warr_i\big) = 1 \;\;.$$

```
**Adversary** $B(\overline{pk} : \mathsf{Sig}(sk,\cdot))$
 0  $j^* \leftarrow \{1,\dots,n\};\ j := 0$
    ⋮
 7  if $pk\sigma_1 = \overline{pk}$ and no queries $\mathcal{O}_{\mathsf{Del}}((pk\sigma_1,\cdot\cdot),\{\cdot\cdot,task,\cdot\cdot\},(pk\sigma_2,\cdot\cdot))$
            then return $\big((task,pk\sigma_1,pk\sigma_2),warr_1\big)$
 8  if $\exists\, i : pk\sigma_i = \overline{pk}$ and no queries $\mathcal{O}_{\mathsf{Del}}((pk\sigma_i,\cdot\cdot),warr,\{\cdot\cdot,task,\cdot\cdot\},(pk\sigma_{i+1},\cdot\cdot))$
        with $warr[j][0][1] = pk\sigma_j$ for $1 \le j \le i$
                    then return $\big((task,pk\sigma_1,\dots,pk\sigma_{i+1}),warr_i\big)$
 9  if $pk\sigma_k = \overline{pk}$ and no queries $\mathcal{O}_{\mathsf{PSig}}((pk\sigma_k,\cdot\cdot),warr,task,M)$ with
        $warr[j][0][1] = pk\sigma_j$ for $1 \le j \le k$, then return $\big((task,pk\sigma_1,\dots,pk\sigma_k,M),s\big)$
 10 return 0
```

```
$\mathcal{O}_{\mathsf{ISndToU}}(\emptyset)$ by $B$
 1  $j := j+1$; if $j = j^*$, return $\overline{pk}$
 2  $(pk\sigma,sk\sigma) \leftarrow \mathsf{K}_\sigma(1^\lambda)$
 3  $HU := HU \cup \{(pk\sigma,sk\sigma)\}$
 4  return $pk\sigma$
```

```
$\mathcal{O}_{\mathsf{SK}}((pk\sigma,\cdot\cdot))$ by $B$
 1  if $pk\sigma = \overline{pk}$ then abort
 2  else if $\exists\, sk\sigma : (pk\sigma,sk\sigma) \in HU$
 3      delete entry, return $sk\sigma$
 4  return $\bot$
```

Fig. 9. Adversary B against $\mathcal{DS}$.

So, B returns a valid signature on a message it did not query its signing oracle: only if A queries $\mathcal{O}_{\mathsf{Del}}((pk\sigma_i,\cdot\cdot),warr,\{\cdot\cdot,task,\cdot\cdot\},(pk\sigma_{i+1},\cdot\cdot))$ with $warr[j][0][1] = pk\sigma_j$ for $1 \le j \le i+1$, B queries $(task,pk\sigma_1,\dots,pk\sigma_{i+1})$. Moreover, B simulates perfectly, as there was no query $\mathcal{O}_{\mathsf{SK}}((\overline{pk},\cdot\cdot)$. As for 1., we have $\frac{1}{n(\lambda)}\Pr[E_2] \le \mathbf{Adv}^{\text{euf-cma}}_{\mathcal{DS},B}$.

3. Consider the event $[E_3 \wedge pk\sigma_k = \overline{pk}]$. There were no $\mathcal{O}_{\mathsf{SK}}((\overline{pk},\cdot\cdot)$ queries and S implies that B outputs a valid pair. In addition, B did not query $(task,pk\sigma_1,\dots,pk\sigma_k,M)$ (as A made no query $\mathcal{O}_{\mathsf{PSig}}((pk\sigma_k,\cdot\cdot),warr,task,M)$ with $warr[j][0][1] = pk\sigma_j$ for $1 \le j \le k$). Again, we have $\frac{1}{n(\lambda)}\Pr[E_3] \le \mathbf{Adv}^{\text{euf-cma}}_{\mathcal{DS},B}$.

4. The first clause of the event $\Pr[\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}(\lambda) = 1 \wedge \bar{S}]$ implies

$$\mathsf{V}_k(1^\lambda,(pk\alpha,pk\omega,pk\sigma_1,pk\varepsilon_1,cert\omega_1,task,M,C),\pi,crs) = 1\ \ ,$$

which together with $\bar{S}$ contradicts soundness of $\mathit{\Pi}_k$ and happens thus only with negligible probability (as in the proof of Lemma 5). □

Theorem 7. *Assuming trapdoor permutations, there exists an anonymous traceable non-frameable proxy signature scheme.*

Proof. Follows from Lemmata 4, 5 and 6. □

We have defined a new primitive unifying the concepts of group and proxy signatures and given strong security definitions for it. Moreover, Theorem 7 shows that these definitions are in fact satisfiable in the standard model, albeit by a inefficient scheme. We are nonetheless confident that more practical instantiations of our model will be proposed, as it was the case for group signatures; see e.g. [BW07] for an efficient instantiation of a variation of the model by [BMW03], or [Gro07] for an instantiation of [BSZ05]. We believe in particular that the novel methodology to construct NIZK proofs introduced by [GS08] will lead to practically usable implementations.

Acknowledgments

This work was partially funded by EADS, CELAR, the French ANR-07-SESU-008-01 PAMPA Project and the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT.

References

[BMW03] M. Bellare, D. Micciancio and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. *EUROCRYPT '03*, LNCS 2656, pp. 614–629. Springer-Verlag, 2003.

[BSZ05] M. Bellare, H. Shi and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA 2005*, LNCS 3376, pp. 136–153. Springer-Verlag, 2005.

[BDMP91] M. Blum, A. De Santis, S. Micali, and G. Persiano. Non-interactive zero-knowledge proof systems. *SIAM Journal on Computing,* 20(6):1084–1118, 1991.

[BPW03] A. Boldyreva, A. Palacio and B. Warinschi. Secure proxy signature schemes for delegation of signing rights. *IACR ePrint Archive:* Report 2003/096, 2003.

[BW07] X. Boyen and B. Waters. Full-domain subgroup hiding and constant-size group signatures. *PKC '07*, LNCS 4450, pp. 1–15. Springer-Verlag, 2007.

[CvH91] D. Chaum and E. van Heyst. Group signatures. *EUROCRYPT '91*, LNCS 547, pp. 257–265. Springer-Verlag, 1991.

[DDN00] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM Journal on Computing,* 30(2):391–437, 2000.

[FP08] G. Fuchsbauer, D. Pointcheval. Anonymous proxy signatures. *SCN '08*, LNCS 5229, pp. 201–217. Spinger-Verlag, 2008

[GMR88] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing,* 17(2):281–308, 1988.

[Gro07] J. Groth. Fully anonymous group signatures without random oracles. *ASIACRYPT '07*, LNCS 4833, pp. 164–180. Spinger-Verlag, 2007

[GS08] J. Groth, A. Sahai. Efficient non-interactive proof systems for bilinear groups. *EUROCRYPT '08*, LNCS 4965, pp. 415–432. Springer-Verlag, 2008

[MUO96] M. Mambo, K. Usuda and E. Okamoto. Proxy signatures for delegating signing operation. *Proceedings of the 3rd ACM Conference on Computer and Communications Security (CCS).* ACM, 1996.

[RS92] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. *CRYPTO '91*, LNCS 576, pp. 433–444, Springer-Verlag, 1992.

[RST01] R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Proceedings of Asiacrypt 2001,* LNCS 2248, pp. 552–565. Springer-Verlag, 2001.

[Rom90] J. Rompel. One-way functions are necessary and sufficient for secure signatures. *22nd Annual Symposium on Theory of Computing,* pp. 387–394. ACM, 1990.

[Sah99] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. *40th Symposium on Foundations of Computer Science*, pp. 543–553, IEEE, 1999.

[SK02] K. Shum and Victor K. Wei. A strong proxy signature scheme with proxy signer privacy protection. *11th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE '02),* pp. 55–56. IEEE, 2002.

[TL04] Z. Tan and Z. Liu. Provably secure delegation-by-certification proxy signature schemes. *IACR ePrint Archive:* Report 2004/148, 2004.

[TW05] M. Trolin and D. Wikström. Hierarchical group signatures. *Automata, Languages and Programming, 32nd International Colloquium (ICALP'05),* LNCS 3580, pp. 446–458. Springer-Verlag, 2005.

A Formal Definitions of the Employed Primitives

A.1 Signature Scheme $\mathcal{DS} = (\mathsf{K}_\sigma, \mathsf{Sig}, \mathsf{Ver})$

$\mathcal{DS}$ is a digital signature scheme, that is

$$\forall \lambda \in \mathbb{N}\ \forall m \in \{0,1\}^*\ \forall (pk, sk) \leftarrow \mathsf{K}_\sigma(1^\lambda) :\ \mathsf{Ver}\big(pk, m, \mathsf{Sig}(sk, m)\big) = 1$$

We assume $\mathcal{DS}$ is secure against *existential forgery under chosen-message attack*, that is

$$\forall \text{ p.p.t. } A :\ \Pr\left[\mathbf{Exp}^{\text{euf-cma}}_{\mathcal{DS},A}(\lambda) = 1\right] \ =\ \text{negl}(\lambda) \qquad \text{with}$$

```
$\mathbf{Exp}^{\text{euf-cma}}_{\mathcal{DS},A}(\lambda)$
    $(pk, sk) \leftarrow \mathsf{K}_\sigma(1^\lambda)$
    $(m, \sigma) \leftarrow A(pk : \mathsf{Sig}(sk, \cdot))$
    if $\mathsf{Ver}(pk, m, \sigma) = 1$ and $A$ never queried $m$, return 1, else return 0
```

A.2 Public-key Encryption Scheme $\mathcal{PKE} = (\mathsf{K}_\varepsilon, \mathsf{Enc}, \mathsf{Dec})$

$\mathcal{PKE}$ is a public-key encryption scheme, that is

$$\forall\, \lambda \in \mathbb{N}\ \forall\, m \in \{0,1\}^*\ \forall\, (pk, sk) \leftarrow \mathsf{K}_\varepsilon(1^\lambda) :\ \mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m$$

We assume that $\mathcal{PKE}$ satisfies *indistinguishability under adaptive chosen-ciphertext attacks*, i.e.,

$$\forall \text{ p.p.t. } A = (A_1, A_2) :\ \left|\Pr\left[\mathbf{Exp}^{\text{ind-cca-1}}_{\mathcal{PKE},A}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-cca-0}}_{\mathcal{PKE},A}(\lambda) = 1\right]\right| \ = \ \text{negl}(\lambda) \quad \text{with}$$

```
$\mathbf{Exp}^{\text{ind-cca-b}}_{\mathcal{PKE},A}(\lambda)$
    $(pk, sk) \leftarrow \mathsf{K}_\varepsilon(1^\lambda)$
    $(m_0, m_1, \text{ST}) \leftarrow A_1(pk : \mathsf{Dec}(sk, \cdot))$
    $y \leftarrow \mathsf{Enc}(pk, m_b)$
    $d \leftarrow A_2(\text{ST}, y : \mathsf{Dec}(sk, \cdot))$
    if $|m_0| = |m_1|$ and $A_2$ never queried $y$ return $d$, else return 0
```

A.3 Non-interactive Zero-knowledge Proof System $\mathit{\Pi} = (\mathsf{P}, \mathsf{V}, \mathsf{Sim})$ for L_R

We require that $\mathit{\Pi}$ satisfy the following properties:

– **Completeness**

$$\forall\, \lambda \in \mathbb{N}\ \ \forall\, (x, w) \in R \text{ with } |x| < \ell(\lambda)\ \ \forall\, r \in \{0,1\}^{p(\lambda)} :\ \mathsf{V}\big(1^\lambda, x, \mathsf{P}(1^\lambda, x, w, r), r\big) = 1$$

– **Soundness**

$$\forall \text{ p.p.t. } A :\ \Pr\left[r \leftarrow \{0,1\}^{p(\lambda)};\ (x, \pi) \leftarrow A(r) :\ x \notin L \wedge \mathsf{V}(1^\lambda, x, \pi, r) = 1\right] \ = \ \text{negl}(\lambda)$$

– **Adaptive Single-Theorem Zero Knowledge** $\quad \forall$ p.p.t. A :

$$\mathbf{Adv}^{\text{zk}}_{\mathit{\Pi},A}(\lambda) := \left|\Pr\left[\mathbf{Exp}^{\text{zk}}_{\mathit{\Pi},A}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{zk-S}}_{\mathit{\Pi},A}(\lambda) = 1\right]\right| = \text{negl}(\lambda) \quad \text{with}$$

```
$\mathbf{Exp}^{\text{zk}}_{\mathit{\Pi},A}(\lambda)$
    $r \leftarrow \{0,1\}^{p(\lambda)}$
    $(x, w, \text{ST}_A) \leftarrow A_1(r)$
    $\pi \leftarrow \mathsf{P}(x, w, r)$
    return $A_2(\text{ST}_A, \pi)$
```

```
$\mathbf{Exp}^{\text{zk-S}}_{\mathit{\Pi},A}(\lambda)$
    $(r, \text{ST}_S) \leftarrow \mathsf{Sim}_1(1^\lambda)$
    $(x, w, \text{ST}_A) \leftarrow A_1(r)$
    $\pi \leftarrow \mathsf{Sim}_2(\text{ST}_S, x)$
    return $A_2(\text{ST}_A, \pi)$
```

– **Simulation Soundness**

$$\forall \text{ p.p.t. } A :\ \Pr\left[\mathbf{Exp}^{\text{ss}}_{\mathit{\Pi},A}(\lambda) = 1\right] \ = \ \text{negl}(\lambda) \qquad \text{with}$$

```
$\mathbf{Exp}^{\text{ss}}_{\mathit{\Pi},A}(\lambda)$
    $(r, \text{ST}_S) \leftarrow \mathsf{Sim}_1(1^\lambda)$
    $(y, \text{ST}_A) \leftarrow A_1(r)$
    $\pi \leftarrow \mathsf{Sim}_2(\text{ST}_S, y)$
    $(x, \pi') \leftarrow A_2(\text{ST}_A, \pi)$
    if $\pi \neq \pi'$ and $x \notin L_R$ and $\mathsf{V}(1^\lambda, x, \pi', r) = 1$ return 1, else return 0
```