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

Georg Fuchsbauer David Pointcheval

ENS, CNRS, INRIA, Paris, France
`www.di.ens.fr/{~fuchsbau,~pointche}`

April 2008

**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 the chain of delegations 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 cannot 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* distributes signing keys to *group members*, who can then sign on behalf of the group, which can be viewed as delegating the group's signing rights to its members—there is one single *group signature verification key*. The central feature 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 of signers are set up once and for all or 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 other requirements are *traceability* (every valid signature can be traced to its signer) 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]

The main result of this paper is to unify the two above-mentioned seemingly rather different concepts, establishing a general model which encompasses proxy and group signatures. We give security notions which imply the formal ones for both primitives. Moreover, we consider consecutive delegations where *all* delegators (except the first of course) 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 allowing to define non-frameability which we extend to additionally protect against wrongful accusation of delegation.

The most prominent example of a proxy signature scheme 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 which together can by verified using the delegator's verification key only. Although not adaptable to the anonymous case—after all, the warrant

[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 $\mathit{task}'$ is considered successful (cf. Section 3.3).

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* specified in the warrant. Since our model supports re-delegation, it is conceivable that a user wishes 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 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 achieved solely 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 benefit 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.

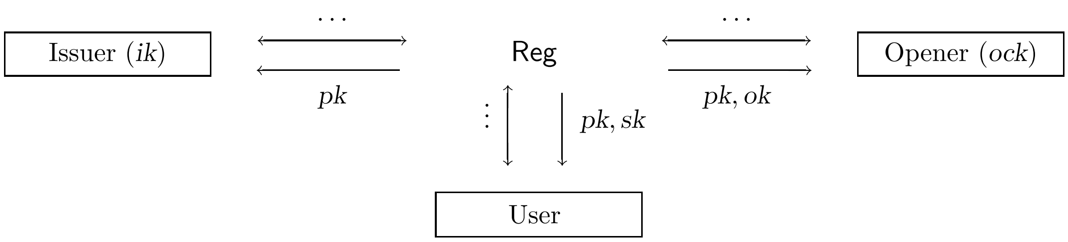
After defining the new primitive and a corresponding security model, in order to show satisfiability of the definitions, we give an instantiation and prove it secure under the (standard) assumption that families of trapdoor permutations exist. The problem of devising a more efficient construction is 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 $\mathsf{Setup}$ with the security parameter $\lambda$ 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 $\mathsf{Reg}$ with the issuer. In the end, all parties hold the user's public key *pk*, the user 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 to user $U_2$ holding $pk_2$ for a set of tasks $\mathit{TList}$ by running $\mathsf{Del}(sk_1, \mathit{TList}, pk_2)$ to produce a warrant $\mathit{warr}_{1\to 2}$ enabling $U_2$ to proxy sign on behalf of $U_1$. Now if $U_2$ wishes to re-delegate the received signing rights for a possibly reduced set of tasks $\mathit{TList}' \subseteq \mathit{TList}$ to user $U_3$ holding $pk_3$, she runs $\mathsf{Del}(sk_2, \mathit{warr}_{1\to 2}, \mathit{TList}', pk_3)$, that is, with her warrant as additional argument, to produce $\mathit{warr}_{1\to 2\to 3}$. Every user in possession of a warrant valid for a task *task* can produce proxy signatures $\sigma$ for messages $M$ corresponding to *task* by running $\mathsf{PSig}(sk, \mathit{warr}, \mathit{task}, M)$.[2] Anyone can then verify $\sigma$ under the public key $pk_1$ of the first delegator (sometimes called "original signer" in the literature) by running $\mathsf{PVer}(pk_1, \mathit{task}, M, \sigma)$.

[2]Note that it depends on the concrete application to check whether $M$ lies within the scope of *task*.




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

Figure 1: Inputs and outputs of the algorithms

Finally, using the opening key $ok_1$ corresponding to $pk_1$, a signature $\sigma$ can be opened by running $\mathsf{Open}(ok_1, task, M, \sigma)$, which returns 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. 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 with it pass verification.

**Remark 1** (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 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.

# 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. While this holds even in the presence of a corrupt issuer, the *number* of delegators involved may not remain hidden.

[3]We include *task* and $M$ in the parameters of Open to enable the opener to verify the signature before opening it.

A quite "holistic" approach to define anonymity is the following experiment in the spirit of CCA2-indistinguishability: The adversary $A$, who may control the issuer and all users, is provided with an oracle to communicate with an opening authority—who is assumed to be honest for obvious reasons. $A$ 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.[4]

Figure 2 shows 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.

$\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)$

$\quad (pp, ik, ock) \longleftarrow \mathsf{Setup}(1^\lambda)$

$\quad (\textsc{st}, pk, (sk^0, warr^0), (sk^1, warr^1), task, M)$

$\quad\quad\quad\quad \longleftarrow A_1(pp, ik : \mathsf{USndToO}, \mathsf{ISndToO}, \mathsf{OK}, \mathsf{Open})$

$\quad$ if $pk \notin OReg$, return 0

$\quad$ for $c = 0 \,..\, 1$

$\quad\quad \sigma^c \longleftarrow \mathsf{PSig}(sk^c, warr^c, task, M)$

$\quad\quad$ if $\mathsf{PVer}(pk, task, M, \sigma^c) = 0$, return 0

$\quad\quad (pk^c_2, \ldots, pk^c_{k_c}) \longleftarrow \mathsf{Open}(\mathsf{OK}(pk), task, M, \sigma)$

$\quad$ if opening succeeded and $k_0 \neq k_1$, return 0

$\quad d \longleftarrow A_2(\textsc{st}, \sigma^b : \mathsf{Open})$

$\quad$ if $A_1$ did not query $\mathsf{OK}(pk)$ and $A_2$ did not query $\mathsf{Open}(pk, task, M, \sigma^b)$, return $d$,

$\quad$ else return 0

Figure 2: Experiment for ANONYMITY

The experiment simulates an honest opener as specified by $\mathsf{Reg}$ 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 created and the corresponding public keys. $\mathsf{OK}$, called with a public key, returns the corresponding opening key from $OReg$ and when $\mathsf{Open}$ 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 2** (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

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

**Remark 3** (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 for use in hostile environments—that one could define the following variant of the scheme:

[4]Note that our definition implies all conceivable anonymity notions, such as proxy-signer anonymity, last-delegator anonymity, etc.

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 for which it is valid such that opening of the signature fails.

$\mathbf{Exp}^{\mathrm{trace}}_{\mathcal{PS},A}(\lambda)$

$\quad (pp, ik, ock) \longleftarrow \mathsf{Setup}(1^\lambda)$

$\quad (pk, task, M, \sigma) \longleftarrow A(pp : \mathsf{SndToI}, \mathsf{SndToO})$

$\quad$ if $\mathsf{PVer}(pk, task, M, \sigma) = 1$ and $\mathsf{Open}(\mathsf{OK}(pk), task, M, \sigma) = \perp$

$\qquad$ return 1, else return 0

Figure 3: Experiment for TRACEABILITY

Figure 3 shows the experiment for traceability, where the oracles $\mathsf{SndToI}$ and $\mathsf{SndToO}$ simulate issuer and opener respectively, according to the protocol $\mathsf{Reg}$. In addition, they return a transcript of the communication between them. The experiment maintains a list of generated opening keys, so $\mathsf{OK}$ 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 $\mathsf{Open}$ returns $\perp$, too.

**Definition 4** (Traceability). A proxy signature scheme $\mathcal{PS}$ is TRACEABLE if for any p.p.t. adversary $A$, we have

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

### 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, we accord the adversary as much liberty as possible in his oracle queries; unfortunately, this entails introduction of an auxiliary functionality of the

[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.

proxy signature scheme: Function $\mathsf{OpenW}$ 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 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 querying re-delegation from 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 success. Note furthermore that it is the adversary that chooses the opening key to be used. See Figure 4 for the experiment for non-frameability.

$\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}(\lambda)$

$(pp, ik, ock) \longleftarrow \mathsf{Setup}(1^\lambda)$

$(ok, pk, task, M, \sigma) \longleftarrow A(pp, ik, ock : \mathsf{ISndToU}, \mathsf{OSndToU}, \mathsf{SK}, \mathsf{Del}, \mathsf{PSig})$

if $\mathsf{PVer}(pk, task, M, \sigma) = 0$ or $\mathsf{Open}(ok, task, M, \sigma) = \bot$, 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 $\mathsf{OpenW}(warr) = (pk_1, \ldots, pk_{k-1})$ made, return 1 (Case 3)

return 0

Figure 4: Experiment for NON-FRAMEABILITY

ORACLES FOR NON-FRAMEABILITY: $\mathsf{ISndToU}$ ($\mathsf{OSndToU}$) enables the adversary impersonating a corrupt issuer (opener) to communicate with an honest user. When first called without arguments, the oracle simulates a user starting the registration procedure and makes a new entry in $HU$, the list of honest users. Oracles $\mathsf{Del}$ and $\mathsf{PSig}$ 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 $\mathsf{Del}$ with parameters $(pk_1, TList, pk_2)$ returns $\mathsf{Del}(sk_1, TList, pk_2)$. Oracle $\mathsf{SK}$ takes a public key $pk$ as argument and returns the corresponding private key after deleting $pk$ from $HU$.

**Definition 5** (Non-frameability)**.** A proxy signature scheme $\mathcal{PS}$ is NON-FRAMEABLE if for any p.p.t. adversary $A$ we have

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

**Remark 6.** In the experiment $\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},\mathcal{A}}$, the opening algorithm is run by the experiment, which by definition behaves honestly. To guard against a corrupt opener, it suffices to add a (possibly interactive) zero-knowledge proof to the system and have the opener prove correctness of decryption.

# 4 An Instantiation of the Scheme

## 4.1 Building Blocks

To construct the generic scheme $\mathcal{PS}$, we will use the following cryptographic primitives (cf. Appendix A for the formal definitions) 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 [BdSMP91, Sah99].

## 4.2 Algorithms

The algorithm $\mathsf{Setup}$ 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:

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

When a user joins the system, she creates a pair of verification/signing keys $(pk\sigma, sk\sigma)$ and *signs* $pk\sigma$ (possibly 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 he sends together with $pk\varepsilon$ to the issuer, who forwards it to the user.[6] See Figure 5.

It is by having users create their own signing keys $sk\sigma$ that a corrupt authority is prevented from framing users. 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

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

**Reg** public: $pp = (\lambda, pk\alpha, pk\omega, crs)$

**User $x$**

- $(pk\sigma, sk\sigma) \longleftarrow \mathsf{K}_\sigma(1^\lambda)$
- produce $sig$, a signature on $pk\sigma$

$\xrightarrow{pk\sigma,\, sig}$

**Issuer (holds $sk\alpha$)**

- if $sig$ invalid for $pk\sigma$, return $\perp$
- $cert \longleftarrow \mathsf{Sig}(sk\alpha, pk\sigma)$
- write $(pk\sigma, sig)$ to ***IReg***

Issuer $\xrightarrow{pk\sigma}$ Opener; Opener $\xrightarrow{pk\varepsilon,\, cert\omega}$ Issuer

**Opener for $x$ (holds $sk\omega$)**

- $(pk\varepsilon, sk\varepsilon) \longleftarrow \mathsf{K}_\varepsilon(1^\lambda)$
- $cert\omega \longleftarrow \mathsf{Sig}(sk\omega, (pk\sigma, pk\varepsilon))$
- write $(pk\sigma, pk\varepsilon, sk\varepsilon)$ to ***OReg***

$\xleftarrow{cert,\, pk\varepsilon,\, cert\omega}$

**User $x$**

- verify $cert$ and $cert\omega$

$pk := (pk\sigma, pk\varepsilon, cert, cert\omega, pp)$
$sk := (pk, sk\sigma)$

Figure 5: Registration protocol

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 $\mathsf{Del}$ 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 in $\boldsymbol{warr}_{\text{old}}$ for the tasks in $TList$. A warrant is an array where $\boldsymbol{warr}[i]$ corresponds to the $i^{\text{th}}$ delegation and $\boldsymbol{warr}[i][task]$ contains basically a signature by the $i^{\text{th}}$ delegator on the next delegator's public key and $task$.

More specifically, user $x$, being the $k^{\text{th}}$ delegator, creates $\boldsymbol{warr}$, and writes her public key to $\boldsymbol{warr}[k][0]$ that will later be used by an eventual delegator or signer. In case of re-delegation, for all $task$ to re-delegate, she copies $\boldsymbol{warr}_{\text{old}}[i][task]$ to $\boldsymbol{warr}[i][task]$ for all $i < k$. Finally, she 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**

Input: $sk_x$, $[\boldsymbol{warr}_{\text{old}}]$, $TList, pk_y$

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

Output: $\boldsymbol{warr}$

For every $k$, let $\Pi_k := (\mathsf{P}_k, \mathsf{V}_k, \mathsf{Sim}_k)$ be a simulation-sound NIZK proof system for the following NP-relation:

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

Note that for every $k$, the above relation $R_k$ defines in fact an NP-language $L_{R_k}$, since given a witness, membership of a candidate theorem is efficiently verifiable and furthermore the length of a witness is polynomial in the length of the theorem.
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 $pk_1 \to \cdots \to pk_{i+1}$,

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

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

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**

$sk, \boldsymbol{warr}, task, M \rightarrow$

$k := |\boldsymbol{warr}| + 1$, parse $sk \rightsquigarrow (pk_k, sk\sigma)$

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 $pk_i := \boldsymbol{warr}[i][0] \rightsquigarrow (pk\sigma_i, pk\varepsilon_i, cert_i, cert\omega_i, pp)$

set $warr_i := \boldsymbol{warr}[i][task]$

$s \longleftarrow \mathsf{Sig}\big(sk\sigma, (task, pk\sigma_1, \ldots, pk\sigma_k, M)\big)$; $\rho \longleftarrow \{0,1\}^{p_\varepsilon(\lambda,k)}$

$W := (pk\sigma_2, \ldots, pk\sigma_k, cert_2, \ldots, cert_k, warr_1, \ldots, warr_{k-1}, s)$

$C \longleftarrow \mathsf{Enc}(pk\varepsilon_x, W; \rho)$

$\pi \longleftarrow \mathsf{P}_k\big(1^\lambda, (pk\alpha, pk\omega, pk\sigma_1, pk\varepsilon_1, warr\omega_1, task, M, C), W \,\|\, \rho, crs\big)$

$\sigma \leftarrow$ $\sigma := (C, \pi)$

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

**PVer**

$pk_x, task, M, \sigma \rightarrow$ parse $pk_x \rightsquigarrow \big(pk\sigma_x, pk\varepsilon_x, cert_x, cert\omega_x, (\lambda, pk\alpha, pk\omega, crs)\big)$, $\sigma \rightsquigarrow (C, \pi)$

$b \leftarrow$ $b := \mathsf{V}_k\big(1^\lambda, (pk\alpha, pk\omega, pk\sigma_x, pk\varepsilon_x, cert\omega_x, task, M, C), \pi, crs\big)$

To open a signature, after checking its validity, decrypt the ciphertext contained in it:

| Open | |
|---|---|
| $ok_x, task, M, \sigma \rightarrow$ | 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 $\perp$ |
| | $(pk\sigma_2, \ldots, pk\sigma_k, cert_2, \ldots, cert_k, warr_1, \ldots, warr_{k-1}, s) := \mathsf{Dec}(sk\varepsilon_x, C)$ |
| $(pk_2, \ldots, pk_k) \leftarrow$ | if for some $i$, $pk_i$ is not in $\boldsymbol{IReg}$, return $\perp$ |

### 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 Section 3:

**Lemma 7.** *The proxy signature scheme $\mathcal{PS}$ is* ANONYMOUS *(Definition 2).*

See Appendix B.2 for the proof.

**Lemma 8.** *The proxy signature scheme $\mathcal{PS}$ is* TRACEABLE *(Definition 4).*

See Appendix B.1 for the proof.

**Lemma 9.** *The proxy signature scheme $\mathcal{PS}$ is* NON-FRAMEABLE *(Definition 5).*

*Proof.* Figure 6 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 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 registered by $A$ via ISndToU, $B$ sets $pk\sigma$ to his 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 proxy signature returned by $A$. Let $n(\lambda)$ be the maximal number of ISndToU queries $A$ makes. Figure 7 details adversary $B$ and how he answers $A$'s ISndToU and SK oracle queries.

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 his own signing oracle. For all other pulic 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:[7]

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

We now show that the four summands are negligible:

[7]If not otherwise defined, we use $\mathbf{Adv}^{\bullet}_{\bullet\bullet}(\cdot)$ as shortcut for $\Pr[\mathbf{Exp}^{\bullet}_{\bullet\bullet}(\cdot) = 1]$.

$\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}(\lambda)$

$(pk\alpha, sk\alpha) \longleftarrow \mathsf{K}_\sigma(1^\lambda);\ (pk\omega, sk\omega) \longleftarrow \mathsf{K}_\sigma(1^\lambda);\ crs \longleftarrow \{0,1\}^{p(\lambda)}$

$pp := (\lambda, pk\alpha, pk\omega, crs)$

$(ok, pk, task, M, \sigma) \longleftarrow A(pp, sk\alpha, sk\omega\ :\ \mathsf{ISndToU}, \mathsf{SK}, \mathsf{Del}, \mathsf{PSig})$

parse $ok \rightsquigarrow ((pk\sigma_1, pk\varepsilon_1, cert_1, cert\omega_1, pp), sk\varepsilon_1);\ \sigma \rightsquigarrow (C, \pi)$

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

$(pk\sigma_2, \ldots, pk\sigma_k, cert_2, \ldots, cert_k, warr_1, \ldots, warr_{k-1}, s) := \mathsf{Dec}(sk\varepsilon_1, C)$

if $pk_1 \in HU$ and no queries $\mathcal{O}_{\mathsf{Del}}(pk_1, \{\cdot\cdot, task, \cdot\cdot\}, pk_2)$ then return 1

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

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

return 0

$\mathcal{O}_{\mathsf{ISndToU}}(\emptyset)$

$(pk\sigma, sk\sigma) \longleftarrow \mathsf{K}_\sigma(1^\lambda)$

$HU := HU \cup \{(pk\sigma, sk\sigma)\}$

return $pk\sigma$

$\mathcal{O}_{\mathsf{SK}}((pk\sigma, \cdot\cdot))$

if $\exists\, sk\sigma : (pk\sigma, sk\sigma) \in HU$,

delete the entry and return $sk\sigma$

otherwise, return $\perp$

Figure 6: Instantiated experiment for non-frameability

Adversary $B(\overline{pk}\ :\ \mathsf{Sig}(sk, \cdot))$

$j^* \longleftarrow \{1, \ldots, n\};\ j := 0$

$\vdots$

if $pk\sigma_1 = \overline{pk}$ and no queries $\mathcal{O}_{\mathsf{Del}}((pk_1, \cdot\cdot), \{\cdot\cdot, task, \cdot\cdot\}, (pk\sigma_2, \cdot\cdot))$
then return $\big((task, pk\sigma_1, pk\sigma_2), warr_1\big)$

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 $\big((task, pk\sigma_1, \ldots, pk\sigma_{i+1}), warr_i\big)$

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 $\big((task, pk\sigma_1, \ldots, pk\sigma_k, M), s\big)$

return 0

$\mathcal{O}_{\mathsf{ISndToU}}(\emptyset)$ by $B$

$j := j + 1$; if $j = j^*$, return $\overline{pk}$

$(pk\sigma, sk\sigma) \longleftarrow \mathsf{K}_\sigma(1^\lambda)$

$HU := HU \cup \{(pk\sigma, sk\sigma)\}$

return $pk\sigma$

$\mathcal{O}_{\mathsf{SK}}((pk\sigma, \cdot\cdot))$ by $B$

if $pk\sigma = \overline{pk}$ then abort

else if $\exists\, sk\sigma : (pk\sigma, sk\sigma) \in HU$

delete entry, return $sk\sigma$

return $\perp$

Figure 7: Adversary $B$ against $\mathcal{DS}$ simulating $\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}$

1. Consider the event $E_1^* := [E_1 \wedge pk\sigma_1 = \overline{pk}]$. Then $\mathsf{Ver}\big(\overline{pk}, (task, pk\sigma_1, pk\sigma_2), warr_1\big) = 1$, by $S$. So, $B$ returns a valid message/signature pair. The forgery is valid, since $B$ did not query its signing 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

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

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

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

So, $B$ returns a valid signature on a message he 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 $1 \leq j \leq i+1$, $B$ queries $(task, pk\sigma_1, \ldots, pk\sigma_{i+1})$. Moreover, $B$ simulates perfectly, as there was no query $\mathcal{O}_{\mathsf{SK}}((\overline{pk}, \cdot\cdot)$. As for 1., we have $\frac{1}{n(\lambda)} \Pr[E_2] \leq \mathbf{Adv}^{\text{euf-cma}}_{\mathcal{DS},B}$.

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 by $S$, $B$ outputs a valid pair. $B$ did not query $(task, pk\sigma_1, \ldots, pk\sigma_k, M)$ (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 $1 \leq j \leq k$). Again, we have $\frac{1}{n(\lambda)} \Pr[E_3] \leq \mathbf{Adv}^{\text{euf-cma}}_{\mathcal{DS},B}$

4. The event $\Pr[\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}(\lambda) = 1]$ 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 $\Pi$: based on $\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}$, we could construct an adversary $B_{\mathrm{s}}$ against soundness of $\Pi$ which after receiving *crs* (rather than choosing it itself), runs along the lines of the experiment until line 4 and subsequently outputs $\big((pk\alpha, pk\omega, pk\sigma_1, pk\varepsilon_1, cert\omega_1, task, M, C), \pi\big)$. We have thus

$$\Pr[\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}(\lambda) = 1 \wedge \bar{S}] \;\leq\; \mathbf{Adv}^{\text{ss}}_{\Pi,B_{\mathrm{s}}} \qquad \square$$

**Theorem 10.** *Assuming trapdoor permutations, there exists an anonymous traceable non-frameable proxy signature scheme.*

*Proof.* Follows from Lemmata 7, 8 and 9. $\square$

We have thus defined a new primitive unifying the concepts of group and proxy signatures and given strong security definitions for it. Moreover, Theorem 10 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]. We believe in particular that the novel methodology to construct NIZK proofs introduced by [GS07] will lead to practically usable implementations.

## 5 Acknowledgements

This work was partially funded by EADS, CELAR and 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.

[BdSMP91] 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.

[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.

[GS07] J. Groth, A. Sahai. Efficient non-interactive proof systems for bilinear groups. *EUROCRYPT '08*, to appear; preliminary verion: *IACR ePrint Archive:* Report 2007/155

[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

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

$\mathbf{Exp}^{\text{euf-cma}}_{\mathcal{DS},A}(\lambda)$
- $(pk, sk) \longleftarrow \mathsf{K}_\sigma(1^\lambda)$
- $(m, \sigma) \longleftarrow A(pk : \mathsf{Sig}(sk, \cdot))$
- 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.,

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

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

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

We require that $\Pi$ 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

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

- Adaptive Single-Theorem Zero Knowledge

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

$\mathbf{Exp}^{\text{zk}}_{\Pi,A}(\lambda)$

$r \longleftarrow \{0,1\}^{p(\lambda)}$

$(x, w, \text{ST}_A) \longleftarrow A_1(r)$

$\pi \longleftarrow \mathsf{P}(x, w, r)$

$\texttt{return } A_2(\text{ST}_A, \pi)$

$\mathbf{Exp}^{\text{zk-S}}_{\Pi,A}(\lambda)$

$(r, \text{ST}_S) \longleftarrow \mathsf{Sim}_1(1^\lambda)$

$(x, w, \text{ST}_A) \longleftarrow A_1(r)$

$\pi \longleftarrow \mathsf{Sim}_2(\text{ST}_S, x)$

$\texttt{return } A_2(\text{ST}_A, \pi)$

- Simulation Soundness

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

$\mathbf{Exp}^{\text{ss}}_{\Pi,A}(\lambda)$

$(r, \text{ST}_S) \longleftarrow \mathsf{Sim}_1(1^\lambda)$

$(y, \text{ST}_A) \longleftarrow A_1(r)$

$\pi \longleftarrow \mathsf{Sim}_2(\text{ST}_S, y)$

$(x, \pi') \longleftarrow A_2(\text{ST}_A, \pi)$

if $\pi \neq \pi'$ and $x \notin L_R$ and $\mathsf{V}(1^\lambda, x, \pi', r) = 1$ return 1, else return 0

# B Further Proofs of Security Results

## B.1 Proof of Lemma 8

First, note that the requirement to have $pk\varepsilon$ certified by the opener prevents the adversary from trivially winning the game by using a different $pk\varepsilon'$ to encrypt which would lead to a signature that is not openable with the opener's key. Figure 8 shows the experiment $\mathbf{Exp}^{\text{trace}}_{\mathcal{PS},A}$ 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.

We construct two adversaries $B^1$, $B^2$ against existential unforgeability of $\mathcal{DS}$ that simulate $\mathbf{Exp}^{\text{trace}}_{\mathcal{PS},A}$, while using its input $\overline{pk}$ as either the opener's certifying key ($B^1$) or the issuer's signing key ($B^2$). When answering $A$'s SndToI queries, $B^1$ and $B^2$ use their oracle for the respective signature.

Adversary $B^1(\overline{pk} : \mathsf{Sig})$

$(pk\alpha, sk\alpha) \longleftarrow \mathsf{K}_\sigma(1^\lambda);\ pk\omega := \overline{pk}$

$\vdots$

if no entry $pk$ in $\boldsymbol{OReg}$, return $((pk\sigma^*, pk\varepsilon^*), cert\omega^*)$

return $\perp$

Adversary $B^2(\overline{pk} : \mathsf{Sig})$

$pk\alpha := \overline{pk};\ (pk\omega, sk\omega) \longleftarrow \mathsf{K}_\sigma(1^\lambda)$

$\vdots$

if for some $i$, $pk\sigma_i$ not in $\boldsymbol{IReg}$, return $(pk\sigma_i, cert_i)$

return $\perp$

$\mathbf{Exp}^{\text{trace}}_{\mathcal{PS},A}(\lambda)$

```
1  (pkα, skα) ⟵ K_σ(1^λ);  (pkω, skω) ⟵ K_σ(1^λ)
2  crs ⟵ {0,1}^{p(λ)};  pp := (λ, pkα, pkω, crs)
3  (pk, task, M, σ) ⟵ A(pp : SndTol)
4  parse pk ⇝ (pkσ*, pkε*, cert*, certω*, pp);  σ ⇝ (C, π)
5  if V_k(1^λ, (pkα, pkω, pkσ*, pkε*, certω*, task, M, C), π, crs) = 0, return 0
6  if no entry pk in OReg, return 1              // opening fails
            otherwise look up the corresponding skε*.
7  (pkσ_2, ..., pkσ_k, cert_2, ..., cert_k, warr_1, ..., warr_{k-1}, s) := Dec(skε*, C)
8  if for some i, pkσ_i not in IReg, return 1
9  return 0
```

$\mathcal{O}_{\mathsf{SndTol}}(pk\sigma, sig)$

```
if verification of sig on pkσ fails then return ⊥
cert ⟵ Sig(skα, pkσ); write (pkσ, sig) to IReg
(pkε, skε) ⟵ K_ε(1^λ);  certω ⟵ Sig(skω, (pkσ, pkε))
write (pkσ, pkε, skε) to OReg
return (cert, pkε, certω)
```

Figure 8: Experiment for traceability

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

$E_1 \wedge S$: We have $(pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, task, M, C) \in L_R$, so

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

On the other hand, $E_1$ implies that $(pk\sigma^*, pk\varepsilon^*)$ is not in ***OReg***, thus $B^1$ never queried it its signing oracle and returns thus a valid forgery. Consequently we have

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

$E_2 \wedge S$: Again, $(pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, task, M, C) \in L_R$, so for all $2 \leq j \leq k$ we have $\mathsf{Ver}(pk\alpha, pk\sigma_j, cert_j) = 1$, but $pk\sigma_i$ being not in ***IReg*** means $B^2$ returns a valid forgery, thus

$$\Pr[E_2 \wedge S] \;\leq\; \Pr[\mathbf{Exp}^{\text{euf-cma}}_{\mathcal{DS},B^2}(\lambda) = 1].$$

$(E_1 \vee E_2) \wedge \bar{S}$: $(E_1 \vee E_2)$ implies $\mathsf{V}_k(1^\lambda, (pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, task, M, C), \pi, crs) = 1$, so the event $(E_1 \vee E_2) \wedge \bar{S}$ contradicts soundness of $\Pi$ and happens thus only with negligible probability (cf. the proof of Lemma 9 for non-frameability).

## B.2 Proof of Lemma 7

The natural way to prove anonymity is a reduction to indistinguishability of the underlying encryption scheme: if the adversary can distinguish between two signatures $(C_1, \pi_1)$ and $(C_2, \pi_2)$, it must be by distinguishing $C_1$ from $C_2$, as the proofs $\pi_i$ do not help since they are zero-knowledge (simulating the proofs does not alter the experiments in any computationally distinguishable manner and can be performed by the adversary himself). The only case that needs special treatment in the reduction is when the $\mathcal{PS}$ adversary, after receiving $\sigma = (C, \pi)$, queries $(C, \pi')$—which is perfectly legitimate, but poses a problem to the $\mathcal{PKE}$-adversary that cannot forward $C$ to its decryption oracle.

$\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)$

$\quad crs \longleftarrow \{0,1\}^{p(\lambda)}$

$\quad (pk\alpha, sk\alpha) \longleftarrow \mathsf{K}_\sigma(1^\lambda);\ (pk\omega, sk\omega) \longleftarrow \mathsf{K}_\sigma(1^\lambda);\ pp := (\lambda, pk\alpha, pk\omega, crs)$

$\quad \big(\text{ST}, pk, (warr^0, sk^0), (warr^1, sk^1), task, M\big) \longleftarrow A_1(pp, ik : \mathsf{ISndToO}, \mathsf{OK}, \mathsf{Open})$

$\quad$ if $pk \notin OReg$, return 0, else parse $pk \rightsquigarrow (pk\sigma^*, pk\varepsilon^*, cert^*, cert\omega^*, pp)$

$\quad$ if $|warr^0| \neq |warr^1|$, return 0, else $k := |warr| + 1$

$\quad$ for $c = 0 \mathinner{..} 1$

$\qquad$ parse $sk^c \rightsquigarrow \big((pk\sigma^c_k, pk\varepsilon^c_k, cert^c_k, cert\omega^c_k, pp), sk\sigma^c\big)$

$\qquad$ for $i = 1 \mathinner{..} k-1$

$\qquad\quad pk^c_i := warr^c[i][0] \rightsquigarrow (pk\sigma^c_i, pk\varepsilon^c_i, cert^c_i, cert\omega^c_i, pp)$

$\qquad s^c \longleftarrow \mathsf{Sig}(sk\sigma^c, (task, pk\sigma^c_1, \ldots, pk\sigma^c_k, M)$

$\qquad m^c := (pk\sigma^c_2, \ldots, pk\sigma^c_k, cert^c_2, \ldots, cert^c_k,$
$\qquad\qquad warr^c[1][task], \ldots, warr^c[k-1][task], s)$

$\qquad$ if $R^*_k(pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, task, M), m^c) = 0$, return 0

$\quad \rho \longleftarrow \{0,1\}^{p_\varepsilon(\lambda)};\ C \longleftarrow \mathsf{Enc}(pk\varepsilon^*, m^b; \rho)$

$\quad \pi \longleftarrow \mathsf{P}_k\big(1^\lambda, (pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, task, M, C),\ m^b \,\|\, \rho,\ crs\big)$

$\quad d \longleftarrow A_2\big(\text{ST}, (C, \pi) : \mathsf{Open}\big)$

$\quad$ if no oracle calls $\mathsf{Open}\big(pk, task, M, (C, \pi)\big)$, return $d$, otherwise return 0

Oracle $\mathcal{O}_{\mathsf{ISndToO}}(pk\sigma)$

$\quad (pk\varepsilon, sk\varepsilon) \longleftarrow \mathsf{K}_\varepsilon(1^\lambda)$

$\quad cert\omega \longleftarrow \mathsf{Sig}\big(sk\omega, (pk\sigma, pk\varepsilon)\big)$

$\quad$ save $(pk\sigma, pk\varepsilon, cert\omega, sk\varepsilon)$ in $OReg$

$\quad$ return $(pk\varepsilon, cert\omega)$

Oracle $\mathcal{O}_{\mathsf{OK}}((pk\sigma^*, \cdot\cdot))$

$\quad$ if $(pk\sigma^*, \cdot, \cdot, sk\varepsilon) \in OReg$
$\qquad$ for some $sk\varepsilon$

$\quad$ delete the entry from $OReg$

$\quad$ return $sk\varepsilon$

Figure 9: Experiment for anonymity

Figure 9 shows the experiment for anonymity after plugging in the algorithm definitions, and some simplifications, with $R^*$ defined as $R$ restricted to the first 4 clauses, i.e., there is no check of encryption (this 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 $\mathsf{Reg}$, the $\mathsf{USndToO}$ oracle is obsolete.

We define a first variant of the original experiment by substituting the zero-knowledge proof $\pi$ by a simulated one:

$\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)^{(1)}$

1 $(crs, \textsc{st}_S) \longleftarrow \mathsf{Sim}_1(1^\lambda)$

$\vdots$

14 $\pi \longleftarrow \mathsf{Sim}_2(\textsc{st}_S, (pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert^*, task, M, C)$

$\vdots$

Since $\Pi$ is a zero-knowledge proof system, we have:

**Claim 11.**

$$\left|\Pr[\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)^{(1)} = 1]\right| \;\leq\; \mathbf{Adv}^{\text{zk}}_{\Pi,D}(\lambda)$$

*where $D$ is a p.p.t. algorithm that in the first stage, on input crs, simulates $\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)$ from line 2 to 13 and outputs $(pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert^*, task, M, C), m^b \,\|\, \rho)$; after receiving $\pi$ in the second stage, $D$ continues simulating lines 15 and 16.*

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

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

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

$\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)^{(2)}$

$\vdots$

16 if no queries $\big(pk, task, M, (C, \pi)\big)$ and no *valid* queries $\big(pk, task, M, (C, \pi')\big)$
return $d$, otherwise return 0

**Claim 12.**

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

(See below for the proof.) Due to the above claims, in order to proof Lemma 7, it suffices to relate $\Pr[\mathbf{Exp}^{\text{anon-b}(2)} = 1]$ to $\Pr[\mathbf{Exp}^{\text{ind-cca\_b}} = 1]$. 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 $j^* \longleftarrow \{1, \ldots, n\}$; $j := 0$; $(crs, \textsc{st}_S) \longleftarrow \mathsf{Sim}_1(1^\lambda)$

$\vdots$

12 if $R^*_k(pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, task, M), m^c) = 0$, return 0

13 return $(m^0, m^1, \textsc{status})$

Adversary $B_2(\textsc{status}, C : \mathsf{Dec})$

$\pi \longleftarrow \mathsf{Sim}_2(\textsc{st}_S, (pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, task, M, C)$
$d \longleftarrow A_2(\textsc{st}, (k, C, \pi) : \mathsf{Open})$
if no queries $(pk, task, M, (C, \pi))$ and no *valid* queries $(pk, task, M, (C, \pi'))$
  return $d$, otherwise return 0

Oracle $\mathcal{O}_{\mathsf{ISndToO}}(pk\sigma)$ by $B_1$

$j := j + 1$
if $j = j^*$ then $pk\varepsilon := \overline{pk}$
else $(pk\varepsilon, sk\varepsilon) \longleftarrow \mathsf{K}_\varepsilon(1^\lambda)$
$cert\omega \longleftarrow \mathsf{Sig}\big(sk\omega, (pk\sigma, pk\varepsilon)\big)$; save $(pk\sigma, pk\varepsilon, cert\omega)$ in $OReg$
return $(pk\varepsilon, cert\omega)$

When $A$ calls its $\mathsf{Open}$ oracle with a public key containing $\overline{pk}$, $B$ uses his own $\mathsf{Dec}$ oracle to decrypt the ciphertext in the signature.

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, $B$ makes an illegal $C$ query if and only if line 16 of $\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)^{(2)}$ is violated (an *invalid* query $(C, \pi')$ by $A$ does not provoke an oracle call by $B$). 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:

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

which, because of (6) and Claims 11 and 12 yields:

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

We conclude by proving the second claim:

*Proof of Claim 12.* We show that after receiving $(C, \pi)$, $A$ is very unlikely to make a *valid* open query $(C, \pi')$, i.e., create a different proof $\pi'$ 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 $\Pi$, such a query happens only with negligible probability. However, indistinguishability 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 $B^b_1(\overline{pk} : \mathsf{Dec})$

$\vdots$
return $(0^{|m^b|}, m^b, \textsc{status})$

Adversary $B_2^b(\textsc{status}, C : \mathsf{Dec})$

$\pi \longleftarrow \mathsf{Sim}_2\big(\textsc{st}_S, (pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, M, \mathit{task}, C)\big)$
$d \longleftarrow A_2(\textsc{st}, (C, \pi) : \mathsf{Open})$
if at some point $A$ queries $(C, \pi')$ with $\pi' \neq \pi$ and
$\mathsf{V}_k(1^\lambda, \big(pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, M, \mathit{task}, C), \pi', R\big) = 1$ then return 1
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 $\pi$ as input from the experiment instead of creating them itself. Now when when $A$ first makes a valid query $(C, \pi')$, it outputs $\big(X := (pk\alpha, pk\omega, pk\sigma^*, pk\varepsilon^*, cert\omega^*, M, \mathit{task}, C), \pi'\big)$, 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 $(C, \pi')$. It remains to bound the probability of event $E_b$:

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

since in the event of the second summand, $S^{b,1}$ succeeds, since $X \notin L_R$ by $pk\varepsilon^* \neq \overline{pk}$. On the other hand, we have

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

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

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

which proves the claim, for the right hand side of the equation is negligible. $\square$