A preliminary version of this paper appears in the Proceedings of FC'14. This is the full version.

Efficient and Strongly Secure Dynamic Domain-Specific Pseudonymous Signatures for ID Documents

Julien Bringer[1] Hervé Chabanne[12] Roch Lescuyer[1] Alain Patey[12]

Abstract

The notion of domain-specific pseudonymous signatures (DSPS) has recently been introduced for private authentication of ID documents, like passports, that embed a chip with computational abilities. Thanks to this privacy-friendly primitive, the document authenticates to a service provider through a reader and the resulting signatures are anonymous, linkable inside the service and unlinkable across services. A subsequent work proposes to enhance security and privacy of DSPS through group signatures techniques. In this paper, we improve on these proposals in three ways. First, we spot several imprecisions in previous formalizations. We consequently provide a clean security model for *dynamic domain-specific pseudonymous signatures*, where we correctly address the dynamic and adaptive case. Second, we note that using group signatures is somehow an overkill for constructing DSPS, and we provide an optimized construction that achieves the same strong level of security while being more efficient. Finally, we study the implementation of our protocol in a chip and show that our solution is well-suited for these limited environments. In particular, we propose a secure protocol for delegating the most demanding operations from the chip to the reader.

Keywords. ID documents, Privacy-enhancing cryptography, Domain-specific pseudonymous signatures.

[1]Morpho, 11 Bd Galliéni, 92130 Issy-Les-Moulineaux, France. {julien.bringer|herve.chabanne|roch.lescuyer|alain.patey}@morpho.com. [2]Télécom ParisTech, 46 Rue Barrault, 75013 Paris, France.

This work was done within the Identity and Security Alliance (The Morpho and Télécom ParisTech Research Center).

Contents

1 Introduction

Authentication with ID documents. Recently, the German BSI agency introduced several security mechanisms regarding the use of ID documents for authentication purposes [9]. In such situations, a *Machine Readable Travel Document* (MRTD) connects to a Service Provider (SP) through a reader (for concreteness, one might see the MRTD as a passport). The security mechanisms of [9] can be summarized as follows. First of all, during the PACE protocol (*Password Authenticated Connection Establishment*), the MRTD and the reader establish a secure channel. Then, during the EAC protocol (*Extended Access Control*), the MRTD and the SP authenticate each other through another secure channel. The reader transfers the exchanged messages. At last, during the (optional) RI protocol (*Restricted Identification*), the MRTD gives its pseudonym for the service to the SP. This pseudonym enables the SP to link users inside its service. However, across the services, users are still unlinkable. The latter property is called *cross-domain anonymity*. This property is interesting for many applications, since it offers at the same time privacy for the users and usability for the service provider, who might not want to have fully anonymous users, but might want them to use an account to give them more personal services (*e.g.* bank accounts, TV subscriptions, *etc.*).

For authentication purposes, giving pseudonyms is insufficient since the authenticity of the pseudonym is not guaranteed. For this reason, subsequent works [13, 5, 4] adopt a "signature mode" for the RI protocol. This signature mode can be described as follows.

1. The SP sends the MRTD the public key dpk of the service and a message m.
2. The MRTD computes a pseudonym nym as a deterministic function of its secret key usk and the public key dpk.
3. The MRTD signs m with its secret key usk and the pseudonym nym.
4. The MRTD sends the signature σ and the pseudonym nym to the SP.
5. The SP checks the signature σ.

The work of [5] proposes an efficient construction based on groups of prime order (without pairings). Their construction relies on a very strong hypothesis regarding the tamperproofness of the MRTD. In fact, recovering two users' secrets enables to compute the key of the certification authority. To deal with this concern, the authors of [4] propose to introduce group signatures into this signature mode. In addition to providing strong privacy properties, group signatures provide collusion resistance even if several users' secrets do leak.

Our contributions. The authors of [4] claim that the security model of group signatures directly gives a security model for DSPS, and, in fact, leave imprecise the definition of the DSPS security properties. Moreover, the model of [5] only concerns the static case, and their anonymity definition is flawed. So a security model for dynamic DSPS as such has to be supplied. Our first contribution is then a clean security model for *dynamic domain-specific pseudonymous signatures*.

This first contribution highlights the fact that, in some sense, using group signatures is "too strong" for constructing DSPS signatures. Following this intuition, we provide a new construction that is more efficient than the one of [4], while achieving the same strong security and privacy properties. Our second contribution is then an efficient, proven secure, dynamic DSPS with short signatures.

Finally, we concentrate on the use of our DSPS scheme in the RI protocol for MRTD private authentication. Our construction is based on bilinear pairings, but, as a first advantage, no pairing computation is necessary during the signature. However, we can go a step further, by taking advantage of the computational power of the reader. If some computations are delegated to the reader, then the chip only performs computation in a group of prime order. This is a valuable practical advantage since existing chips might be used. Otherwise, one needs to deploy *ad hoc* chips, which has an industrial cost.

Related notions. As a privacy-preserving cryptographic primitive, a DSPS scheme shares some properties with other primitives. We now discuss common points and differences. DSPS schemes share some similarities with *group signatures with verifier local revocation* (VLR) [8] in the sense that, in both primitives, the revocation is done on the verifier's side. However, the anonymity properties are not the same: group signatures are always unlinkable, whereas DSPS achieve some partial linkability. Moreover, one can establish a parallel with the notion of *cross-unlinkable VLR group signatures* [3], where users employ several group signatures for several domains such that the signatures are unlinkable across domains. Within a domain, the group signatures are however unlinkable, which is too strong for the context of DSPS.

The difference between DSPS and *pseudonym systems* [14] or *anonymous credential systems* [10] is that DSPS pseudonyms are deterministic whereas anonymous credentials pseudonyms must be unlinkable. In a DSPS scheme, the unlinkability is required across domains only, which is a weaker notion compared to anonymity in anonymous credentials. In fact, the anonymity of DSPS is a weaker notion compared to the anonymity of group signatures, as noticed above, and (multi-show) anonymous credentials are often constructed through group signatures techniques [10].

A point of interest is to clarify the relation between *pseudonymous signatures* and *direct anonymous attestations* (DAA) [2]. A DAA scheme might be seen (cf. [6]) as a group signature where (i) the user is split between a TPM and a host, (ii) signatures are unlinkable but in specific cases and (iii) there is no opening procedure. More precisely, the partial linkability is achieved by the notion of *basename*, a particular token present in all signature processes. Two signatures are linkable if, and only if, they are issued with the same basename.

At a first sight, a DSPS scheme is a DAA scheme where basenames are replaced by pseudonyms, and where the underlying group signature is replaced by a VLR group signature. The VLR group signatures introduce revocation concerns that are away from DAA. Moreover, in the ID document use-case, the MRTD/reader pair might be seen as the TPM/host pair of DAA scheme. However, both primitives remain distinct. The choice of pseudonyms in DSPS is more restrictive than the choice of the basename in DAA. Moreover, the host always embeds the same chip, but a MRTD is not linked to a specific reader, and might authenticate in front of several readers. Both differences impact the DSPS notion of anonymity.

Organization of the paper. In Section 2, we supply a security model for dynamic domain-specific pseudonymous signatures, and discuss in details some tricky points to formalize. Then in Section 3, we present our efficient construction of dynamic DSPS, and prove it secure in the random oracle model. Finally in Section 4, we discuss some implementation considerations and, among other things, analyse the possibility to delegate some parts of signature computation from the MRTD to the reader.

2 Definition and Security Properties of Dynamic DSPS

2.1 Algorithms and Protocols

A *dynamic domain-specific pseudonymous signature scheme* is given by an issuing authority IA, a set of users $\mathcal{U}$, a set of domains $\mathcal{D}$, and the functionalities $\{\texttt{Setup}, \texttt{DomainKeyGen}, \texttt{Join}, \texttt{Issue}, \texttt{NymGen}, \texttt{Sign}, \texttt{Verify}, \texttt{DomainRevoke}, \texttt{Revoke}\}$ as described below. By convention, users are enumerated here with indices $i \in \mathbb{N}$ and domains with indices $j \in \mathbb{N}$.

`Setup`. On input a security parameter λ, this algorithm computes global parameters gpk and an issuing secret key isk. A message space $\mathcal{M}$ is specified. The sets $\mathcal{U}$ and $\mathcal{D}$ are initially empty. The global parameters gpk are implicitly given to all algorithms, if not explicitly specified.

$$(\mathsf{gpk}, \mathsf{isk}) \leftarrow \texttt{Setup}(1^\lambda)$$

`DomainKeyGen`. On input the global parameters gpk and a domain $j \in \mathcal{D}$, this algorithm outputs a public key dpk_j for j. Together with the creation of a public key, an empty revocation list RL_j associated to this domain j is created.

$$(\mathsf{dpk}_j, RL_j) \leftarrow \texttt{DomainKeyGen}(\mathsf{gpk}, j)$$

`Join` $\leftrightarrow$ `Issue`. This protocol involves a user $i \in \mathcal{U}$ and the issuing authority IA. `Join` takes as input the global parameters gpk. `Issue` takes as input the global parameters gpk and the issuing secret key isk. At the end of the protocol, the user i gets a secret key usk_i and the issuing authority IA gets a revocation token rt_i.

$$\mathsf{usk}_i \leftarrow \texttt{Join}(\mathsf{gpk}) \leftrightarrow \texttt{Issue}(\mathsf{gpk}, \mathsf{isk}) \rightarrow \mathsf{rt}_i$$

`NymGen`. On input the global parameters gpk, a public key dpk_j for a domain $j \in \mathcal{D}$ and a secret key usk_i of a user $i \in \mathcal{U}$, this *deterministic* algorithm outputs a pseudonym nym_{ij} for the user i usable in the domain j.

$$\mathsf{nym}_{ij} \leftarrow \texttt{NymGen}(\mathsf{gpk}, \mathsf{dpk}_j, \mathsf{usk}_i)$$

`Sign`. On input the global parameters gpk, a public key dpk_j of a domain $j \in \mathcal{D}$, a user secret key usk_i of a user $i \in \mathcal{U}$, a pseudonym nym_{ij} for the user i and the domain j and a message $m \in \mathcal{M}$, this algorithm outputs a signature σ.

$$\sigma \leftarrow \texttt{Sign}(\mathsf{gpk}, \mathsf{dpk}_j, \mathsf{usk}_i, \mathsf{nym}_{ij}, m)$$

`Verify`. On input the global parameters gpk, a public key dpk_j of a domain $j \in \mathcal{D}$, a pseudonym nym_{ij}, a message $m \in \mathcal{M}$, a signature σ and the revocation list RL_j of the domain j, this algorithm outputs a decision $d \in \{\mathsf{accept}, \mathsf{reject}\}$.

$$d \leftarrow \texttt{Verify}(\mathsf{gpk}, \mathsf{dpk}_j, \mathsf{nym}_{ij}, m, \sigma, RL_j)$$

`DomainRevoke`. On input the global parameters gpk, a public key dpk_j of a domain $j \in \mathcal{D}$, an auxiliary information aux_j and the revocation list RL_j of the domain j, this algorithm outputs an updated revocation list RL'_j.

$$RL'_j \leftarrow \texttt{DomainRevoke}(\mathsf{gpk}, \mathsf{dpk}_j, \mathsf{aux}_j, RL_j)$$

`Revoke`. On input the global parameters gpk, a revocation token rt_i of a user $i \in \mathcal{U}$ and a list of domain public keys $\{\mathsf{dpk}_j\}_{j\in\mathcal{D}'\subseteq\mathcal{D}}$, this algorithm outputs a list of auxiliary information $\{\mathsf{aux}_j\}_{j\in\mathcal{D}'\subseteq\mathcal{D}}$ intended to the subset $\mathcal{D}' \subseteq \mathcal{D}$ of domains.

$$\{\mathsf{aux}_j\}_{j\in\mathcal{D}'\subseteq\mathcal{D}} \leftarrow \mathtt{Revoke}(\mathsf{gpk}, \mathsf{rt}_i, \{\mathsf{dpk}_j\}_{j\in\mathcal{D}'\subseteq\mathcal{D}})$$

We consider the dynamic case where both users and domains may be added to the system. Users might also be revoked. Moreover, the global revocation may concern all the domains at a given point, or a subset of them.

A global revocation protocol enabling to revoke the user i from every domain is implicit here: it suffices to publish rt_i. Using rt_i and public parameters, anyone can revoke user i, even for domains that will be added later.

Pseudonyms are deterministic. This implies the existence of an implicit `Link` algorithm to link signatures inside a specific domain. On input a domain public key dpk and two triples $(\mathsf{nym}, m, \sigma)$ and $(\mathsf{nym}', m', \sigma')$, this algorithm outputs 1 if $\mathsf{nym} = \mathsf{nym}'$ and outputs 0 otherwise.

The fact that pseudonyms are deterministic also gives implicit procedures for the service providers to put the users on a white list or a black list, without invoking the `Revoke` or `DomainRevoke` algorithms: it suffices to publish the pseudonym of the concerned user.

2.2 Security Definitions

To be secure, a DSPS scheme should satisfy the *correctness*, *cross-domain anonymity*, *seclusiveness* and *unforgeability* properties. Informally, a DSPS scheme is

(i) *correct* if honest and non-revoked users are accepted (signature correctness) and if the revocation of users effectively blacklists them (revocation correctness),

(ii) *cross-domain anonymous* if signatures are unlinkable but within a specific domain,

(iii) *seclusive* if it is impossible to exhibit a valid signature without involving a single existing user, and

(iv) *unforgeable* if corrupted authority and domains owners cannot sign on behalf of an honest user.

Let us now formalize each of these intuitions.

2.2.1 Correctness

A DSPS scheme is *correct* if for all large enough $\lambda \in \mathbb{N}$, all $(\mathsf{gpk}, \mathsf{isk}) :=$ DSPS.$\mathtt{Setup}(1^\lambda)$, all $\mathsf{usk} :=$ DSPS.$\mathtt{Join}(\mathsf{gpk}) \leftrightarrow$ DSPS.$\mathtt{Issue}(\mathsf{gpk}, \mathsf{isk}) \rightarrow \mathsf{rt}$, all finite polynomial $\mathcal{D} \subset \mathbb{N}$, all $(\mathsf{dpk}_n, RL_n) :=$ DSPS.$\mathtt{DomainKeyGen}(\mathsf{gpk}, n)$ for all $n \in \mathcal{D}$, all $\mathsf{nym}_n :=$ DSPS.$\mathtt{NymGen}(\mathsf{gpk}, \mathsf{dpk}_n, \mathsf{usk})$ for all $n \in \mathcal{D}$, all $\{m_n\}_{n\in\mathcal{D}}$ such that $m_n \in \mathcal{M}$ for all $n \in \mathcal{D}$, all $\sigma_n :=$ DSPS.$\mathtt{Sign}(\mathsf{gpk}, \mathsf{dpk}_n, \mathsf{usk}, \mathsf{nym}_n, m_n)$ for all $n \in \mathcal{D}$, we have:

(i) [Signature correctness] DSPS.$\mathtt{Verify}(\mathsf{gpk}, \mathsf{dpk}_n, \mathsf{nym}_n, m_n, \sigma_n, \{\}) = \mathsf{accept}$ for all $n \in \mathcal{D}$,

(ii) [Revocation correctness] for all $\mathcal{D}' \subseteq \mathcal{D}$, all $\{\mathsf{aux}_j\}_{j\in\mathcal{D}'} :=$ DSPS.$\mathtt{Revoke}(\mathsf{gpk}, \mathsf{rt}, \{\mathsf{dpk}_j\}_{j\in\mathcal{D}'})$, all $RL'_j :=$ DSPS.$\mathtt{DomainRevoke}(\mathsf{gpk}, \mathsf{dpk}_j, \mathsf{aux}_j, RL_j)$ for all $j \in \mathcal{D}'$, we have DSPS.$\mathtt{Verify}(\mathsf{gpk}, \mathsf{dpk}_j, \mathsf{nym}_j, m_j, \sigma_j, RL'_j) = \mathsf{reject}$ for all $j \in \mathcal{D}'$.

2.2.2 Oracles and Variables

We model algorithms as probabilistic polynomial Turing machines (with internal states state and decisions dec). We formalize the security properties as games between an adversary and a challenger. The adversary may have access to some oracles that are given Figure 1. Moreover, games involve the following global variables: $\mathcal{D}$ is a set of domains, $\mathcal{HU}$ of honest users, $\mathcal{CU}$ of corrupted users and $\mathcal{CH}$ of inputs to the challenge. UU is the list of "uncertainty" (see the anonymity definition below) that is: the list, for each pseudonym, of the users that might be linked to this pseudonym (in the adversary's view). **usk** records the users' secret keys, **rt** the revocation tokens, **nym** the pseudonyms, **dpk** the domain public keys, **RL** the revocation lists and $\mathbf{\Sigma}$ the signed messages.

2.2.3 Seclusiveness

Informally, a DSPS scheme achieves seclusiveness if, by similarity with the traceability property of the group signatures, an adversary A is unable to forge a valid signature that cannot "trace" to a valid user. In the group signature case, there is an opening algorithm, which enables to check if a valid user produced a given signature. However, there is no opening here, so one might ask how to define "tracing" users. Nevertheless, the management of the revocation tokens allows to correctly phrase the gain condition, as in VLR group signatures [8], providing that we take into account the presence of the pseudonyms. At the end of the game, we revoke all users on the domain supplied by the adversary. If the signature is still valid, then the adversary has won the game. Indeed, in this case, the signature does not involve any existing user. (This is an analogue of "the opener cannot conclude" in the group signature case).

$\mathsf{Seclusiveness}_A^{\mathrm{DSPS}}(\lambda)$

1. $(\mathsf{gpk}, \mathsf{isk}) \leftarrow \mathrm{DSPS}.\mathtt{Setup}(1^\lambda)$
2. $\mathcal{D}, \mathcal{HU}, \mathcal{CU} \leftarrow \{\}$
3. $\mathcal{O} \leftarrow \{\mathtt{AddDomain}(\cdot),\ \mathtt{AddUser}(\cdot),\ \mathtt{CorruptUser}(\cdot),\ \mathtt{UserSecretKey}(\cdot),\ \mathtt{Sign}(\cdot,\cdot,\cdot),\ \mathtt{ReadRegistrationTable}(\cdot),\ \mathtt{SendToIssuer}(\cdot,\cdot)\}$
4. $(\mathsf{dpk}_*, \mathsf{nym}_*, m_*, \sigma_*) \leftarrow A^{\mathcal{O}}(\mathsf{gpk})$
5. Find $j \in \mathcal{D}$ such that $\mathsf{dpk}_* := \mathbf{dpk}[j]$. If no match is found, then return 0.
6. Return 1 if, for all $i \in \mathcal{U}$, one of the following statements holds:
   (a) $\mathbf{rt}[i] = \perp$
   (b) $\mathrm{DSPS}.\mathtt{Verify}(\mathsf{gpk}, \mathsf{dpk}_*, \mathsf{nym}_*, m_*, \sigma_*, RL) = \mathsf{accept}$ where $RL := \mathrm{DSPS}.\mathtt{DomainRevoke}\big(\mathsf{gpk}, \mathsf{dpk}_*, aux, \mathbf{RL}[j]\big)$ and $aux := \mathrm{DSPS}.\mathtt{Revoke}(\mathsf{gpk}, \mathbf{rt}[i], \{\mathsf{dpk}_*\})$.

   Otherwise, return 0.

The advantage of an adversary A against the $\mathsf{Seclusiveness}$ game is defined by

$$\mathbf{Adv}^{seclusiveness}_{A,\mathrm{DSPS}}(\lambda) := \Pr[\mathsf{Seclusiveness}_A^{\mathrm{DSPS}}(\lambda) = 1].$$

A DSPS scheme achieves *seclusiveness* if, for all polynomial adversaries A, the function $\mathbf{Adv}^{seclusiveness}_{A,\mathrm{DSPS}}(\cdot)$ is negligible.

$\texttt{AddDomain}(j)$
- if $j \in \mathcal{D}$, then abort
- $\mathbf{RL}[j] := \{\}$; $\texttt{All}[j] := copy(\mathcal{HU})$
- $\mathbf{dpk}[j] \leftarrow \texttt{DomainKeyGen}(\mathsf{gpk}, j)$
- $\forall i \in \mathcal{HU}$,
  - $\mathbf{\Sigma}[(i,j)] := \{\}$; $\mathsf{UU}[(i,j)] := \&(\texttt{All}[j])$
  - $\mathbf{nym}[i][j] \leftarrow \texttt{NymGen}(\mathsf{gpk}, \mathbf{dpk}[j], \mathbf{usk}[i])$
- return $\mathbf{dpk}[j]$

$\texttt{CorruptUser}(i)$
- if $i \in \mathcal{HU} \cup \mathcal{CU}$, then abort
- $\mathcal{CU} := \mathcal{CU} \cup \{i\}$
- $\mathbf{usk}[i] := \perp$; $\mathbf{nym}[i] := \perp$; $\mathbf{rt}[i] := \perp$
- $\mathsf{dec}[IA][i] := \mathsf{cont}$; $\mathsf{state}[IA][i] := (\mathsf{gpk}, \mathsf{isk})$

$\texttt{Nym}(i, j)$
- if $i \notin \mathcal{HU}$ or $j \notin \mathcal{D}$ or $(i,j) \in \mathcal{CH}$, abort
- $\mathsf{UU}[(i,j)] := \{i\}$; $\texttt{All}[j] := \texttt{All}[j] \setminus \{i\}$
- $\forall i' \in \mathcal{HU} \setminus \{i\}$, if $\mathsf{UU}[(i',j)] \neq \&(\texttt{All}[j])$, then $\mathsf{UU}[(i',j)] := \mathsf{UU}[(i',j)] \setminus \{i\}$
- return $\mathbf{nym}[i][j]$

$\texttt{NymDomain}(j)$
- if $j \notin \mathcal{D}$, then abort
- $result := random\_perm(copy(\texttt{All}[j]))$
- $\forall i \in \mathcal{HU}$,
  - if $\mathsf{UU}[(i,j)] == \&(\texttt{All}[j])$,
    - $\mathsf{UU}[(i,j)] := copy(\texttt{All}[j])$
- $\texttt{All}[j] := \{\}$; return $\{\mathbf{nym}[i][j]\}_{i \in result}$

$\texttt{Sign}(i, j, m)$
- if $i \notin \mathcal{HU}$ or $j \notin \mathcal{D}$, then abort
- $\mathbf{\Sigma}[(i,j)] := \mathbf{\Sigma}[(i,j)] \cup \{m\}$
- return $\texttt{Sign}(\mathsf{gpk}, \mathbf{dpk}[j], \mathbf{usk}[i], \mathbf{nym}[i][j], m)$

$\texttt{ReadRegistrationTable}(i)$
- return $\mathbf{rt}[i]$

$\texttt{WriteRegistrationTable}(i, M)$
- $\mathbf{rt}[i] := M$

$\texttt{AddUser}(i)$
- if $i \in \mathcal{HU} \cup \mathcal{CU}$, then abort
- $\mathcal{HU} := \mathcal{HU} \cup \{i\}$
- run $\mathsf{usk} \leftarrow \texttt{Join}(\mathsf{gpk}) \leftrightarrow \texttt{Issue}(\mathsf{gpk}, \mathsf{isk}) \rightarrow \mathsf{rt}$
- $\mathbf{usk}[i] := \mathsf{usk}$; $\mathbf{rt}[i] := \mathsf{rt}$
- $\forall j \in \mathcal{D}$,
  - $\mathbf{\Sigma}[(i,j)] := \{\}$; $\texttt{All}[j] := \texttt{All}[j] \cup \{i\}$
  - $\mathbf{nym}[i][j] \leftarrow \texttt{NymGen}(\mathsf{gpk}, \mathbf{dpk}[j], \mathbf{usk}[i])$
  - $\mathsf{UU}[(i,j)] := \&(\texttt{All}[j])$

$\texttt{UserSecretKey}(i)$
- if $i \notin \mathcal{HU}$ or $\exists j \in \mathcal{D}$, s.t. $(i,j) \in \mathcal{CH}$, abort
- $\mathcal{HU} := \mathcal{HU} \setminus \{i\}$; $\mathcal{CU} := \mathcal{CU} \cup \{i\}$
- $\forall j \in \mathcal{D}$,
  - $\mathsf{UU}[(i,j)] := \{i\}$; $\texttt{All}[j] := \texttt{All}[j] \setminus \{i\}$
  - $\forall i' \in \mathcal{HU}$, if $\mathsf{UU}[(i',j)] \neq \&(\texttt{All}[j])$, then $\mathsf{UU}[(i',j)] := \mathsf{UU}[(i',j)] \setminus \{i\}$
- return $(\mathbf{usk}[i], \mathbf{nym}[i])$

$\texttt{Revoke}(i, \mathcal{D}')$
- $\forall j \in \mathcal{D}'$, call $\texttt{DomainRevoke}(i, j)$
- return $\{\mathbf{RL}[j]\}_{j \in \mathcal{D}'}$

$\texttt{DomainRevoke}(i, j)$
- if $i \notin \mathcal{HU}$ or $j \notin \mathcal{D}$ or $(i,j) \in \mathcal{CH}$, then abort
- $\mathsf{aux} \leftarrow \texttt{Revoke}(\mathsf{gpk}, \mathbf{rt}[i], \{\mathbf{dpk}[j]\})$
- $\mathbf{RL}[j] \leftarrow \texttt{DomainRevoke}(\mathbf{dpk}[j], \mathsf{aux}, \mathbf{RL}[j])$
- $\mathsf{UU}[(i,j)] := \{i\}$; $\texttt{All}[j] := \texttt{All}[j] \setminus \{i\}$
- $\forall i' \in \mathcal{HU} \setminus \{i\}$, if $\mathsf{UU}[(i',j)] \neq \&(\texttt{All}[j])$, then $\mathsf{UU}[(i',j)] := \mathsf{UU}[(i',j)] \setminus \{i\}$
- return $\mathbf{RL}[j]$

$\texttt{NymSign}(\mathsf{nym}, j, m)$
- if $j \notin \mathcal{D}$, then abort
- find $i \in \mathcal{HU}$ such that $\mathbf{nym}[i][j] == \mathsf{nym}$ if no match is found, then abort
- $\mathbf{\Sigma}[(i,j)] := \mathbf{\Sigma}[(i,j)] \cup \{m\}$
- return $\texttt{Sign}(\mathsf{gpk}, \mathbf{dpk}[j], \mathbf{usk}[i], \mathbf{nym}[i][j], m)$

$\texttt{SendToUser}(i, M_{in})$
- if $i \in \mathcal{CU}$, then abort ; if $i \notin \mathcal{HU}$, then $\mathcal{HU} := \mathcal{HU} \cup \{i\}$; $M_{in} := \varepsilon$; $\mathbf{usk}[i] := \perp$; $\mathsf{state}[i][IA] := \mathsf{gpk}$; $\mathsf{dec}[i][IA] := \mathsf{cont}$
- $(\mathsf{state}[i][IA], M_{out}, \mathsf{dec}[i][IA]) \leftarrow \texttt{Join}(\mathsf{state}[i][IA], M_{in}, \mathsf{dec}[i][IA])$
- if $\mathsf{dec}[i][IA] == \mathsf{accept}$, then $\mathbf{usk}[i] := \mathsf{state}[i][IA]$
- return $(M_{out}, \mathsf{dec}[i][IA])$

$\texttt{SendToIssuer}(i, M_{in})$
- if $i \notin \mathcal{CU}$, then abort
- $(\mathsf{state}[IA][i], M_{out}, \mathsf{dec}[IA][i]) \leftarrow \text{DSPS.}\texttt{Issue}(\mathsf{state}[IA][i], M_{in}, \mathsf{dec}[IA][i])$
- if $\mathsf{dec}[IA][i] == \mathsf{accept}$, then set $\mathbf{rt}[i] := \mathsf{state}[IA][i]$
- return $(M_{out}, \mathsf{dec}[IA][i])$

$\texttt{Challenge}(b_A, b_B, j_A, j_B, i_0, i_1)$
- if $i_0 \notin \mathcal{HU}$ or $i_1 \notin \mathcal{HU}$ or $i_0 == i_1$ or $j_A \notin \mathcal{D}$ or $j_B \notin \mathcal{D}$ or $j_A == j_B$, then abort
- if $\forall j \in \{j_A, j_B\}$, $\exists i \in \{i_0, i_1\}$ such that $\{i_0, i_1\} \not\subseteq \mathsf{UU}[(i,j)]$, then abort
- $\mathcal{CH} := \{(i_0, j_A), (i_0, j_B), (i_1, j_A), (i_1, j_B)\}$; return $(\mathbf{nym}[i_{b_A}][j_A], \mathbf{nym}[i_{b_B}][j_B])$

Figure 1: Oracles provided to adversaries

2.2.4 Unforgeability

Informally, we want that a corrupted authority and corrupted owners of the domains cannot sign on behalf of an honest user.

$\mathsf{Unforgeability}_A^{\mathrm{DSPS}}(\lambda)$

1. $(\mathsf{gpk}, \mathsf{isk}) \leftarrow \mathrm{DSPS.}\mathtt{Setup}(1^\lambda)$
2. $\mathcal{D}, \mathcal{HU}, \mathcal{CU} \leftarrow \{\}$
3. $\mathcal{O} \leftarrow \{\mathtt{AddDomain}(\cdot), \mathtt{WriteRegistrationTable}(\cdot,\cdot), \mathtt{Sign}(\cdot,\cdot,\cdot), \mathtt{SendToUser}(\cdot,\cdot)\}$
4. $(\mathsf{dpk}_*, \mathsf{nym}_*, m_*, \sigma_*) \leftarrow A^{\mathcal{O}}(\mathsf{gpk}, \mathsf{isk})$
5. Return 1 if all the following statements hold.
   (a) There exists $j \in \mathcal{D}$ such that $\mathsf{dpk}_* = \mathbf{dpk}[j]$
   (b) There exists $i \in \mathcal{HU}$ such that $\mathsf{nym}_* = \mathbf{nym}[i][j]$, $\mathbf{usk}[i] \neq \perp$ and $\mathbf{rt}[i] \neq \perp$
   (c) $m_* \notin \mathbf{\Sigma}[(i,j)]$
   (d) $\mathrm{DSPS.}\mathtt{Verify}(\mathsf{gpk}, \mathsf{dpk}_*, \mathsf{nym}_*, m_*, \sigma_*, \{\}) = \mathsf{accept}$
   (e) $\mathrm{DSPS.}\mathtt{Verify}(\mathsf{gpk}, \mathsf{dpk}_*, \mathsf{nym}_*, m_*, \sigma_*, L) = \mathsf{reject}$
   where $L := \mathtt{DomainRevoke}(\mathsf{gpk}, \mathsf{dpk}_*, \mathrm{DSPS.}\mathtt{Revoke}(\mathsf{gpk}, \mathbf{rt}[i], \{\mathsf{dpk}_*\}), \{\})$

   Otherwise, return 0.

The advantage of an adversary A against the Unforgeability game is defined by

$$\mathbf{Adv}^{unforgeability}_{A,\mathrm{DSPS}}(\lambda) := \Pr[\mathsf{Unforgeability}_A^{\mathrm{DSPS}}(\lambda) = 1].$$

A DSPS scheme achieves *unforgeability* if, for all polynomial adversaries A, the function $\mathbf{Adv}^{unforgeability}_{A,\mathrm{DSPS}}(\cdot)$ is negligible.

2.2.5 Cross-Domain Anonymity

Informally, a DSPS scheme achieves cross-domain anonymity if an adversary is not able to link users across domains. We formalize this intuition thanks to a *left-or-right* challenge oracle. Given two users i_0 and i_1 and two domains j_A and j_B, the challenger picks two bits $b_A, b_B \in \{0,1\}$ and returns $(\mathsf{nym}_0, \mathsf{nym}_1)$ where nym_0 is the pseudonym of i_{b_A} for the first domain and nym_1 the pseudonym of i_{b_B} for the second domain. The adversary wins if he correctly guesses the bit ($b_A == b_B$), in other words if he correctly guesses that underlying users are the same user or not. The `Challenge` oracle is called once.

$\mathsf{Anonymity}_A^{\mathrm{DSPS}}(\lambda)$

1. $(\mathsf{gpk}, \mathsf{isk}) \leftarrow \mathrm{DSPS.}\mathtt{Setup}(1^\lambda)$
2. $\mathcal{D}, \mathcal{HU}, \mathcal{CU}, \mathcal{CH} \leftarrow \{\}$
3. $b_A, b_B \xleftarrow{\$} \{0,1\}$
4. $\mathcal{O} \leftarrow \{\mathtt{AddDomain}(\cdot), \mathtt{AddUser}(\cdot), \mathtt{CorruptUser}(\cdot), \mathtt{UserSecretKey}(\cdot), \mathtt{Revoke}(\cdot,\cdot), \mathtt{DomainRevoke}(\cdot,\cdot), \mathtt{Nym}(\cdot,\cdot), \mathtt{NymDomain}(\cdot), \mathtt{NymSign}(\cdot,\cdot,\cdot), \mathtt{SendToIssuer}(\cdot,\cdot), \mathtt{Challenge}(b_A, b_B, \cdot,\cdot,\cdot,\cdot)\}$
5. $b' \leftarrow A^{\mathcal{O}}(\mathsf{gpk})$

6. Return 1 if $b' == (b_A == b_B)$, and return 0 otherwise.

The advantage of an adversary A against the Anonymity game is defined by

$$\mathbf{Adv}^{\textit{c-d-anon}}_{A,\mathrm{DSPS}}(\lambda) := \Pr[\mathsf{Anonymity}^{\mathrm{DSPS}}_{A}(\lambda) = 1].$$

A DSPS scheme achieves *cross-domain-anonymity* if, for all polynomial adversaries A, the function $\mathbf{Adv}^{\textit{c-d-anon}}_{A,\mathrm{DSPS}}(\cdot)$ is negligible.

The `SendToIssuer` oracle might be surprising here. But, contrary to group signatures, the issuing authority IA is not corrupted. This assumption is minimal since the IA may trace all honest users. Hence we must give the adversary the ability to interact as a corrupted user with the honest issuer.

Our model takes into account the case where pseudonyms leak from the network. To this aim, the `NymDomain` oracle gives the adversary a collection of pseudonyms.

Discussion about anonymity. We want to catch the intuition of being anonymous across domains, so we propose that the adversary supplies two domains of its choice, and aims at breaking anonymity across these domains. Moreover, the `Challenge` oracle, in our model, does not output two signatures, but two pseudonyms belonging to the different domains. The adversary's goal is to guess if those pseudonyms belong to the same user or not. To obtain signatures, the adversary may call a `NymSign` oracle. The adversary does not directly supply a user, but a pseudonym and obtains a signature on behalf of the underlying user. If the adversary A wants a signature from a particular user, A asks for this user's pseudonym and then asks the `NymSign` oracle for a signature.

Since the functionality is dynamic, there might be no anonymity at all if we do not take care of the formalization. For instance, an adversary might ask for adding two domains, two users, i_0, i_1, ask for their pseudonyms through two calls to `NymDomain`, add a user i_2 and win a challenge involving i_0, i_2 with non-negligible probability. This attack does not work here, since the `All` list is emptied after each `NymDomain` call.

To correctly address the cross-domain anonymity definition, we introduce a notion of "uncertainty" in the oracles. The challenger maintains, for each pseudonym, a list of the possible users the pseudonym might be linked to from the adversary's point of view. These lists evolve in function of the adversary's queries. Thus, the challenger ensures that the pseudonyms returned by the `Challenge` oracle contain enough uncertainty for at least one domain. Note that the uncertainty is required for only one domain. A user queried to the `Challenge` might be known or revoked in a domain: the adversary has to guess whether the other pseudonym belongs to the same user.

Comparison to previous security models. First, the model of [5] is static: all users and domains are created at the beginning of the games, while our security games are all dynamic. Second, let us focus on the cross-domain anonymity and show that their definition is flawed. The adversary is given all pseudonyms and all domain parameters. The left-or-right challenge takes as input two pseudonyms for the same domain and a message and outputs a signature on this message by one of the corresponding users. A simple strategy to win the game, independently of the construction, is to verify this signature using both pseudonyms: it will be valid for only one of them. This observation motivates our choice for our challenge output to be a pair of pseudonyms and not a pair of signatures, since it is easy to verify correctness using pseudonyms. Moreover, in their game, both pseudonyms queried to the challenge oracle are in the same domain, which does not fit the *cross-domain* anonymity, while our challenge

involving two different domains does. Third, the model of [5] does not allow for collusions: the adversary can be given at most one user secret key (indeed, with their construction, using two users' secret keys, one can recover the issuing keys)[1].

The model of [4] is largely inspired by the security model of VLR group signatures. That is why it does not enough take into account the specificities of DSPS. The challenge of the cross-domain anonymity game also considers a single domain and outputs a signature (but it does not take as input the pseudonyms of the users, only identifiers, so it does not inherit the security flaw of [5]). The model also lacks from a precise description of the oracles, thus leaving looseness on what are the exact inputs and outputs. Our model is more precise and separated from the model of group signatures, which leads, as we will see in the following, to a more efficient construction.

3 An Efficient Construction of Dynamic DSPS

In this section, we present an efficient construction of dynamic DSPS we call the `D` scheme and prove it secure in the sense of the previous Section in the random oracle model.

3.1 Bilinear Pairings and Hard Problems

Bilinear parings. Our construction makes use of bilinear pairings. A bilinear environment is given by a tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ where p is a prime number, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are three groups of order p (in multiplicative notation) and e is a bilinear and non-degenerate application $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The property of bilinearity states that for all $G \in \mathbb{G}_1$, $H \in \mathbb{G}_2$, $a, b \in \mathbb{Z}_p$, we have $e(G^a, H^b) = e(G, H)^{ab} = e(G^b, H^a)$. The property of non-degeneracy states that for all $G \in \mathbb{G}_1 \setminus \{\mathbf{1}_{\mathbb{G}_1}\}$, $H \in \mathbb{G}_2 \setminus \{\mathbf{1}_{\mathbb{G}_2}\}$, $e(G, H) \neq \mathbf{1}_{\mathbb{G}_T}$. Bilinear environments may be symmetric if $\mathbb{G}_1 = \mathbb{G}_2$ or asymmetric if $\mathbb{G}_1 \neq \mathbb{G}_2$.

Hard problems. We now state the problems under which we will prove our scheme secure.

Discrete Logarithm DL. Let $\mathbb{G}$ be a cyclic group of prime order p. Given $(G, H) \xleftarrow{\$} \mathbb{G}^2$, find $x \in \mathbb{N}$ such that $G^x = H$.

Decisional Diffie-Hellman DDH. Let p be a prime number, $\mathbb{G}$ be a cyclic group of order p and $a, b, c \xleftarrow{\$} \mathbb{Z}_p$. Given $\vec{G} := (G, \mathsf{A}, \mathsf{B}, \mathsf{C}) \in \mathbb{G}^4$, decide whether $\vec{G} = (G, G^a, G^b, G^{a+b})$ or $\vec{G} = (G, G^a, G^b, G^c)$.

q-Strong Diffie-Hellman q-SDH. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a bilinear environment, $H_1 \xleftarrow{\$} \mathbb{G}_1$, $H_2 \xleftarrow{\$} \mathbb{G}_2$ and $\theta \xleftarrow{\$} \mathbb{Z}_p$. Given $(H_1, {H_1}^{\theta}, {H_1}^{\theta^2}, \ldots, {H_1}^{\theta^q}, H_2, {H_2}^{\theta}) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2$, find a pair $(c, {G_1}^{1/(\theta+c)}) \in \mathbb{Z}_p \setminus \{-\theta\} \times \mathbb{G}_1$.

3.2 Description of our Solution

Let `D` be the following scheme.

`Setup`(1^λ)

1. Generate an asymmetric bilinear environment $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$
2. Pick generators $G_1, H \xleftarrow{\$} \mathbb{G}_1 \setminus \{\mathbf{1}_{\mathbb{G}_1}\}$ and $G_2 \xleftarrow{\$} \mathbb{G}_2 \setminus \{\mathbf{1}_{\mathbb{G}_2}\}$

[1] For sake of clarity, note that $(\mathsf{nym}_i, \mathsf{dsnym}_{ij})$ in [5] maps to (i, nym_{ij}) in our model.

3. Pick $\gamma \in \mathbb{Z}_p$; Set[2] $(Y_1, Y_2) := (H^\gamma, {G_2}^\gamma)$
4. Choose a hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^\lambda$
5. Return $\mathsf{gpk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, H, G_2, Y_1, Y_2, \mathcal{H})$; $\mathsf{isk} := \gamma$

$\texttt{DomainKeyGen}(\mathsf{gpk}, j)$[3]

1. Pick $r \xleftarrow{\$} \mathbb{Z}_p^*$; Set $RL_j \leftarrow \{\}$; Return $\mathsf{dpk}_j := {G_1}^r$; RL_j

$\texttt{Join}(\mathsf{gpk}) \leftrightarrow \texttt{Issue}(\mathsf{gpk}, \mathsf{isk})$

1. $[i]$ Pick $f' \xleftarrow{\$} \mathbb{Z}_p$; Set $F' := H^{f'}$
2. $[i]$ Compute $\Pi := \mathsf{PoK}\{C := \mathsf{Ext\text{-}Commit}(f') \wedge \mathsf{NIZKPEqDL}(f', C, F', H)\}$[4]
3. $[U \to IA]$ Send F, Π $\quad$ $[IA]$ Check Π
4. $[IA]$ Pick $x, f'' \in \mathbb{Z}_p$; Set $F := F' \cdot H^{f''}$; $A := \left(G_1 \cdot F\right)^{\frac{1}{\gamma + x}}$; $Z := e(A, G_2)$
5. $[U \leftarrow IA]$ Send f'', A, x, Z
6. $[i]$ Set $f := f' + f''$; Check $e(A, {G_2}^x \cdot Y_2) \stackrel{?}{=} e(G_1 \cdot H^f, G_2)$

The user gets $\mathsf{usk}_i := (f, A, x, Z)$; The issuer gets $\mathsf{rt}_i := (F, x)$

$\texttt{NymGen}(\mathsf{gpk}, \mathsf{dpk}_j, \mathsf{usk}_i)$

1. Parse usk_i as (f_i, A_i, x_i, Z_i) ; Return $\mathsf{nym}_{ij} := H^{f_i} \cdot (\mathsf{dpk}_j)^{x_i}$

$\texttt{Sign}(\mathsf{gpk}, \mathsf{dpk}, \mathsf{usk}, \mathsf{nym}, m)$

1. Parse usk as (f, A, x, Z)
2. Pick $a, r_a, r_f, r_x, r_b, r_d \xleftarrow{\$} \mathbb{Z}_p$; Set $T := A \cdot H^a$
3. Set $R_1 := H^{r_f} \cdot \mathsf{dpk}^{r_x}$; $R_2 := \mathsf{nym}^{r_a} \cdot H^{-r_d} \cdot \mathsf{dpk}^{-r_b}$
4. Set $R_3 := Z^{r_x} \cdot e(H, G_2)^{a \cdot r_x - r_f - r_b} \cdot e(H, Y_2)^{-r_a}$
5. Compute $c := \mathcal{H}(\mathsf{dpk} \| \mathsf{nym} \| T \| R_1 \| R_2 \| R_3 \| m)$
6. Set $s_f := r_f + c \cdot f$; $s_x := r_x + c \cdot x$; $s_a := r_a + c \cdot a$; $s_b := r_b + c \cdot a \cdot x$; $s_d := r_d + c \cdot a \cdot f$
7. Return $\sigma := (T, c, s_f, s_x, s_a, s_b, s_d)$

$\texttt{Verify}(\mathsf{gpk}, \mathsf{dpk}, \mathsf{nym}, m, \sigma, RL)$

1. If $\mathsf{nym} \in RL$, then return reject and abort.
2. Parse σ as $(T, c, s_f, s_x, s_a, s_b, s_d)$
3. Set $R_1' := H^{s_f} \cdot \mathsf{dpk}^{s_x} \cdot \mathsf{nym}^{-c}$; $R_2' := \mathsf{nym}^{s_a} \cdot H^{-s_d} \cdot \mathsf{dpk}^{-s_b}$

[2]The element Y_1 is only used in Section 4. If no delegation is considered, this element can be dropped.

[3]This description is independent of the concrete use-case. In particular, it is not said which entity generates the parameters and the domain keys. We assume here that these parameters (including the domains' parameters) are honestly computed. Note that r is not a private key associated to dpk. dpk is just a public random element in $\mathbb{G}_1$ that identifies a given domain.

[4]$\mathsf{Ext\text{-}Commit}$ is an extractable commitment scheme (a perfectly binding computationally hiding commitment scheme where an extraction key allows to extract the committed value). $\mathsf{NIZKPEqDL}(f, C, F, H)$ denotes a Non Interactive Zero Knowledge Proof of Equality of the Discrete Logarithm f of F in basis H with the value committed in C.

4. Set $R'_3 := e(T, G_2)^{s_x} \cdot e(H, G_2)^{-s_f - s_b} \cdot e(H, Y_2)^{-s_a} \cdot \left[e(G_1, G_2) \cdot e(T, Y_2)^{-1}\right]^{-c}$
5. Compute $c' := \mathcal{H}(\mathsf{dpk} \| \mathsf{nym} \| T \| R'_1 \| R'_2 \| R'_3 \| m)$
6. Return accept if $c = c'$, otherwise return reject.

$\mathtt{Revoke}(\mathsf{gpk}, \mathsf{rt}_i, \mathcal{D}')$

1. Parse rt_i as (F_i, x_i) ; Return $\{\mathsf{aux}_j := F_i \cdot (\mathsf{dpk}_j)^{x_i}\}_{j \in \mathcal{D}'}$

$\mathtt{DomainRevoke}(\mathsf{gpk}, \mathsf{dpk}_j, \mathsf{aux}_j, RL_j)$

1. Return $RL_j := RL_j \cup \{\mathsf{aux}_j\}$

Note about the revocation. A revocation list is a set of revoked pseudonyms. Given a (pseudonym, signature) pair, the revocation test is a simple membership test. In practice, this can be done very efficiently.

3.3 A Proof of Knowledge of a Valid Certificate

The Sign procedure is obtained by applying the Fiat-Shamir heuristic [12] to a proof of knowledge of a valid user's certificate. We explicitly give this proof of knowledge Figure 2. More precisely, the P protocol given Figure 2 allows for proving knowledge of $(f, (A, x))$ such that $A = \left(G_1 \cdot H^f\right)^{\frac{1}{\gamma + x}}$ and $\mathsf{nym} = H^f \cdot \mathsf{dpk}^x$. In the Appendices A.1 to A.3, we prove the following lemmas.

Lemma 1 (Completeness) *The* P *protocol is complete.*

Lemma 2 (Zero-Knowledge) *The* P *protocol is honest-verifier zero-knowledge.*

Lemma 3 (Proof of Knowledge) *There exists an extractor for the* P *protocol.*

3.4 Security Properties of our Solution

In the Appendices A.4 to A.7, we prove the following theorem regarding the security of our scheme.

Theorem 4 *The* D *scheme achieves seclusiveness, unforgeability and cross-domain anonymity in the sense of Section 2 in the random oracle model under the* DL*,* q-SDH *and* DDH *assumptions.*

We first show that, under a chosen-message attack, in the random oracle model, it is computationally impossible to produce a valid D signature $\sigma := (T, c, s_f, s_x, s_a, s_b, s_d)$ without the knowledge of a valid certificate (f, A, x, Z). In other words, from a valid signature, we can extract a valid certificate. This "extraction step" of a valid certificate is standard when signature schemes are built by applying the Fiat-Shamir heuristic [12] to a given Σ-protocol (cf. [16, 15, 11]).

Lemma 5 (Σ-unforgeability) *If there exists a* (t, ϵ)*-adversary* A *such that* $\epsilon \geq \frac{1}{2^\lambda} + \eta + \frac{q_S \cdot (q_H + q_S)}{p^4}$ *for some* $\eta > \frac{240 q_H}{2^\lambda}$ *after* q_H *queries to the random oracle* $\mathcal{H}$ *and* q_S *queries to the* Sign *oracle, then one can build a certificate* (f, A, x, Z) *in expecting time* $O(\frac{q_H t}{\eta})$*.*

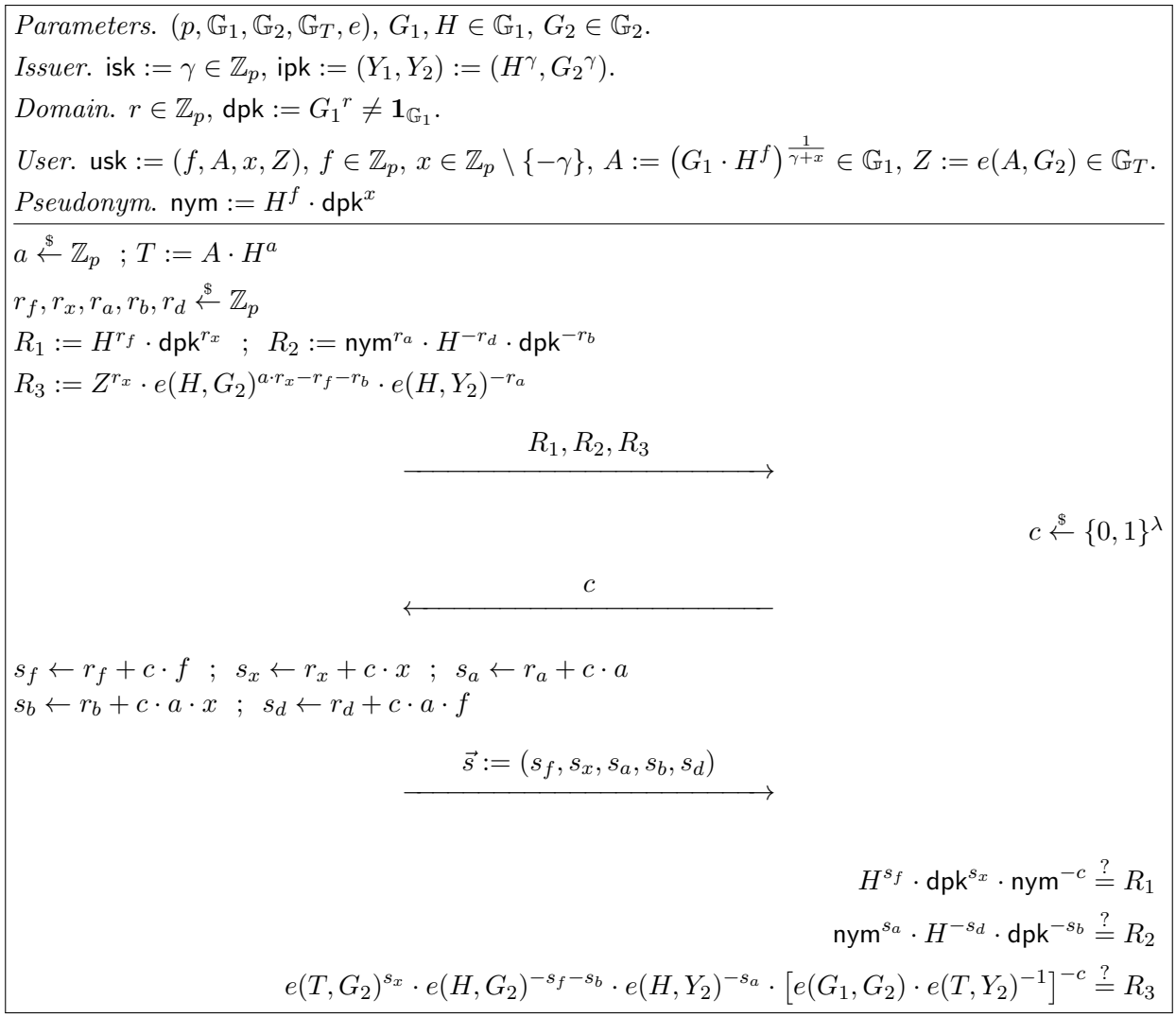

Figure 2: The P protocol

Then, we prove the following theorems.

Theorem 6 (Seclusiveness) *In the random oracle model, the* D *scheme achieves seclusiveness in the sense of Section 2 if the* SDH *problem is hard.*

More precisely, if the SDH *problem is (q, t', ϵ')-hard in $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, then the* D *scheme is (q_U, t, ϵ)-seclusive where q_U is the number of queries to the* AddUser *and* SendToIssuer *oracles as long as $t \leq t' - O(q^2)$, $q_U \leq q$ and $\epsilon \leq 2q\epsilon'$.*

Theorem 7 (Unforgeability) *In the random oracle model, the* D *scheme achieves unforgeability in the sense of the Section 2 if the* DL *problem is hard.*

More precisely, given $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, if the DL *problem is (t', ϵ')-hard in $\mathbb{G}_1$, then the* D *scheme is (t, q_U, ϵ)-unforgeable where q_U is the number of queries to the* SendToUser *oracle as long as $t \leq t' - O(q_U)$ and $\epsilon \leq q_U\epsilon'$.*

Theorem 8 (Cross-domain anonymity) *The* D *scheme achieves cross-domain anonymity in the sense of Section 2 if the* DDH *problem is hard in $\mathbb{G}_1$.*

More precisely, given bilinear environment $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, if the DDH *problem is (t', ϵ')-hard in $\mathbb{G}_1$, then the* D *scheme is $(t, q_H, q_S, q_U, q_D, q_C, \epsilon)$-cross-domain-anonymous where q_H is*

the number of queries to the random oracle, q_S to the `NymSign` *oracle, q_U to the* `AddUser` *and* `SendToIssuer` *oracles, q_D to the* `AddDomain` *oracle and q_C to the* `UserSecretKey` *oracle, as long as* $t \leq t' - (q_U + 2) \cdot T_{\text{pairing}} - (3 \cdot q_S + q_U \cdot (q_D + 1) + 1) \cdot T_{\text{exp:}\mathbb{G}_1} - q_S \cdot T_{\text{exp:}\mathbb{G}_T}$ *and*

$$\epsilon' \geq \frac{\epsilon}{q_U \cdot q_D} \cdot \left(1 - \frac{q_S \cdot (q_H + q_S)}{p^4}\right)$$

4 Implementation Considerations

4.1 Signature Size

A signature $\sigma := (T, c, s_f, s_x, s_a, s_b, s_d)$ is composed of one element in $\mathbb{G}_1$, a challenge of size λ and five scalars, which is particularly short for this level of security. By comparison, a signature of [4] is of the form $(B, J, K, T, c, s_f, s_x, s_a, s_b) \in {\mathbb{G}_1}^4 \times \{0,1\}^\lambda \times {\mathbb{Z}_p}^4$. The short group signature of [11] lies in $\in {\mathbb{G}_1}^4 \times \{0,1\}^\lambda \times {\mathbb{Z}_p}^4$ as well, which highlights the fact that we do not need the whole power of group signatures here.

4.2 Pre-Computations and Delegation of Computation

In the `D` scheme, the issuer computes the element $Z := e(A, G_2)$ and adds it to the user secret key. Thanks to this pre-computation, the user avoids to compute any pairing. In the signature procedure, the user only computes (multi)-exponentiations in $\mathbb{G}_1$ and $\mathbb{G}_T$. This is an advantage if we consider that the user is a smart-card, as in the ID document use-case.

But we can go a step further by delegating some computation from the card to the reader. The MRTD interacts with the SP through the reader but, in the RI protocol, even in signature mode, the reader just transfers the messages. In our case however, we take advantage of the computational power of the reader. A proposal[5] for this kind of delegation is given Figure 3. We obtain a piece of valuable advantages since there is no need to implement large groups operations (like operations in $\mathbb{G}_T$) in the MRTD. As a consequence, we do not need to develop specific chips for achieving those heavy computations, and existing chips can be used. We implemented our protocol on a PC. Following first estimations of a partial implementation on a chip, the overall signature and communication (including delegation) between the reader and the passport cost around 890ms, for equipment currently in use.

4.3 Security of the Delegation

Of course, this delegation of computation must be done without compromising the security. In the DAA analysis of [6], a DAA scheme (with distinct host and TPM) is built upon a pre-DAA scheme (where TPM and host are not separated). However, our analysis differs, because the MRTD is not linked to a single reader. Therefore we adapt our model. We add a pair of successive oracles (with a lock mechanism between their calls): $\texttt{GetPreComp}(i, j, m)$, enabling a corrupted reader to obtain pre-computations from an honest user, and $\texttt{Sign}'(i, j, D)$, where the same user produces a signature given a delegated computation D supplied by the adversary. Formal definitions are given Figure 4.

[5] In a preliminary version of this work, we gave a less efficient proposal, with two elements B_1, B_2 instead of a single element B here. In the proposal of Figure 3, the domain public keys must be computed with a new generator of $\mathbb{G}_1$ (if any), different from G_1 and H (cf. Section A.8).

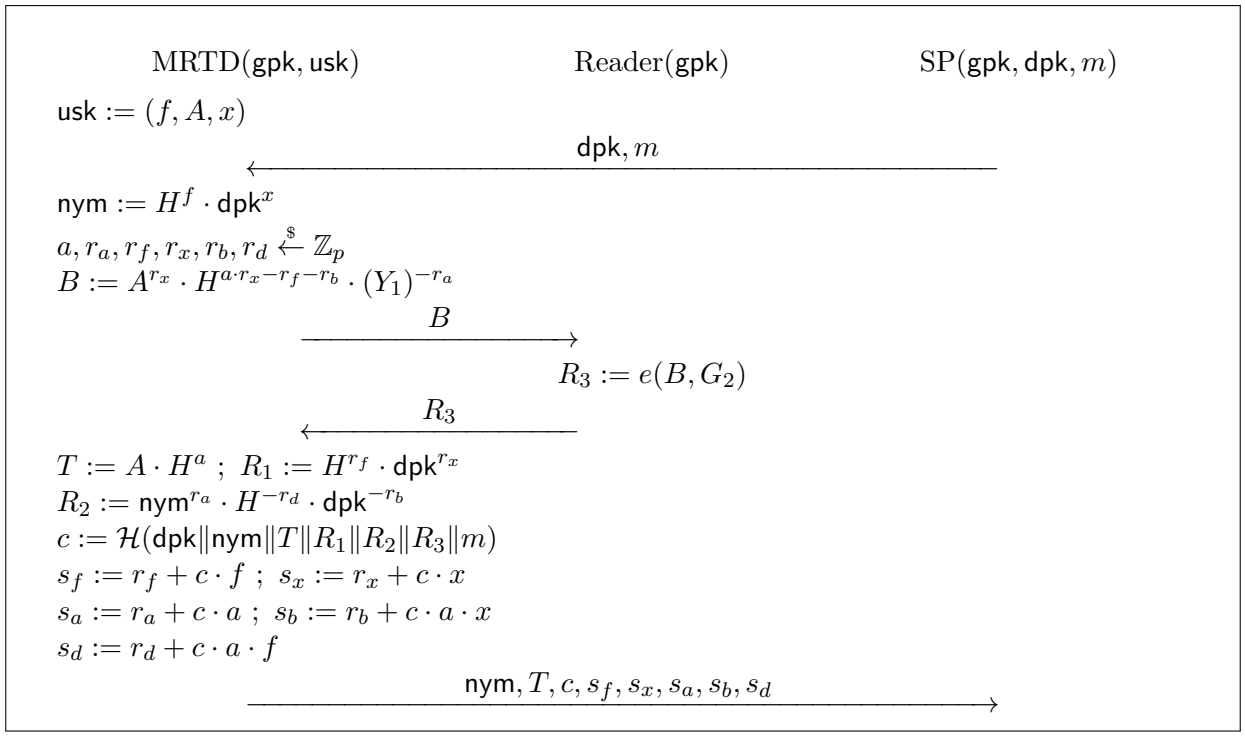

Figure 3: Delegation of computation from the MRTD to the reader

Now, in the seclusiveness game, users are corrupted and try to cheat with the issuer and the verifier. We can assume that readers are corrupted, so the adversary might call `GetPreComp` and `Sign′` to interact with honest users. In the unforgeability game, we can also assume that the reader is corrupted and add the two oracles above. Regarding the anonymity, in our use case, the reader is able to read the data on the ID document, so there is no anonymity in front of the reader (for the concerned domain/user), as there is no anonymity of the TPM from the host's point of view in a DAA scheme. However, we still want a notion of unlinkability across domains. Even if a reader is corrupted, the same user must remain anonymous in other domains, which is exactly our DSPS notion of anonymity. So the adversary might call `GetPreComp` and `Sign′`, and we restrict the `Challenge` query to involve at most one user for which the adversary called `GetPreComp` (before and after the `Challenge` call).

Finally, we adapt our proofs. In particular, we must show that the challenger can still simulate signatures in front of an active adversary impersonating a corrupted reader (the way we computed the pseudonym is important for that case). In the Appendix A.8, we show the following theorem.

Theorem 9 (Delegation security) *The* `D` *scheme with delegated computations is secure in the model with delegated computations.*

5 Conclusion

In this paper, we supplied a clean security model for *dynamic domain-specific pseudonymous signatures*, and compared this notion with other privacy-friendly cryptographic primitives. We then highlighted the fact that, in some sense, using group signatures is "too strong" for constructing DSPS signatures. Following this intuition, we provided a new construction that

Three algorithms are added in the $DSPS$ functionality.

$\texttt{PreComp}$. On input the global parameters gpk, a public key dpk_j of a domain $j \in \mathcal{D}$, a user secret key usk_i of a user $i \in \mathcal{U}$, a pseudonym nym_{ij} for the user i and the domain j and a message $m \in \mathcal{M}$, this algorithm outputs two pre-computations P_{in} and P_{out}.

$$(P_{\mathsf{in}}, P_{\mathsf{out}}) \leftarrow \texttt{PreComp}(\mathsf{gpk}, \mathsf{dpk}_j, \mathsf{usk}_i, \mathsf{nym}_{ij}, m)$$

$\texttt{DelComp}$. On input the global parameters gpk and some pre-computations P_{out}, this algorithm outputs some delegated computation D.

$$D \leftarrow \texttt{DelComp}(\mathsf{gpk}, P_{\mathsf{out}})$$

$\texttt{Sign}'$. On input the global parameters gpk, a public key dpk_j of a domain $j \in \mathcal{D}$, a user secret key usk_i of a user $i \in \mathcal{U}$, a pseudonym nym_{ij} for the user i and the domain j, some pre-computations P_{in} and some delegated computation D, this algorithm outputs a signature σ.

$$\sigma \leftarrow \texttt{Sign}'(\mathsf{gpk}, \mathsf{dpk}_j, \mathsf{usk}_i, \mathsf{nym}_{ij}, P_{\mathsf{in}}, D)$$

The *correctness* requires that for all gpk, dpk_j, usk_i, nym_{ij}, $m \in \mathcal{M}$, P such that $(P_{\mathsf{in}}, P_{\mathsf{out}}) := \texttt{PreComp}(\mathsf{gpk}, \mathsf{dpk}_j, \mathsf{usk}_i, \mathsf{nym}_{ij}, m)$, D such that $D := \texttt{DelComp}(\mathsf{gpk}, P_{\mathsf{out}})$ and σ such that $\sigma := \texttt{Sign}'(\mathsf{gpk}, \mathsf{dpk}_j, \mathsf{usk}_i, \mathsf{nym}_{ij}, P_{\mathsf{in}}, D)$, we have $\texttt{Verify}(\mathsf{gpk}, \mathsf{dpk}_j, \mathsf{nym}_{ij}, m, \sigma) = \mathsf{accept}$.

Two tables, **lock** and **precomp**, and two oracles are added in the games.

$\texttt{GetPreComp}(i, j, m)$
- if $i \notin \mathcal{HU}$ or $j \notin \mathcal{D}$ or $(i, j) \in \mathcal{CH}$, abort
- if $\mathbf{lock}[(i, j)] == \mathsf{true}$, then abort
- $\mathsf{UU}[(i, j)] := \{i\}$; $\texttt{All}[j] := \texttt{All}[j] \setminus \{i\}$
- $\forall i' \in \mathcal{HU} \setminus \{i\}$, if $\mathsf{UU}[(i', j)] \neq \&(\texttt{All}[j])$, then $\mathsf{UU}[(i', j)] := \mathsf{UU}[(i', j)] \setminus \{i\}$
- $(P_{\mathsf{in}}, P_{\mathsf{out}})$ $:= \texttt{PreComp}(\mathsf{gpk}, \mathbf{dpk}[j], \mathbf{usk}[i], \mathbf{nym}[i][j], m)$
- $\mathbf{precomp}[(i, j)] := P_{\mathsf{in}}$
- $\mathbf{lock}[(i, j)] := \mathsf{true}$
- return P_{out}

$\texttt{Sign}'(i, j, D)$
- if $\mathbf{lock}[(i, j)] == \mathsf{false}$, then abort
- $P_{\mathsf{in}} := \mathbf{precomp}[(i, j)]$
- $\sigma := \texttt{Sign}'(\mathsf{gpk}, \mathbf{dpk}[j], \mathbf{usk}[i], \mathbf{nym}[i][j], P_{\mathsf{in}}, D)$
- $\mathbf{\Sigma}[(i, j)] := \mathbf{\Sigma}[(i, j)] \cup \{m\}$
- $\mathbf{lock}[(i, j)] := \mathsf{false}$
- return σ

Figure 4: Extended model of DSPS with delegation of computation

is more efficient than the one of [4], while achieving the same strong security and privacy properties. Finally, we concentrated on the use of our DSPS scheme in the RI protocol for MRTD private authentication. Our construction might be implemented on existing chips if one takes advantage of the computational power of the reader. We supplied an analysis of such a delegation of computation.

Acknowledgements. The authors would like to thanks the anonymous reviewers of the Financial Cryptography and Data Security conference for their valuable comments and suggestions to improve the quality of this paper. The authors would also like to thanks Jacques Traoré for useful feedbacks and discussions. This work has been partially funded by the European FP7 FIDELITY project (SEC-2011-284862). The opinions expressed in this document only represent the authors' view. They reflect neither the view of the European Commission nor the view of their employer.

References

[1] Dan Boneh, Xavier Boyen, Short signatures without random oracles, the SDH assumption in bilinear groups, *J. of Crypt.*, 21(2), pp. 149-177, 2008.

[2] Ernest Brickell, Jan Camenisch, Liqun Chen, Direct anonymous attestation, *CCS'04*, pp. 132-145, ACM, 2004.

[3] Julien Bringer, Hervé Chabanne, Alain Patey, Cross-Unlinkable Hierarchical Group Signatures, *EuroPKI 2012*, LNCS 7868, pp. 161-177, Springer, 2013.

[4] Julien Bringer, Hervé Chabanne, Alain Patey, Collusion-resistant domain-specific pseudonymous signatures, *NSS'13*, LNCS 7873, pp. 649-655, Springer, 2013.

[5] Jens Bender, Özgür Dagdelen, Marc Fischlin, Dennis Kügler, Domain-specific pseudonymous signatures for the German identity card, *ISC'12*, LNCS 7483, pp. 104-119, Springer, 2012.

[6] David Bernhard, Georg Fuchsbauer, Essam Ghadafi, Nigel Smart, Bogdan Warinschi, Anonymous attestation with user-controlled linkability, *Int. J. Inf. Sec.*, 12(3), pp. 219-249, 2013.

[7] Paulo Barreto, Michael Naehrig, Pairing-friendly elliptic curves of prime order, *SAC'05*, LNCS 3897, pp. 319-331, Springer, 2006.

[8] Dan Boneh, Hovav Shacham, Group signatures with verifier-local revocation, *CCS'04*, pp. 168-177, ACM, 2004.

[9] Bundesamt für Sicherheit in der Informationstechnik (BSI), *Advanced Security Mechanisms for Machine Readable Travel Documents, Part 2 – Extended Access Control Version 2 (EACv2), Password Authenticated Connection Establishment (PACE), Restricted Identification (RI)*, TR-03110-2, March 2012.

[10] Jan Camenisch, Anna Lysyanskaya, Signature schemes, anonymous credentials from bilinear maps, *CRYPTO'04*, LNCS 3152, pp. 56-72, Springer, 2004.

[11] Cécile Delerablée, David Pointcheval, Dynamic fully anonymous short group signatures, *VIETCRYPT'06*, LNCS 4341, pp. 193-210, Springer, 2006.

[12] Amos Fiat, Adi Shamir, How to prove yourself: practical solutions to identification, signature problems, *CRYPTO'86*, LNCS 263, pp. 186-194, Springer, 1987.

[13] Miroslaw Kutylowski, Jun Shao, Signing with multiple ID's and a single key, *38th CCNC*, pp. 519-520, IEEE, 2011.

[14] Anna Lysyanskaya, Ronald Rivest, Amit Sahai, Stefan Wolf, Pseudonym systems, *SAC'99*, LNCS 1758, pp. 184-199, Springer, 2000.

[15] David Pointcheval, Jacques Stern, Security arguments for digital signatures, blind signatures, *J. of Crypt.*, 13(3), pp. 361-396, 2000.

[16] Claus-Peter Schnorr, Efficient identification, signatures for smart cards, *CRYPTO'89*, LNCS 435, pp. 239-252, Springer, 1989.

A Appendix

A.1 Proof of Lemma 1

By inspection, one can be convinced that the P protocol is complete. In particular, we have:

$$\begin{aligned}
H^{s_f} \cdot \mathsf{dpk}^{s_x} \cdot \mathsf{nym}^{-c} &= H^{r_f+c\cdot f} \cdot \mathsf{dpk}^{r_x+c\cdot x} \cdot (H^f \cdot \mathsf{dpk}^x)^{-c} = H^{r_f+c\cdot f-f\cdot c} \cdot \mathsf{dpk}^{r_x+c\cdot x-x\cdot c} \\
&= H^{r_f} \cdot \mathsf{dpk}^{r_x} = R_1 \\
\mathsf{nym}^{s_a} \cdot H^{-s_d} \cdot \mathsf{dpk}^{-s_b} &= (H^f \cdot \mathsf{dpk}^x)^{r_a+c\cdot a} \cdot H^{-r_d-c\cdot a\cdot f} \cdot \mathsf{dpk}^{-r_b-c\cdot a\cdot x} \\
&= (H^f \cdot \mathsf{dpk}^x)^{r_a} \cdot H^{f\cdot c\cdot a-r_d-c\cdot a\cdot f} \cdot \mathsf{dpk}^{x\cdot c\cdot a-r_b-c\cdot a\cdot x} \\
&= (H^f \cdot \mathsf{dpk}^x)^{r_a} \cdot H^{-r_d} \cdot \mathsf{dpk}^{-r_b} = R_2 \\
e(T,G_2)^{s_x} \cdot e(H,&G_2)^{-s_f-s_b} \cdot e(H,Y_2)^{-s_a} \cdot \left[e(G_1,G_2)\cdot e(T,Y_2)^{-1}\right]^{-c} \\
&= e(T^{r_x} \cdot H^{-r_f-r_b}, G_2) \cdot e((A\cdot H^a)^{c\cdot x} \cdot H^{-c\cdot f-c\cdot a\cdot x} \cdot {G_1}^{-c}, G_2) \\
&\qquad \cdot e(H^{-r_a}, Y_2) \cdot e(H^{-c\cdot a} \cdot (A\cdot H^a)^c, Y_2) \\
&= e(T^{r_x} \cdot H^{-r_f-r_b}, G_2) \cdot e(H^{-r_a}, Y_2) \cdot e(A^x \cdot H^{-f} \cdot {G_1}^{-1}, G_2)^c \cdot e(A, Y_2)^c \\
&= R_3 \cdot e(\left(G_1 \cdot H^f\right)^{\frac{1}{x+\gamma}}, {G_2}^x \cdot {G_2}^\gamma)^c \cdot e(G_1 \cdot H^f, G_2)^{-c} = R_3
\end{aligned}$$

A.2 Proof of Lemma 2

For an honest verifier, the transcripts T, (R_1, R_2, R_3), c, $(s_f, s_x, s_a, s_b, s_d)$ can be simulated in an indistinguishable way, without knowing any valid certificate.

We first simulate T, which can be done by picking $T \xleftarrow{\$} \mathbb{G}_1$. This element is indistinguishable from the output of any prover since, given $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, $G_1, H \xleftarrow{\$} \mathbb{G}_1 \setminus \{\mathbf{1}\}$, $G_2 \xleftarrow{\$} \mathbb{G}_2 \setminus \{\mathbf{1}\}$, $\gamma, f, r \xleftarrow{\$} \mathbb{Z}_p$, $x \xleftarrow{\$} \mathbb{Z}_p \setminus \{-\gamma\}$, $A := \left(G_1 \cdot H^f\right)^{\frac{1}{\gamma+x}}$, $\mathsf{dpk} := {G_1}^r$ and $\mathsf{nym} := H^f \cdot \mathsf{dpk}^x$, the following distributions Δ and Δ' are the same.

$$\Delta := \left\{ T \;\middle|\; a \xleftarrow{\$} \mathbb{Z}_p \;;\; T := A \cdot H^a \right\} \text{ and } \Delta' := \left\{ T \;\middle|\; T \xleftarrow{\$} \mathbb{G}_1 \right\}$$

Then, we pick $c \xleftarrow{\$} \{0,1\}^\lambda$ and $s_f, s_x, s_a, s_b, s_d \xleftarrow{\$} \mathbb{Z}_p$. We simply compute (R_1, R_2, R_3) using the verification equations: $R_1 := H^{s_f} \cdot \mathsf{dpk}^{s_x} \cdot \mathsf{nym}^{-c}$, $R_2 := \mathsf{nym}^{s_a} \cdot H^{-s_d} \cdot \mathsf{dpk}^{-s_b}$, $R_3 := e(T^{s_x} \cdot H^{-s_f-s_b}, G_2) \cdot e(H^{-s_a}, Y_2) \cdot \left[e(G_1,G_2)\cdot e(T,Y_2)^{-1}\right]^{-c}$. This second step does not assume any knowledge about the data used to generate usk, dpk, nym and T in the first step. So the simulation of the second step is perfect. □

A.3 Proof of Lemma 3

Let us assume that a prover is able to give two valid responses $\vec{s}, \vec{s'}$ to two different challenges c, c' given the same values $T, (R_1, R_2, R_3)$. First, by dividing

$$H^{s_f} \cdot \mathsf{dpk}^{s_x} = R_1 \cdot \mathsf{nym}^c \qquad \text{by} \qquad H^{s'_f} \cdot \mathsf{dpk}^{s'_x} = R_1 \cdot \mathsf{nym}^{c'},$$

we obtain $H^{s_f-s'_f} \cdot \mathsf{dpk}^{s_x-s'_x} = \mathsf{nym}^{c-c'}$. Since $c \neq c'$, then $c - c' \neq 0 \mod p$, so $c-c'$ is invertible modulo p. Hence $\tilde{f} := (s_f - s'_f)/(c-c')$ and $\tilde{x} := (s_x - s'_x)/(c-c')$ such that $\mathsf{nym} = H^{\tilde{f}} \cdot \mathsf{dpk}^{\tilde{x}}$. Then by dividing

$$\mathsf{nym}^{s_a} \cdot H^{-s_d} \cdot \mathsf{dpk}^{-s_b} = R_2 \qquad \text{by} \qquad \mathsf{nym}^{s'_a} \cdot H^{-s'_d} \cdot \mathsf{dpk}^{-s'_b} = R_2,$$

we obtain $\mathsf{nym}^{s_a - s'_a} = H^{s_d - s'_d} \cdot \mathsf{dpk}^{s_b - s'_b}$. Substituting $\mathsf{nym} = H^{\tilde{f}} \cdot \mathsf{dpk}^{\tilde{x}}$ gives that $(s_d - s'_d) = \tilde{f} \cdot (s_a - s'_a)$ and $(s_b - s'_b) = \tilde{x} \cdot (s_a - s'_a)$ (which holds if H and dpk are generators of $\mathbb{G}_1$). Finally by dividing

$$e(T, G_2)^{s_x} \cdot e(H, G_2)^{-s_f - s_b} \cdot e(H, Y_2)^{-s_a} = R_3 \cdot \left[e(G_1, G_2) \cdot e(T, Y_2)^{-1}\right]^c \qquad \text{by}$$
$$e(T, G_2)^{s'_x} \cdot e(H, G_2)^{-s'_f - s'_b} \cdot e(H, Y_2)^{-s'_a} = R_3 \cdot \left[e(G_1, G_2) \cdot e(T, Y_2)^{-1}\right]^{c'},$$

we obtain: $e(T, G_2)^{s_x - s'_x} \cdot e(H, G_2)^{-s_f + s'_f - s_b + s'_b} \cdot e(H, Y_2)^{-s_a + s'_a} = \left[e(G_1, G_2) \cdot e(T, Y_2)^{-1}\right]^{c - c'}$. If we set $\tilde{a} := (s_a - s'_a)/(c - c')$ and use the previously obtained equality $(s_b - s'_b) = \tilde{x} \cdot (s_a - s'_a)$, then we have that

$$e(T, {G_2}^{\tilde{x}}) \cdot e(H^{-\tilde{f} - \tilde{a} \cdot \tilde{x}}, G_2) \cdot e(H^{-\tilde{a}}, Y_2) = e(G_1, G_2) \cdot e(T, Y_2)^{-1}.$$

Thus $e(T \cdot H^{-\tilde{a}}, {G_2}^{\tilde{x}} \cdot Y_2) = e(G_1 \cdot H^{\tilde{f}}, G_2)$. By setting $\tilde{A} := T \cdot H^{-\tilde{a}}$, $(\tilde{f}, (\tilde{A}, \tilde{x}))$ is a valid witness. □

A.4 Proof of Lemma 5

A signature scheme based on a Σ-protocol is a signature scheme $\{\mathtt{KeyGen}, \mathtt{Sign}, \mathtt{Verify}\}$ such that each signature σ is of the form $(m, \vec{R}, c, \vec{s})$ where m is the message (and more generally the statement to be proven, that may contain information in addition to the message), $\vec{R}$ is a commitment vector, $c := \mathcal{H}(\vec{R} \| m)$ is a hash belonging to $\{0,1\}^\lambda$ for a security parameter λ and $\vec{s}$ is a response vector. We now recall some lemmas from [15, 11].

Lemma 10 (Forking [11]) *Let A be a probabilistic polynomial time Turing machine whose input is pk and which can ask q_H queries to the random oracle, with $q_H > 0$ (this is then a no message attack against a generic signature scheme based on a Σ-protocol in the random oracle model). Let us assume that, within time t, A produces a valid signature $(m, \vec{R}, c, \vec{s})$, with probability $\epsilon \geq \frac{1}{2^\lambda} + \eta$ for some $\eta > \frac{240 q_H}{2^\lambda}$. Then, within time $t' \leq \frac{9 q_H t}{\epsilon}$ and with probability $\epsilon' \geq \frac{1}{6}$, a replay of this machine outputs two valid signatures $(m, \vec{R}, c, \vec{s})$ and $(m, \vec{R}, c', \vec{s'})$ such that $c \neq c'$.*

(Cf. [11] for more details): with probability greater than η, A outputs a valid signature $(m, \vec{R}, c, \vec{s})$ such that c has been obtained as an $\mathcal{H}$ answer on $\vec{R} \| m$. So $\frac{2}{\eta}$ replays of A with different random tapes provide a success such that the query $\mathcal{H}(\vec{R} \| m)$ has been asked and answered by c (the crucial query) with probability greater than $1 - e^{-2} \geq \frac{6}{7}$. By applying Lemma 11, we know that with probability of $\frac{1}{4}$, for each replay, we have a new success with probability greater than $\frac{\eta}{4 q_H}$: we thus replay the attack $\frac{8 q_H}{\eta}$ times with a new random oracle (but the same answers until the crucial query). With probability greater than $\frac{6}{7}$, we get another success. The challenge is different from the previous one with probability $\frac{8 q_H}{\eta 2^\lambda}$. Finally, after less than $\frac{2(1 + 4 q_H)}{\eta}$ replays of the attack, with probability greater than $\frac{1}{5} - \frac{8 q_H}{\eta 2^\lambda}$ which is greater than $\frac{1}{6}$ as soon as $\eta \geq \frac{240 q_H}{2^\lambda}$, we get two valid signatures $(m, \vec{R}, c, \vec{s})$ and $(m, \vec{R}, c', \vec{s'})$ with $c' \neq c$.

Lemma 11 (Splitting [15]) *Let X, Y be sets and $A \subseteq X \times Y$ such that $\Pr[(x, y) \in A] \geq \epsilon$. For any $\alpha < \epsilon$, define $B := \{(x, y) \in X \times Y | \Pr_{y' \in Y}(x, y') \in A \geq \epsilon - \alpha\}$ and $\bar{B} = (X \times Y) \setminus B$. Then the following statements hold: (i) $\Pr[B] \geq \alpha$, (ii) for all $(x, y) \in B$, $\Pr_{y' \in Y}[(x, y') \in A] \geq \epsilon - \alpha$, and (iii) $\Pr[B|A] \geq \alpha/\epsilon$.*

Lemma 12 (ΣUnforgeability) *Under a chosen-message attack in the random oracle model, it is computationally impossible to produce a valid* $\mathtt{D}$ *signature* $\sigma := (T, c, s_f, s_x, s_a, s_b, s_d)$ *without the knowledge of a valid certificate* (f, A, x, Z).

More precisely, if there exists a (t, ϵ)*-adversary* A *against the* $\Sigma\mathsf{Unforgeability}$ *game below such that* $\epsilon \geq \frac{1}{2^\lambda} + \eta + \frac{q_S \cdot (q_H + q_S)}{p^4}$ *for some* $\eta > \frac{240 q_H}{2^\lambda}$ *after* q_H *queries to the random oracle* $\mathcal{H}$ *and* q_S *queries to the* $\mathtt{Sign}$ *oracle, then one can build a certificate* (f, A, x, Z) *in expecting time* $O(\frac{q_H t}{\eta})$.

Let A be an adversary against the following $\Sigma\mathsf{Unforgeability}$ game under *chosen message* attack in the random oracle model, and B be the challenger. We introduce two games, depending on whether B gives A the secret issuing key or not.

$\Sigma\mathsf{Unforgeability}^{\mathtt{D}}_A(\lambda)$

1. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a bilinear environment ; $\quad G_1, H \xleftarrow{\$} \mathbb{G}_1$
2. *Game 1:* $Y_1 \xleftarrow{\$} \mathbb{G}_1$; $G_2, Y_2 \xleftarrow{\$} \mathbb{G}_2$
2. *Game 2:* $G_2 \xleftarrow{\$} \mathbb{G}_2$; $\gamma \xleftarrow{\$} \mathbb{Z}_p$; $Y_1 := {G_1}^\gamma$; $Y_2 := {G_2}^\gamma$
3. $\mathsf{gpk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, H, G_2, Y_1, Y_2)$
4. *Game 1:* $(\mathsf{dpk}_*, \mathsf{nym}_*, m_*, \sigma_*) \leftarrow A^{\mathcal{H}(\cdot), \mathtt{Sign}(\cdot,\cdot,\cdot)}(\mathsf{gpk})$
4. *Game 2:* $(\mathsf{dpk}_*, \mathsf{nym}_*, m_*, \sigma_*) \leftarrow A^{\mathcal{H}(\cdot), \mathtt{Sign}(\cdot,\cdot,\cdot)}(\mathsf{gpk}, \gamma)$
5. Return 1 if
   (a) $\sigma_* \notin \mathsf{S}[\mathsf{dpk}_* \| \mathsf{nym}_* \| m_*]$ and
   (b) $\mathtt{D.Verify}(\mathsf{gpk}, \mathsf{dpk}_*, \mathsf{nym}_*, m_*, \sigma_*, \{\}) = \mathsf{accept}$

   return 0 otherwise.

We show that B can, in both games, simulate signatures *without knowledge of the secret key.* B maintains two tables S for the signed messages and H for the hash queries, both initially empty.

$\mathcal{H}(m)$. On input $m \in \{0,1\}^*$, if $\mathsf{H}[m] \neq \perp$, then B returns $\mathsf{H}[m]$, otherwise B picks $c \xleftarrow{\$} \{0,1\}^\lambda$, sets $\mathsf{H}[m] := c$ and returns c.

$\mathtt{Sign}(\mathsf{dpk}, \mathsf{nym}, m)$. On input $\mathsf{dpk}, \mathsf{nym} \in \mathbb{G}_1$, $m \in \{0,1\}^*$, B picks $T \xleftarrow{\$} \mathbb{G}_1$, $c \xleftarrow{\$} \{0,1\}^\lambda$ and $s_f, s_x, s_a, s_b, s_d \xleftarrow{\$} \mathbb{Z}_p$, sets $R_1 := H^{s_f} \cdot \mathsf{dpk}^{s_x} \cdot \mathsf{nym}^{-c}$, $R_2 := \mathsf{nym}^{s_a} \cdot H^{-s_d} \cdot \mathsf{dpk}^{-s_b}$, $R_3 := e(T, G_2)^{s_x} \cdot e(H, G_2)^{-s_f - s_b} \cdot e(H, Y_2)^{-s_a} \cdot \left[e(G_1, G_2) \cdot e(T, Y_2)^{-1}\right]^{-c}$.

B sets $\mathsf{m} := \mathsf{dpk} \| \mathsf{nym} \| T \| R_1 \| R_2 \| R_3 \| m$.

If $\mathsf{H}[\mathsf{m}] \neq \perp$ and $c \neq \mathsf{H}[\mathsf{m}]$, then B returns $\perp$ and aborts. Otherwise, B sets $\mathsf{H}[\mathsf{m}] := c$, $\sigma := (T, c, s_f, s_x, s_a, s_b, s_d)$, $\mathsf{S}[\mathsf{dpk} \| \mathsf{nym} \| m] := \mathsf{S}[\mathsf{dpk} \| \mathsf{nym} \| m] \cup \{\sigma\}$ and returns σ.

Therefore B can simulate the $\mathtt{Sign}$ oracle in the name of any pseudonym, with a probability of failure when setting a random oracle value. This probability is less than $\frac{(q_H + q_S)}{p^4}$ for each signature simulation.

Now, since B can simulate the signatures without the knowledge of the secret key, then we can build from B and A a *no message* adversary (that is an adversary against the

ΣUnforgeability game without call to the `Sign` oracle), say C, and apply Lemma 10. If A succeeds with probability $\epsilon \geq \frac{1}{2^\lambda} + \eta + \frac{q_S \cdot (q_H + q_S)}{p^4}$, then C makes a forgery with probability greater than $\frac{1}{2^\lambda} + \eta$. From Lemma 10, we extract two related signatures, with the same hash-query but different challenges $(m, \mathsf{dpk}, \mathsf{nym}, T)$, (R_1, R_2, R_3), $c, (s_f, s_x, s_a, s_b, s_d)$, $c', (s'_f, s'_x, s'_a, s'_b, s'_d)$ in expected time $O(\frac{q_H t}{\eta})$. Finally, by applying Lemma 3, B gets a valid (f, A, x) and returns $(f, A, x, e(A, G_2))$.

A.5 Proof of Theorem 6

Let $(H_1, {H_1}^\theta, {H_1}^{\theta^2}, \ldots, {H_1}^{\theta^q}, H_2, {H_2}^\theta) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2$ be a SDH instance on a bilinear environment $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ for an unknown and uniformly distributed $\theta \in \mathbb{Z}_p^*$. We build an algorithm B that outputs $(c, {H_1}^{1/(\theta+c)})$, for a $c \in \mathbb{Z}_p \setminus \{-\theta\}$, from an adversary A against the seclusiveness of our scheme.

Parameters. B picks $k \xleftarrow{\$} \{1, \ldots, q\}$, $x_1, \ldots, x_q \xleftarrow{\$} \mathbb{Z}_p$ and $s_1, \ldots, s_q \xleftarrow{\$} \mathbb{Z}_p$.
B sets $G_2 := H_2$, $Y_1 := ({H_1}^\theta) \cdot {H_1}^{-x_k}$, and $Y_2 := ({H_2}^\theta) \cdot {H_2}^{-x_k}$.
[*Note.* The issuing secret key is $\gamma := \theta - x_k$, but B does not know γ.]
For $\{x_1, \ldots, x_q\} \in \mathbb{F}_p$, let P, P_m and P_m^- for $m \in [1, q]$ be the following polynomials on $\mathbb{F}_p[X]$:

$$P := \prod_{n=1}^{q} (X + x_n - x_k) \quad P_m := \prod_{n=1, n\neq m}^{q} (X + x_n - x_k) \quad P_m^- := \prod_{n=1, n\neq m, n\neq k}^{q} (X + x_n - x_k).$$

[*Note.* We have $P = X \cdot P_k$, $P_m = X \cdot P_m^-$ if $m \neq k$, and $P_k = P_k^-$.]
Expanding P on θ, we get $P(\theta) = \sum_{n=0}^{q} a_n \theta^n$ for some $\{a_n\}_{n=0}^q$ depending on the x_n. Since B knows ${H_1}^{\theta^n}$ from the q-SDH challenge, B is able to compute ${H_1}^{P(\theta)}$ without the knowledge of θ. The same remark is equally true for P_m and P_m^-.
B picks $\alpha \xleftarrow{\$} \mathbb{Z}_p$, $\beta \xleftarrow{\$} \mathbb{Z}_p^*$, sets $G_1 := {H_1}^{\beta(\alpha P(\theta) - s_k P_k(\theta))}$ and $H := {H_1}^{\beta P_k(\theta)}$.
B gives A parameters $(e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, H, G_2, Y_1, Y_2, \mathcal{H})$.
[*Note.* Thanks to α and β, the parameters are distributed as in the `D` scheme.]

Simulating the issuing algorithm. Let `Aux` be the following sub-routine, taking as input $(f', \mathsf{ctr}) \in \mathbb{Z}_p \times \mathbb{N}$ and outputting (f'', A, x, Z) as in the fourth step of the `D.Issue` algorithm.
B sets $A_{\mathsf{ctr}} := {H_1}^{\beta(\alpha P_{\mathsf{ctr}}(\theta) + P_{\mathsf{ctr}}^-(\theta)(s_{\mathsf{ctr}} - s_k))}$ and returns $(s_{\mathsf{ctr}} - f', A_{\mathsf{ctr}}, x_{\mathsf{ctr}}, e(A_{\mathsf{ctr}}, G_2))$.
One can check that $(s_{\mathsf{ctr}}, A_{\mathsf{ctr}}, x_{\mathsf{ctr}})$ is a valid certificate under w:

$$\begin{aligned}
\left(G_1 \cdot H^{s_{\mathsf{ctr}}}\right)^{\frac{1}{\gamma + x_{\mathsf{ctr}}}} &= \left({H_1}^{\beta(\alpha P(\theta) - s_k P_k(\theta))} \cdot {H_1}^{\beta P_k(\theta) s_{\mathsf{ctr}}}\right)^{\frac{1}{\theta - x_k + x_{\mathsf{ctr}}}} \\
&= {H_1}^{(\beta\alpha P_{\mathsf{ctr}}(\theta)(\theta + x_{\mathsf{ctr}} - x_k) - \beta s_k P_{\mathsf{ctr}}^-(\theta)(\theta + x_{\mathsf{ctr}} - x_k) + \beta s_{\mathsf{ctr}} P_{\mathsf{ctr}}^-(\theta)(\theta + x_{\mathsf{ctr}} - x_k)) \frac{1}{\theta - x_k + x_{\mathsf{ctr}}}} \\
&= {H_1}^{(\beta\alpha P_{\mathsf{ctr}}(\theta) - \beta s_k P_{\mathsf{ctr}}^-(\theta) + \beta s_{\mathsf{ctr}} P_{\mathsf{ctr}}^-(\theta)) \frac{\theta + x_{\mathsf{ctr}} - x_k}{\theta - x_k + x_{\mathsf{ctr}}}} = {H_1}^{\beta(\alpha P_{\mathsf{ctr}}(\theta) + P_{\mathsf{ctr}}^-(\theta)(s_{\mathsf{ctr}} - s_k))} = A_{\mathsf{ctr}}.
\end{aligned}$$

We note that, if $\mathsf{ctr} = k$, we have $A_k = {H_1}^{\beta\alpha P_k(\theta)}$.

Simulating the oracles. B initializes a counter $\mathsf{ctr} := 0$.

`AddDomain`(j). When A wants to add a domain j, B sets $\mathbf{RL}[j] := \{\}$, computes a new public key $\mathsf{dpk}_j := {G_1}^{r_j}$ for a fresh $r_j \xleftarrow{\$} \mathbb{Z}_p$, records $\mathbf{dpk}[j] := \mathsf{dpk}_j$, computes pseudonyms for all existing honest users $i \in \mathcal{HU}$ as $\mathsf{nym}_{ij} := H^{f_i} \cdot {\mathsf{dpk}_j}^{x_i}$ for $(f_i, A_i, x_i, Z_i) \leftarrow \mathbf{usk}[i]$, records $\mathbf{nym}[i][j] := \mathsf{nym}_{ij}$, and sends back $(\mathsf{dpk}_j, \{\mathsf{nym}_{ij}\}_{i \in \mathcal{HU}})$ to A.

$\texttt{AddUser}(i)$. When A asks for adding a new honest user, B increments the counter $\mathsf{ctr} := \mathsf{ctr}+1$, picks $f'_{\mathsf{ctr}} \xleftarrow{\$} \mathbb{Z}_p$, calls the $\texttt{Aux}$ procedure on input $(f'_{\mathsf{ctr}}, \mathsf{ctr})$, gets $(f''_{\mathsf{ctr}}, A_{\mathsf{ctr}}, x_{\mathsf{ctr}}, Z_{\mathsf{ctr}})$, records $\mathbf{usk}[\mathsf{ctr}] := (f'_{\mathsf{ctr}} + f''_{\mathsf{ctr}}, A_{\mathsf{ctr}}, x_{\mathsf{ctr}}, Z_{\mathsf{ctr}})$ and $\mathbf{rt}[\mathsf{ctr}] := (H^{f'+f''_{\mathsf{ctr}}}, x_{\mathsf{ctr}})$, and computes pseudonyms for all existing $j \in \mathcal{D}$ as $\mathbf{nym}[\mathsf{ctr}] := H^{f'+f''_{\mathsf{ctr}}} \cdot \mathsf{dpk}_j{}^{x_{\mathsf{ctr}}}$ where $\mathsf{dpk}_j \leftarrow \mathbf{dpk}[j]$.

$\texttt{CorruptUser}(i)$. When A asks for adding a new corrupted user, B does nothing.

$\texttt{ReadRegistrationTable}(i)$. When A asks for the content of the registration table, B returns $\mathbf{rt}[i]$.

$\texttt{SendToIssuer}(i, M_{in})$. When A asks for interacting with the issuer as a corrupted user, B increments the counter $\mathsf{ctr} := \mathsf{ctr} + 1$ and initializes the table $\mathbf{nym}[\mathsf{ctr}] := \perp$. Since the $\texttt{D.Join} \leftrightarrow \texttt{Issue}$ protocol is one-round, then $M_{in} := (F, \Pi)$. From Π, thanks to the extraction key ek, B extracts f' such that $F := H^{f'}$. B then calls the $\texttt{Aux}$ procedure on the input (f', ctr), and gets $(f''_{\mathsf{ctr}}, A_{\mathsf{ctr}}, f_{\mathsf{ctr}}, Z_{\mathsf{ctr}})$ back, which B transfers to A. B records $\mathbf{usk}[\mathsf{ctr}] := (f' + f''_{\mathsf{ctr}}, A_{\mathsf{ctr}}, x_{\mathsf{ctr}}, Z_{\mathsf{ctr}})$ and $\mathbf{rt}[\mathsf{ctr}] := (H^{f'+f''_{\mathsf{ctr}}}, x_{\mathsf{ctr}})$.

$\texttt{Sign}(i, j, m)$. When A asks for a signature from $i \in \mathcal{HU}$ near $j \in \mathcal{D}$ on a message m, then B returns $\texttt{D.Sign}(\mathsf{gpk}, \mathbf{dpk}[j], \mathbf{usk}[i], \mathbf{nym}[i][j], m)$.

Response. A eventually outputs $(\mathsf{dpk}_*, \mathsf{nym}_*, m_*, \sigma_*)$. If this is a non trivial response, then there exists $j \in \mathcal{D}$ such that $\mathsf{dpk}_* = \mathbf{dpk}[j]$. At this point, B blacklists all users near j, by updating $\mathbf{RL}[j]$. Since the initialization for corrupted users was postponed to the issuing step, then for all $i \in \mathcal{U}$, we have (i) $\mathbf{usk}[i] \neq \perp$ and (ii) $\mathbf{rt}[i] \neq \perp$. If the response is valid, then $\texttt{Verify}(\mathsf{gpk}, \mathsf{dpk}_*, \mathsf{nym}_*, m, \sigma, \mathbf{RL}[j]) = \mathsf{accept}$. This means that through the Σ-unforgeability Lemma (Lemma 5, Game 1), B can extract a new certificate (f_*, A_*, x_*, Z_*) in reasonable expecting time. Since this certificate is valid, we have

$$A_* = \left(G_1 \cdot H^{f_*}\right)^{\frac{1}{\gamma+x_*}} = \left({H_1}^{\beta(\alpha P(\theta) - s_k P_k(\theta))} \cdot ({H_1}^{\beta P_k(\theta)})^{f_*}\right)^{\frac{1}{\theta - x_k + x_*}} = {H_1}^{\beta P_k(\theta)(\alpha\theta - s_k + f_*)\frac{1}{\theta - x_k + x_*}}.$$

Solving the SDH challenge. Since from (ii) for all $i \in \mathcal{U}$, $\mathbf{rt}[i] \neq \perp$, then, if the signature is not rejected, then there is no $n \in [1, q]$, such that $\mathsf{nym}_* = H^{f_n} \cdot (\mathsf{dpk}_*)^{x_n}$. Hence (iii) $(f_*, x_*) \notin \{(f_1, x_1), \ldots, (f_q, x_q)\}$. We distinguish two cases A and B.

(A) $x_* \in \{x_1, \ldots, x_q\}$. (A.I) If $x_* \neq x_k$, B returns $\perp$ and aborts. (A.II) Let us now assume that $x_* = x_k$. We have $f_* \neq s_k$ (since $f_* = s_k$ contradicts (iii)) and

$$\begin{aligned}
({A_*}^{s_k} \cdot {A_k}^{-f_*})^{\frac{1}{s_k - f_*}} &= \left({H_1}^{s_k \beta P_k(\theta)(\alpha\theta - s_k + f_*)\frac{1}{\theta - x_k + x_*}} \cdot {H_1}^{-f_* \beta \alpha P_k(\theta)}\right)^{\frac{1}{s_k - f_*}} \\
&= {H_1}^{(s_k \beta P_k(\theta)(\alpha\theta - s_k + f_*)\frac{1}{\theta} - f_* \beta \alpha P_k(\theta))\frac{1}{s_k - f_*}} \\
&= {H_1}^{s_k \beta P_k(\theta)(\alpha\theta - s_k + f_*)\frac{1}{\theta}\frac{1}{s_k - f_*} - f_* \beta \alpha \theta P_k(\theta)\frac{1}{\theta}\frac{1}{s_k - f_*}} \\
&= {H_1}^{(\beta P_k(\theta)\alpha\theta(s_k - f_*) + s_k \beta P_k(\theta)(s_k + f_*))\frac{1}{\theta}\frac{1}{s_k - f_*}} = {H_1}^{\beta(\alpha P(\theta) - s_k P_k(\theta))\frac{1}{\theta}}
\end{aligned}$$

P vanishes in 0, but this is not the case for P_k. Then by dividing $\beta(\alpha P(\theta) - s_k P_k(\theta))$ by θ we get R and Q such that

$$C := R(0) = -\beta s_k \left[\prod_{n=1, n\neq k}^{q} (x_n - x_*)\right] \quad \text{and} \quad ({A_*}^{s_k} \cdot {A_k}^{-f_*})^{\frac{1}{s_k - f_*}} = {H_1}^{\frac{C}{\theta} + Q(\theta)}$$

where $C \neq 0$. B computes ${H_1}^{1/\theta} := (({A_*}^{s_k} \cdot {A_k}^{-f_*})^{\frac{1}{s_k - f_*}} \cdot {H_1}^{-Q(\theta)})^{1/C}$, sets $c := 0$ and returns $(0, {H_1}^{1/\theta})$ as a solution to the SDH challenge.

(B) $x_* \notin \{x_1, \ldots, x_q\}$. In particular, we have (iv) $x_n - x_* \neq 0$ for all $n \in [1, q]$. Let us now consider the quantity $\beta P_k(\theta)(\alpha\theta + f_* - s_k)$ as a polynomial D in θ. If we carry out the Euclidean division of D by $(\theta + x_* - x_k)$, we get Q and R such that $D(\theta) = (\theta + x_* - x_k)Q(\theta) + R(\theta)$. As $(\theta + x_* - x_k)$ is a first degree polynomial $X - (x_k - x_*)$, we know that $R(\theta) = D(x_k - x_*)$, so B can compute

$$C := R(\theta) = D(x_k - x_*) = \beta \left[\prod_{n=1, n\neq k}^{q} (x_n - x_*) \right] (\alpha(x_k - x_*) + f_* - s_k).$$

Let us recall that $A_* = {H_1}^{\beta P_k(\theta)(\alpha\theta - s_k + f_*)\frac{1}{\theta - x_k + x_*}}$ so $A_* = {H_1}^{\frac{(\theta + x_* - x_k)Q(\theta) + R(\theta)}{\theta - x_k + x_*}} = {H_1}^{Q(\theta) + \frac{C}{\theta + x_* - x_k}}$. B can compute ${H_1}^{Q(\theta)}$ from the SDH challenge. We again distinguish two cases.

(B.I) $(f_* - s_k) \neq \alpha(x_* - x_k)$. In this case, $C \neq 0$ by (iv) and by the choice of β, so B can compute ${H_1}^{\frac{1}{\theta + x_* - x_k}} = \left(A_* \cdot {H_1}^{-Q(\theta)}\right)^{\frac{1}{C}}$, set $c = x_* - x_k$, and return $(c, {H_1}^{1/(\theta + c)})$ as a solution to the SDH challenge.

(B.II) $(f_* - s_k) = \alpha(x_* - x_k)$. B returns $\perp$ and aborts.

Probability of success. No information is available about k from A's point of view. So:

- in case A, B succeeds with probability $1/q$.
- in case B, if B aborts, then we have

$$A_* = {H_1}^{\beta P_k(\theta)(\alpha\theta - s_k + f_*)\frac{1}{\theta - x_k + x_*}} = {H_1}^{\beta P_k(\theta)(\alpha\theta + \alpha(x_* - x_k))\frac{1}{\theta - x_k + x_*}} = {H_1}^{\beta\alpha P_k(\theta)} = A_k.$$

  If $A_* \notin \{A_n\}_{n=1}^{q}$, then B does not abort. If (f_*, x_*) is a new pair corresponding to an existing A_n, then B aborts with probability $1/q$ (plus a negligible quantity: all A_n are distinct with very high probability). So B succeeds with probability at least $1 - 1/q$.

As a conclusion, if A outputs a valid forgery with probability ϵ, then B solves the SDH challenge with probability at least $\epsilon/2q$ (considering the more pessimistic case).

Cost of the reduction. We need $O(q)$ multiplications in order to compute ${H_1}^{P(\theta)}$, and we need to carry out an Euclidean division of a polynomial of degree q by a polynomial of degree 1, which can be done in $O(q^2)$ (in a standard and non optimized way). □

A.6 Proof of Theorem 7

Let A be an adversary against the unforgeability of the D scheme. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a bilinear environment and (G, H) be a discrete logarithm instance in $\mathbb{G}_1$. We construct an algorithm B that computes $\theta := \log_G \mathsf{H}$.

Parameters. B picks $G_1 \xleftarrow{\$} \mathbb{G}_1$, $G_2 \xleftarrow{\$} \mathbb{G}_2$, $\gamma \xleftarrow{\$} \mathbb{Z}_p$, sets $H := G$, $Y_1 := {G_1}^{\gamma}$, and $Y_2 := {G_2}^{\gamma}$. B gives parameters gpk $:= (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, H, G_2, Y_1, Y_2)$ to A. In addition, B generates parameters for the extractable commitment scheme Ext-Commit and the non-interactive proof system NIZKPEqDL.

Simulating the oracles. At each time B interacts (as an honest user) with A (as the corrupted issuing authority), B follows the Join procedure, but for the i-th user. In the latter case, B sets $F' := \mathsf{H}$ and simulates the proof Π (in the random oracle model, with negligible probability of failure, the precise quantity depending on the actual non-interactive proof) and

receives $(f''_{\mathsf{i}}, A_{\mathsf{i}}, x_{\mathsf{i}}, Z_{\mathsf{i}})$ where $A_{\mathsf{i}} = (G_1 \cdot \mathsf{H} \cdot H^{f''_{\mathsf{i}}})^{\frac{1}{x_{\mathsf{i}}+\gamma}}$ for some f''_{i}. B does not know f_{i}, but can compute the pseudonym for i and all domains $j \in \mathcal{D}$ by setting $\mathsf{nym}_{\mathsf{i}j} := \mathsf{H} \cdot H^{f''_{\mathsf{i}}} \cdot \mathsf{dpk}_j{}^{x_{\mathsf{i}}}$. B records $\mathbf{usk}[\mathsf{i}] := (f''_{\mathsf{i}}, A_{\mathsf{i}}, x_{\mathsf{i}}, Z_{\mathsf{i}})$. When A asks for a signature from the i-th user, B simulates a signature including $\mathsf{nym}_{\mathsf{i}j}$ (in the random oracle model). Other signatures are normally computed and other oracles are normally simulated.

Response. A play of A eventually gives $(\mathsf{dpk}_*, \mathsf{nym}_*, m_*, \sigma_*)$. If this is a valid and non trivial response, then (i) we can find a domain j such that $\mathsf{dpk}_* = \mathbf{dpk}[j]$ and an honest user i with consistent values $\mathsf{nym}_{ij} \in \mathbf{nym}[i][j]$, $(F_i, x_i) \in \mathbf{rt}[i]$ and $(*, A_i, x_i, Z_i) \in \mathbf{usk}[i]$ such that $\mathsf{nym}_* = \mathsf{nym}_{ij} = F_i \cdot (\mathsf{dpk}_*)^{x_i}$, and (ii) according to the Σ-unforgeability (Lemma 5, Game 2), we are able to extract a valid certificate (f_*, A_*, x_*, Z_*) where (in particular) $\mathsf{nym}_* = H^{f_*} \cdot (\mathsf{dpk}_*)^{x_*}$. Since discrete representations in $\mathbb{G}_1$ are unique modulo p, then we have that $f_* = \log_g F_i$ (the pseudonym must be valid in a non trivial forgery) and $x_* = x_i$. With probability $\frac{1}{q_U}$ we have $i = \mathsf{i}$, since i is independent of the view of A. This implies that $A_{\mathsf{i}} = A_*$ (a value A is determined by f, x and γ). Therefore $A_* = (G_1 \cdot G^{f_*})^{\frac{1}{x_*+\gamma}} = (G_1 \cdot \mathsf{H} \cdot H^{f''_{\mathsf{i}}})^{\frac{1}{x_*+\gamma}}$ and we obtain $\theta = f_* - f''_{\mathsf{i}}$. □

A.7 Proof of Theorem 8

Let A be an adversary against the cross-domain-anonymity of the `D` scheme. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a bilinear environment and $(G, \mathsf{A}, \mathsf{B}, \mathsf{C})$ be a Diffie-Hellman instance in $\mathbb{G}_1$. We construct B that decides whether C is the Diffie-Hellman of A and B in base G. B picks two bits $b_A, b_B \xleftarrow{\$} \{0,1\}$, a random user $\mathsf{i} \xleftarrow{\$} \{1, \ldots, q_U\}$ and a random domain $\mathsf{j} \xleftarrow{\$} \{1, \ldots, q_D\}$.

Parameters. The parameters $\mathsf{gpk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, H, G_2, Y_1, Y_2)$ for the `D` scheme are computed honestly, knowing $\mathsf{isk} = \gamma$, except that $G_1 := G$.

Simulating the oracles. Since the challenger knows the issuing secret key, and moreover can simulate signatures on behalf of any user, then the simulation of the oracles is done without noticeable facts, except that B acts as if $\mathsf{dpk}_{\mathsf{j}} = \mathsf{B}$ and $x_{\mathsf{i}} = \log_g \mathsf{A}$. B aborts and returns a random bit if the user i is queried to the `UserSecretKey` oracle (B has no valid $\mathsf{usk}_{\mathsf{i}}$) or if i and j are not queried to the `Challenge` oracle. Moreover, B aborts if $\mathsf{nym}_{\mathsf{ij}}$ is not returned by the `Challenge` oracle. The reduction relies upon the following `SimNym` procedure, called by the `AddUser` and `AddDomain` oracles.

`SimNym`(i, j).

(I) If $i \neq \mathsf{i}$ and $j \neq \mathsf{j}$, then B retrieves $(f_i, A_i, x_i, Z_i) \leftarrow \mathbf{usk}[i]$, $(\mathsf{dpk}_j, r_j) \leftarrow \mathbf{dpk}[j]$ and sets $\mathbf{nym}[i][j] := H^{f_i} \cdot {G_1}^{r_j x_i}$.

(II) If $i = \mathsf{i}$ and $j \neq \mathsf{j}$, then B retrieves $(f_i, *, *, *) \leftarrow \mathbf{usk}[i]$, $(\mathsf{dpk}_j, r_j) \leftarrow \mathbf{dpk}[j]$ and sets $\mathbf{nym}[i][j] := H^{f_i} \cdot \mathsf{A}^{r_j}$.

(III) If $i \neq \mathsf{i}$ and $j = \mathsf{j}$, then B retrieves $(f_i, A_i, x_i, Z_i) \leftarrow \mathbf{usk}[i]$ and sets $\mathbf{nym}[i][j] := H^{f_i} \cdot \mathsf{B}^{x_i}$.

(IV) If $i = \mathsf{i}$ and $j = \mathsf{j}$, then B retrieves $(f_i, *, *, *) \leftarrow \mathbf{usk}[i]$, sets $\mathbf{nym}[i][j] := H^{f_i} \cdot \mathsf{C}$.

Response. Eventually, A outputs a bit b', its guess for $(b_A == b_B)$. B returns true if $(b' == (b_A == b_B))$, or false otherwise, as response to its own challenge.

Probability to win. Let us now estimate the advantage that B has of solving the DDH challenge.

$$\mathbf{Adv}_B^{\mathsf{DDH}} = \left| \Pr[B \Rightarrow \mathsf{true} | \mathsf{C} = \mathsf{DH}_g(\mathsf{A}, \mathsf{B})] - \Pr[B \Rightarrow \mathsf{true} | \mathsf{C} \text{ is random}] \right|$$

$$= \left| \Pr[\mathbf{abort}] \cdot \mathbf{P}_1 + \Pr[\overline{\mathbf{abort}}] \cdot \mathbf{P}_2 - \Pr[\mathbf{abort}] \cdot \mathbf{P}_3 - \Pr[\overline{\mathbf{abort}}] \cdot \mathbf{P}_4 \right|$$

where

$$\mathbf{P}_1 := \Pr[B \Rightarrow \mathsf{true} | \mathbf{abort} \wedge \mathsf{C} = \mathsf{DH}_g(\mathsf{A}, \mathsf{B})], \quad \mathbf{P}_2 := \Pr[B \Rightarrow \mathsf{true} | \overline{\mathbf{abort}} \wedge \mathsf{C} = \mathsf{DH}_g(\mathsf{A}, \mathsf{B})],$$
$$\mathbf{P}_3 := \Pr[B \Rightarrow \mathsf{true} | \mathbf{abort} \wedge \mathsf{C} \text{ is random}], \quad \mathbf{P}_4 := \Pr[B \Rightarrow \mathsf{true} | \overline{\mathbf{abort}} \wedge \mathsf{C} \text{ is random}].$$

We know that $\mathbf{P}_1 = \mathbf{P}_3 = \frac{1}{2}$, since B outputs a random bit if the simulation aborts. Moreover $\mathbf{P}_2 = \frac{(\epsilon+1)}{2}$: if $\mathsf{C} = \mathsf{DH}_g(\mathsf{A}, \mathsf{B})$ and B does not abort, then the simulation is perfect, and $\mathbf{P}_2$ is the probability that A solves its own anonymity challenge. Finally, we have $\mathbf{P}_4 = \frac{1}{2}$: if C is a random element in $\mathbb{G}_1$, then no information is available to guess if the users are identical in the output of the `Challenge` oracle and A has advantage 0 to win the game.

It remains to estimate the probability to abort. B aborts if it fails on setting some random oracle values or if A asks for the secret of i or if $\mathsf{nym}_{\mathsf{ij}}$ is not returned by the `Challenge` oracle. Let us define the following events.

E_0: B does not abort on simulating signatures.

E_1: B does not abort on `UserSecretKey` before the `Challenge` call.

E_2: B does not abort on `Challenge`.

E_3: B does not abort on `UserSecretKey` after the `Challenge` call.

We have: $\Pr[\overline{\mathbf{abort}}] = \Pr[E_0 \wedge E_1 \wedge E_2 \wedge E_3]$
$= \Pr[E_0] \cdot \Pr[E_1|E_0] \cdot \Pr[E_2|E_0 \wedge E_1] \cdot \Pr[E_3|E_0 \wedge E_1 \wedge E_2]$.

Fact: $\Pr[E_0] \geq 1 - \frac{q_S \cdot (q_H + q_S)}{p^4}$. For the simulation of a single signature, the probability to fail on setting some random oracle value is bounded by $\frac{(q_H+q_S)}{p^4}$. There are q_S calls to the `NymSign` oracle, so B fails to simulate signatures with probability at most $\frac{q_S \cdot (q_H+q_S)}{p^4}$.

Fact: $\Pr[E_1|E_0] \geq 1 - \frac{q_C}{q_U}$. The index i is independent of the adversary's view until asked to `UserSecretKey`. The distribution of the corresponding secret keys is indeed correct and the oracles' responses are perfectly simulated: the pseudonym $H^f \cdot \mathsf{C}$ does not leak information about C as long as f is unknown to A (as in a Pedersen commitment) and a signature $(T, c, \vec{s})$ leaks no information about f. Then the probability of not aborting after q_C calls to `UserSecretKey` is at least $1 - \frac{q_C}{q_U}$.

Fact: $\Pr[E_2|E_0 \wedge E_1] \geq \frac{2}{(q_U - q_C) \cdot q_D}$. Let us first estimate the probability that i and j are queried to the `Challenge` oracle. The adversary A chooses two distinct users among $2 \leq q'_U - q'_C \leq q_U$ and two distinct domains among $2 \leq q'_D \leq q_D$, where q'_U, q'_D and q'_C are respectively the number of users, domains and `UserSecretKey` calls until the `Challenge` call. If A did not abort before the Challenge call, i and j are independent of the adversary's view at this point. Moreover, they are mutually independent. So $\Pr[E_2|E_0 \wedge E_1] = \frac{4}{(q'_U - q'_C) \cdot q'_D} \geq \frac{4}{(q_U - q_C) \cdot q_D}$. Then, if i and j are queried to the `Challenge` oracle, B aborts with probability $\frac{1}{2}$, since $\mathsf{nym}_{\mathsf{ij}}$ belongs to the returned pseudonyms with probability $\frac{1}{2}$.

Fact: $\Pr[E_3|E_0 \wedge E_1 \wedge E_2] = 1$. If A did not abort on the `Challenge` call, then i was queried to `Challenge`. So, they will no longer be queried to `UserSecretKey` because the cross-domain anonymity definition disallows it.

Putting it all together we obtain:

$$\Pr[\overline{\mathbf{abort}}] \geq \frac{2}{(q_U - q_C) \cdot q_D} \cdot \left(1 - \frac{q_C}{q_U}\right) \cdot \left(1 - \frac{q_S \cdot (q_H + q_S)}{p^4}\right) = \frac{2}{q_U \cdot q_D} \cdot \left(1 - \frac{q_S \cdot (q_H + q_S)}{p^4}\right)$$

Conclusion. From the precedent facts, we conclude that:

$$\begin{aligned}\mathbf{Adv}_B^{\mathsf{DDH}} &= \left| \Pr[\mathbf{abort}] \cdot \frac{1}{2} + \Pr[\overline{\mathbf{abort}}] \cdot \frac{\epsilon + 1}{2} - \Pr[\mathbf{abort}] \cdot \frac{1}{2} - \Pr[\overline{\mathbf{abort}}] \cdot \frac{1}{2} \right| \\ &\geq \frac{2}{q_U \cdot q_D} \cdot \left(1 - \frac{q_S \cdot (q_H + q_S)}{p^4}\right) \cdot \frac{\epsilon}{2} = \frac{\epsilon}{q_U \cdot q_D} \cdot \left(1 - \frac{q_S \cdot (q_H + q_S)}{p^4}\right)\end{aligned}$$

If ϵ is non negligible, then so is the advantage of B. □

A.8 Proof of Theorem 9

We adapt the proofs of Theorems 6, 7 and 8 to the extended model of Section 4.3 where the oracles `GetPreComp` and `Sign′` are also supplied to the adversary.

Parameters. We first simulate parameters and user secret keys.

Seclusiveness: as in Section A.5.

Unforgeability: for q_u users, pick $G_2 \leftarrow \mathbb{G}_2 \setminus \{\mathbf{1}_{\mathbb{G}_2}\}$; $\beta, \gamma, f_1, \ldots, f_{\mathsf{i}-1}, f_{\mathsf{i}+1}, \ldots, f_{q_u} \leftarrow \mathbb{Z}_p^*$, $x_1, \ldots, x_{q_u} \leftarrow \mathbb{Z}_p \setminus \{\gamma\}$, and set $G_1 := G^{(\gamma+x_1)\cdots(\gamma+x_{q_u})}$, $H := {G_1}^{\beta}$, $Y_1 := H^{\gamma}$, $Y_2 := {G_2}^{\gamma}$ $A_i := G^{(1+\beta \cdot f_i) \cdot \prod_{n \in [1,q_u] \setminus \{i\}} (\gamma + x_n)}$ for all $i \in [1, q_u], i \neq \mathsf{i}$, and $A_{\mathsf{i}} := (G \cdot \mathsf{H}^{\beta})^{\prod_{n \in [1,q_u] \setminus \{\mathsf{i}\}} (\gamma + x_n)}$ where (G, H) is the DL challenge. This implicitly sets $f_{\mathsf{i}} = \log_G \mathsf{H}$.

Cross-domain anonymity: for q_u users, pick $G_2 \leftarrow \mathbb{G}_2 \setminus \{\mathbf{1}_{\mathbb{G}_2}\}$; $\beta, \gamma, f_1, \ldots f_{q_u} \leftarrow \mathbb{Z}_p^*$, $x_1, \ldots, x_{\mathsf{i}-1}, x_{\mathsf{i}+1}, \ldots, x_{q_u} \leftarrow \mathbb{Z}_p \setminus \{\gamma\}$, and sets $G_1 := (G \cdot \mathsf{A})^{\prod_{n \in [1,qu] \setminus \{\mathsf{i}\}} (\gamma + x_n)}$, $H := {G_1}^{\beta}$, $Y_1 := H^{\gamma}$, $Y_2 := {G_2}^{\gamma}$, $A_i := (G^{\gamma} \cdot \mathsf{A})^{(1+\beta \cdot f_i) \cdot \prod_{n \in [1,q_u] \setminus \{i,\mathsf{i}\}} (\gamma + x_n)}$ for all $i \in [1, q_u]$, $i \neq \mathsf{i}$, and $A_{\mathsf{i}} := G^{(1+\beta \cdot f_{\mathsf{i}}) \cdot \prod_{n \in [1,q_u] \setminus \{\mathsf{i}\}} (\gamma + x_n)}$ where $(G, \mathsf{A}, \mathsf{B}, \mathsf{C})$ is the DDH challenge[6]. This implicitly sets $x_{\mathsf{i}} = \log_G \mathsf{A}$.

Pre-computations. We then simulate the pre-computations (identically in the three proofs).

`GetPreComp`(i, j, m). Set $B := {A_i}^{\gamma \cdot c + s_x} \cdot {G_1}^{-c} \cdot H^{a \cdot s_x - s_f - s_b} \cdot {Y_1}^{a \cdot c - s_a}$.

One may check that B is perfectly simulated:

$$\begin{aligned}B &= (A_i)^{\gamma \cdot c + r_x + c \cdot x} \cdot {G_1}^{-c} \cdot (Y_1)^{a \cdot c - r_a - c \cdot a} \cdot H^{a \cdot (r_x + c \cdot x) - r_f - c \cdot f - r_b - c \cdot a \cdot x} \\ &= (A_i)^{c \cdot (\gamma + x)} \cdot {G_1}^{-c} \cdot (Y_1)^{c \cdot a - c \cdot a} \cdot H^{c \cdot a \cdot x - c \cdot f - c \cdot a \cdot x} \cdot (A_i)^{r_x} \cdot H^{a \cdot r_x - r_f - r_b} \cdot (Y_1)^{-r_a} \\ &= \left(G_1 \cdot H^f \cdot {G_1}^{-1} \cdot H^{-f}\right)^c \cdot (A_i)^{r_x} \cdot H^{a \cdot r_x - r_f - r_b} \cdot (Y_1)^{-r_a} \\ &= (A_i)^{r_x} \cdot H^{a \cdot r_x - r_f - r_b} \cdot (Y_1)^{-r_a}.\end{aligned}$$

Signatures. We now simulate the `Sign′` oracle (identically in the three proofs).

[6] In the delegation case, the domain keys are computed from a generator (if any) different from G_1 and H (here denoted by G).

$\texttt{Sign}'(i, j, D)$. Retrieve m, B, c, s_a, s_x, s_a, s_b from the `GetPreComp` step. Whatever D is (D may not equal $e(B, G_2)$), B picks $s_d \xleftarrow{\$} \mathbb{Z}_p$, computes T, R_1 and R_2 as usual ($T \xleftarrow{\$} \mathbb{G}_1$, $R_1 := H^{s_f} \cdot \mathsf{dpk}^{s_x} \cdot \mathsf{nym}^{-c}$ and $R_2 := \mathsf{nym}^{s_a} \cdot H^{-s_d} \cdot \mathsf{dpk}^{-s_b}$) and sets c as the random oracle's hash value for the input $\mathsf{dpk} \| \mathsf{nym} \| T \| R_1 \| R_2 \| D \| m$.

If the D value supplied by the adversary is correctly computed with respect to B, then the challenger returns a valid signature. If not, then the signature is no longer valid but, from the adversary's point of view, the response $(T, c, \vec{s})$ remains consistent with the pre-computed B and the incorrect D supplied by the adversary. In other words, B perfectly simulates the protocol given Figure 3 as long as B does not fail on setting the random oracle value.

Regarding this probability to fail, note that s_d is picked after the reception of R_3 and that R_2 depends on s_d. Since s_d is uniformly picked in $\mathbb{Z}_p$, then R_2 is uniform in $\mathbb{G}_1$. So the simulation fails in setting a random oracle value with probability at most $\frac{q_S + q'_S + q_H}{p}$ for each $\texttt{Sign}'$ simulation (where q'_S is the number of $\texttt{Sign}'$ queries and q_S and q_H are still the number of `Sign` / `NymSign` and $\mathcal{H}$ queries respectively).

This concludes the proof. □