Pseudonymous signatures for eID: efficient and strongly secure dynamic domain-specific pseudonymous signatures

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

[1]Morpho, Issy-Les-Moulineaux, France
[2]Télécom ParisTech, Paris, France

Abstract

The notion of domain-specific pseudonymous signatures (DSPS) has recently been introduced for the private authentication of ID documents by Bender *et al.* at the ISC'12 conference. Thanks to this primitive, the ID document, which embeds a chip with computational abilities, is able to authenticate to a service provider through a reader, and the resulting signatures are anonymous, linkable inside the service and unlinkable across services. In a subsequent work, Bringer *et al.* proposed at the NSS'13 conference to enhance security and privacy of DSPS through group signature techniques. In this work, we improve on these proposals in three ways. We note that using the full power of group signatures is not needed for the construction of DSPS, and we provide an optimized construction that achieves the same strong security and privacy properties as the previous proposal of Bringer *et al.* while being more efficient. We also spot several imprecisions in previous formalizations and provide clean security definitions for the dynamic and adaptive case. Finally, we study the implementation of our protocol in a chip and show that our solution is well-suited for these limited environments.

1 Introduction

Recently, the German BSI agency introduced several security mechanisms regarding the use of identity tokens for authentication purposes [14]. In such situations, a token for electronic IDentification, Authentication and trust Services (aka an eIDAS token) connects to a Service Provider (SP) through a reader (for concreteness, one might see the eIDAS token as a passport equipped with a chip). The security mechanisms of [14] can be summarized as follows. First of all, during the PACE protocol (*Password Authenticated Connection Establishment*), the eIDAS token and the reader establish a secure channel. Then, during the EAC protocol (*Extended Access Control*), the eIDAS token and the service provider authenticate each other through another secure channel. The reader transfers the exchanged messages. At last, during the (optional) RI protocol (*Restricted Identification*), the eIDAS token gives its pseudonym for the service to the service provider. This pseudonym enables the service provider to link users inside its service. However, across the services, users are still unlinkable. The latter property is called *cross-domain anonymity*. Such a 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 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 [22, 6, 12] adopt a "signature mode" for the use of pseudonyms. This approach has also been integrated in the latest developments of the eIDAS protocols [14]. Such a signature mode between an eID token and a service provider can be described as follows.

1. The service provider sends to the eID token the public key dpk of the service and a message m.
2. The eID token computes a pseudonym nym as a deterministic function of its secret key usk and the public key dpk.
3. The eID token signs the message m with its

A preliminary version of this work appeared in the Proceedings of Financial Cryptography and Data Security 2014 [11]. This is an extended version, including new materials and proofs. Part of this work was done while the fourth author was a PhD student at Morpho and Télécom ParisTech.

secret key usk and the pseudonym nym.

4. The eID token sends the signature σ and the pseudonym nym to the service provider.
5. The service provider checks the signature σ of the message m with respect to the pseudonym nym and the domain key dpk.

The interest of pseudonymous signatures exceeds the RI protocol, which motivated their formal study [6]. Such a privacy-preserving cryptographic primitive, coupled with the possession of an eID token, enable users to keep control on the electronic use of their identities.

Our contributions. The work of [6] 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 token. In fact, recovering two users' secrets enables to compute the key of the certification authority. To deal with this concern, the authors of [12] propose to introduce group signatures into this signature mode. In addition to strong privacy properties, group signatures provide collusion resistance even if several users' secrets do leak. The authors of [12] claim that the security model of group signatures gives a security model for DSPS, and, in fact, leave imprecise the definition of the DSPS security properties. Moreover, the model of [6] only concerns the static case, and their anonymity definition is flawed. So clean security definitions for dynamic DSPS remain to be supplied. Our first contribution is then a security model for dynamic DSPS.

We also provide a new construction, which is more efficient than the one of [12], while achieving the same strong security and privacy properties. Our construction highlights the fact that, in some sense, using group signatures is "too strong" for constructing DSPS signatures. Our second contribution is then an efficient, proven secure, dynamic DSPS with short signatures.

Finally, we focus on the use of our DSPS scheme by an eID token with limited capabilities. Our model includes the delegation of some computations to the reader of the token, taking advantage of the computational power of the reader. In this case, the eID token only performs computation in a group of prime order and cheap scalar operations. This is a valuable practical advantage since existing chips might be used. Our proposal avoids the need to deploy *ad hoc* chips, which has an industrial cost.

Related notions. As a privacy-preserving cryptographic primitive, domain-specific pseudonymous signatures share some properties with other primitives. DSPS may appear as a relaxed version of *group signatures*, in which a user is able to sign on behalf of a group, while being anonymous inside the group of the potential signers. More specifically, DSPS share some similarities with *group signatures with verifier local revocation* (VLR) [9] in the sense that, in both primitives, the revocation is done on the verifier's side. However, group signatures are always unlinkable, whereas DSPS achieve some partial linkability. A parallel may be done between DSPS and the notion of *cross-unlinkable VLR group signatures* [13], 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 not the case for the context of DSPS. The difference between DSPS and *pseudonym systems* [23] or *anonymous credential systems* [16] is that DSPS pseudonyms are deterministic whereas anonymous credentials pseudonyms must be unlinkable. DSPS might be seen as anonymous credentials without attributes and with scope-exclusive pseudonyms [15], but with a somehow weaker version of anonymity. In DSPS, the issuer is always able to trace signatures. Moreover, we explicitly address here the splitting of the signature process. *Direct Anonymous Attestations* (DAA) [10] might be seen (following [7]) 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 domains, and where the underlying group signature is replaced by a VLR group signature. Moreover, in the eID token use-case, the token-reader pair might be seen as the TPM-host pair of DAA scheme. However, the primitives remain distinct. The determination of the pseudonyms in DSPS is more restrictive than the usage of the basename in DAA. As a consequence, in DSPS, the issuer is allowed to revoke the users. To the contrary, in DAA, a strong anonymity property might hold even in front of a corrupted issuer (cf. [7]). In addition, the host in DAA always embeds the same chip, but a token is not linked to a specific reader, and might authenticate in front of several readers. These differences im-

pact the DSPS security notions, and allow for different constructions.

Organization of the paper. We first define dynamic domain-specific pseudonymous signatures in Section 2. Then, we present our efficient construction of dynamic DSPS in Section 3. In particular, we discuss some implementation considerations and, among other things, analyse the possibility to delegate some parts of signature computation from the eIDAS token to the reader. In Section 4, we supply a security model for dynamic DSPS, and discuss some tricky points to formalize. Finally, we analyse in Section 5 the security of our construction and prove it secure in the random oracle model according to the security definitions of Section 4, before concluding in Section 6.

Notations. Given a non-empty finite set S, $s \leftarrow S$ means that s is drawn uniformly at random from S. Given a prime number p, we note $\mathbb{Z}_p := \{0, 1, \ldots, p-1\}$ and $\mathbb{Z}_p^* := \mathbb{Z}_p \setminus \{0\}$.

2 Definition of domain-specific pseudonymous signatures

Informally, a DSPS scheme is composed of procedures for generating parameters, keys (for group, users and domains), pseudonyms, generating and verifying signatures, and blacklisting users.

Definition 1 (DSPS) *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* {`Setup`, `GroupKeyGen`, `DomainKeyGen`, `Join`, `Issue`, `NymGen`, `Sign`, `Verify`, `Revoke`} *as described below.*

By convention, users are enumerated with index $i \in \mathbb{N}$ and domains with index $j \in \mathbb{N}$.

Global parameters. On input a security parameter λ, the `Setup` algorithm computes global parameters param. A message space $\mathcal{M}$ is specified. The sets of user $\mathcal{U}$ and domains $\mathcal{D}$ are initially empty.

$$\mathsf{param} \leftarrow \mathtt{Setup}(1^\lambda)$$

The global parameters param are implicitly given to all algorithms, if not explicitly specified.

Group creation. On input the global parameters param, the `GroupKeyGen` algorithm outputs an asymmetric key pair $(\mathsf{gpk}, \mathsf{gsk})$.

$$(\mathsf{gpk}, \mathsf{gsk}) \leftarrow \mathtt{GroupKeyGen}(\mathsf{param})$$

Domain creation. On input the global parameters param and a domain $j \in \mathcal{D}$, the `DomainKeyGen` algorithm outputs a public key dpk_j for j.

$$\mathsf{dpk}_j \leftarrow \mathtt{DomainKeyGen}(\mathsf{param}, j)$$

Together with the generation of a public key, an empty revocation list $\mathcal{L}_j$ associated to the domain j is created.

User enrolment. The enrolment protocol involves a user $i \in \mathcal{U}$ and the issuing authority IA. `Join` takes as input the global parameters param and a group public key gpk. `Issue` takes as input the global parameters param and a group secret key gsk. At the end of the protocol, the user i gets a secret key usk_i and the issuing authority IA keeps a revocation token rt_i.

$$\mathsf{usk}_i \leftarrow \mathtt{Join}(\mathsf{param}, \mathsf{gpk}, i) \leftrightarrow \mathtt{Issue}(\mathsf{param}, \mathsf{gpk}, i, \mathsf{gsk}) \rightarrow \mathsf{rt}_i$$

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

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

Generation of signatures. On input the global parameters param, a group public key 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 $m \in \mathcal{M}$, the `Sign` algorithm outputs a signature σ.

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

Interactive generation of signatures. Part of the signature generation might be delegated to an external entity. In this case, the `Sign` procedure is split into three parts `Delegate`, `PreCompute` and `Finalize`. On input the global parameters param, a group public key 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 $m \in \mathcal{M}$, the `Delegate` algorithm outputs a delegation information *del* and keeps some auxiliary

information aux. On input the delegation information del, the `PreCompute` algorithm outputs a pre-computation pre. On input the auxiliary information aux and the pre-computation pre, the `Finalize` algorithm outputs a signature σ.

$$del, aux \leftarrow \texttt{Delegate}(\mathsf{gpk}, \mathsf{dpk}_j, \mathsf{usk}_i, \mathsf{nym}_{ij}, m)$$
$$pre \leftarrow \texttt{PreCompute}(\mathsf{gpk}, del)$$
$$\sigma \leftarrow \texttt{Finalize}(\mathsf{gpk}, aux, pre)$$

Verification of the signatures. On input the global parameters param, a group public key 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 $\mathcal{L}_j$ of the domain j, the `Verify` algorithm outputs a decision $d \in \{0, 1\}$.

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

The fact that pseudonyms are deterministic implies the existence of an implicit algorithm for linking signatures. On input a domain public key dpk and two triples $(\mathsf{nym}, m, \sigma)$ and $(\mathsf{nym}', m', \sigma')$, the `Link` algorithm outputs 1 if $\mathsf{nym} = \mathsf{nym}'$ and outputs 0 otherwise.

Revocation. On input the global parameters param, a group public key gpk, a revocation token rt_i of a user $i \in \mathcal{U}$ and a public key dpk_j of a domain $j \in \mathcal{D}$, the `Revoke` algorithm outputs the pseudonym nym_{ij}.

$$\mathsf{nym}_{ij} \leftarrow \texttt{Revoke}(\mathsf{param}, \mathsf{gpk}, \mathsf{rt}_i, \mathsf{dpk}_j)$$

A global revocation protocol for revoking the user i everywhere is implicit here: it suffices for the issuer to publish rt_i. Then all pseudonyms will be computable from the past and future domains' public keys.

Correctness. As a first property to be satisfied by a DSPS scheme, honest and non-revoked users should be accepted (signature correctness) and the revocation of users prevents them to produce accepted signatures (revocation correctness). More formally, given global parameters param, a DSPS scheme is *correct* if for all $(\mathsf{gpk}, \mathsf{gsk}) \leftarrow \texttt{GroupKeyGen}(1^\lambda)$, all $\mathsf{usk} \leftarrow \texttt{Join}(\mathsf{gpk}) \leftrightarrow \texttt{Issue}(\mathsf{gpk}, \mathsf{gsk}) \rightarrow \mathsf{rt}$, all $\mathsf{dpk} \leftarrow \texttt{DomainKeyGen}(\mathsf{gpk})$, all $\mathsf{nym} \leftarrow \texttt{NymGen}(\mathsf{gpk}, \mathsf{dpk}, \mathsf{usk})$, all $m \in \mathcal{M}$, we have:

– *Signature correctness*:

1. for all $\sigma \leftarrow \texttt{Sign}(\mathsf{gpk}, \mathsf{dpk}, \mathsf{usk}, \mathsf{nym}, m)$, we have $\texttt{Verify}(\mathsf{gpk}, \mathsf{dpk}, \mathsf{nym}, m, \sigma, \{\}) = 1$.
2. for all $(del, aux) \leftarrow \texttt{Delegate}(\mathsf{gpk}, \mathsf{dpk}, \mathsf{usk}, \mathsf{nym}, m)$, $pre \leftarrow \texttt{PreCompute}(\mathsf{gpk}, del)$, $\sigma \leftarrow \texttt{Finalize}(\mathsf{gpk}, aux, pre)$, we have $\texttt{Verify}(\mathsf{gpk}, \mathsf{dpk}, \mathsf{nym}, m, \sigma, \{\}) = 1$.

– *Revocation correctness*:

1. $\mathsf{nym} = \mathsf{nym}'$ where $\mathsf{nym}' := \texttt{Revoke}(\mathsf{gpk}, \mathsf{rt}, \mathsf{dpk})$.

3 An efficient construction of dynamic DSPS

In this section, we present our construction for a dynamic DSPS scheme.

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 (here in multiplicative notation) and e is a bilinear and non-degenerate application $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. The bi-linearity property 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 non-degeneracy property 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 $\mathbb{G}_1 \neq \mathbb{G}_2$.

Global parameters. We assume that some bilinear groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ of prime order p equipped with a non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ and generators $G_1, H, U \in \mathbb{G}_1$, $G_2 \in \mathbb{G}_2$ are publicly available to all participants. We assume that this pairing is of Type III (according to the classification of [21]), meaning that (i) $\mathbb{G}_1 \neq \mathbb{G}_2$ and (ii) there is no efficiently computable isomorphism between them in both directions. In all the following, if not stated explicitly, all algorithms are implicitly given $\mathsf{param} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, H, U, G_2)$ as input. A hash function $\mathtt{H} : \{0,1\}^* \rightarrow \{0,1\}^\lambda$ is also given (modelled as a random oracle in the security proofs), where λ is the security parameter that manages the bilinear environment generation (we have $|p| \approx 2 \cdot \lambda$). Messages in our scheme are bit strings: $\mathcal{M} := \{0,1\}^*$.

Creation of the group. The `GroupKeyGen` algorithm draws $y \leftarrow \mathbb{Z}_p^*$ and returns the group secret key $\mathsf{gsk} := y$ and the group public key $\mathsf{gpk} := (Y_1, Y_2) := (H^y, (G_2)^y)$. The element Y_1 is only used when the signature process is interactive. If no procedure of delegation is considered in some concrete use of the scheme, this element can be dropped.

Creation of the domains. The `DomainKeyGen` algorithm draws $r \leftarrow \mathbb{Z}_p^*$, and returns the domain public key $\mathsf{dpk} := D := (G_1)^r$. Note that

the random r is not a private key associated to dpk, and may be discarded. dpk is just a public uniform random element in $\mathbb{G}_1$ that identifies a given domain. It might be for instance the hash of the domain's data (the hash function is then considered as a random oracle in the security analysis). We assume that these domain keys are honestly computed.

Enrolment of the users. Given a group secret key $\mathsf{gsk} = y$, a user secret key usk is a triple (f, A, x) where $A := (U \cdot H^f)^{\frac{1}{x+y}}$ for $x \leftarrow \mathbb{Z}_p \backslash \{y\}$ and $f \leftarrow \mathbb{Z}_p^*$. The corresponding revocation token is $\mathsf{rt} := (F, x)$ where $F := H^f$. The following protocol allows the user to obtain a secret key from the issuer (this procedure closely follows the one from [18]).

1. [user] draw $f' \leftarrow \mathbb{Z}_p^*$; set $C := H^{f'}$; compute Π
2. [user] send C, Π
3. [issuer] receive C, Π; check Π
4. [issuer] draw $x, f'' \leftarrow \mathbb{Z}_p$
5. [issuer] set $A := (U \cdot C \cdot H^{f''})^{\frac{1}{y+x}}$
6. [issuer] send (f'', A, x)
7. [issuer] store $\mathsf{rt} := (C \cdot H^{f''}, x)$
8. [user] receive (f'', A, x)
9. [user] check whether $e(A, (G_2)^x \cdot Y_2) = e(U \cdot H^{f+f'}, G_2)$
10. [user] store $\mathsf{usk} := (f, A, x)$ where $f := f' + f''$

The proof Π is computed as

$$\Pi := \mathsf{PoK}\big\{\langle f' \rangle : E = \texttt{Ext-Commit}(f') \wedge \texttt{NIZKPEqDL}(f', E, C, H)\big\},$$

where `Ext-Commit` is an extractable commitment scheme and `NIZKPEqDL` $(f,\ E,\ C,\ H)$ denotes a Non Interactive Zero Knowledge Proof of Equality of the Discrete Logarithm f of C in basis H with the value committed in E.

Generation of the pseudonyms. The `NymGen` algorithm takes a domain public key $\mathsf{dpk} = D$ and a user secret key $\mathsf{usk} = (f, A, x)$ as input, and outputs the pseudonym of the user relatively to domain D defined as $\mathsf{nym} := N := H^f \cdot D^x$.

Generation of the signatures. The `Sign` algorithm takes as input a group public key $\mathsf{gpk} = (Y_1, Y_2)$, a domain public key $\mathsf{dpk} = D$, a user secret key $\mathsf{usk} = (f, A, x)$, a pseudonym $\mathsf{nym} = N$, and a message m, then performs the following steps.

1. draw $a, r_f, r_x, r_a, r_b, r_d \leftarrow \mathbb{Z}_p^*$
2. set $T := A \cdot H^a$
3. set $R_1 := H^{r_f} \cdot D^{r_x}$
4. set $R_2 := N^{r_a} \cdot H^{-r_d} \cdot D^{-r_b}$
5. set $R_3 := e(T^{r_x} \cdot H^{-r_f - r_b} \cdot (Y_1)^{-r_a}, G_2)$
6. set $c := \mathtt{H}(D, N, T, R_1, R_2, R_3, m)$
7. set $s_f := r_f + c \cdot f \mod p$,
   $s_x := r_x + c \cdot x \mod p$,
   $s_a := r_a + c \cdot a \mod p$,
   $s_b := r_b + c \cdot a \cdot x \mod p$, and
   $s_d := r_d + c \cdot a \cdot f \mod p$
8. return the signature $\sigma := (T, c, \vec{s})$ where $\vec{s} := (s_x, s_f, s_a, s_b, s_d)$

A proof of knowledge of a valid certificate. Following a classical way to construct signatures, the signature of our scheme is obtained by applying the Fiat-Shamir heuristic [20] to a Σ-protocol [17] for proving knowledge of a valid user certificate. For the sake of completeness, we explicitly give this proof of knowledge in Appendix B.

Signature size. A pseudonym is a group element and a signature $\sigma := (T, c, \vec{s})$ 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 [12] lies in ${\mathbb{G}_1}^4 \times \{0,1\}^\lambda \times {\mathbb{Z}_p}^4$. A short group signature of [18] lies 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.

Signature generation with pre-computations. In step 5 of the signature generation above, a pairing is computed. The user might also pre-compute an element $e(A, G_2)$ from its secret key, and elements $e(H, G_2)$, $e(H, Y_2)$ from the group public key. The R_3 element is then computed as $R_3 :=$

$$e(A, G_2)^{r_x} \cdot e(H, G_2)^{a \cdot r_x - r_f - r_b} \cdot e(H, Y_2)^{-r_a}.$$

As a result, a pairing is no longer necessary, at a cost of a multi-exponentiation in $\mathbb{G}_T$.

Interactive process for the generation of the signatures. When part of the signature generation is delegated, the user takes as input a group public key $\mathsf{gpk} = (Y_1, Y_2)$, a domain public key $\mathsf{dpk} = D$, a user secret key $\mathsf{usk} = (f, A, x)$, a pseudonym $\mathsf{nym} = N$, and a message m, then engages the following protocol.

`Delegate`

1. [user] draw $a, r_f, r_x, r_a, r_b \leftarrow \mathbb{Z}_p$
2. [user] compute $T := A \cdot H^a$
3. [user] compute $B := T^{r_x} \cdot H^{-r_f - r_b} \cdot (Y_1)^{-r_a}$
4. [user] send B

`PreCompute`

5. [delegatee] receive B
6. [delegatee] compute $R_3 := e(B, G_2)$
7. [delegatee] send R_3

`Finalize`

8. [user] receive R_3
9. [user] draw $r_d \leftarrow \mathbb{Z}_p$
10. [user] set $R_1 := H^{r_f} \cdot D^{r_x}$
11. [user] set $R_2 := N^{r_a} \cdot H^{-r_d} \cdot D^{-r_b}$
12. [user] set $c := \mathtt{H}(D, N, T, R_1, R_2, R_3, m)$
13. [user] set $s_f := r_f + c \cdot f \mod p$,
    $s_x := r_x + c \cdot x \mod p$,
    $s_a := r_a + c \cdot a \mod p$,
    $s_b := r_b + c \cdot a \cdot x \mod p$, and
    $s_d := r_d + c \cdot a \cdot f \mod p$
14. [user] return the signature $\sigma := (T, c, \vec{s})$ where $\vec{s} := (s_x, s_f, s_a, s_b, s_d)$

In the protocol above, the user only performs scalar operations and exponentiations in the group $\mathbb{G}_1$. As a result, there is no need to implement pairing operations or exponentiations in $\mathbb{G}_T$ in the user side. This brings a piece of valuable advantages for a deployment with current technology. We implemented our protocol on a PC, using Barreto-Naehrig curves [1] at 128 bits of security. 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.

Verification of the signatures. A verifier takes as input a group public key $\mathsf{gpk} = (Y_1, Y_2)$, a domain public key $\mathsf{dpk} = D$, a pseudonym $\mathsf{nym} = N$, a message m, a signature $\sigma = (T, c, \vec{s})$ where $\vec{s} := (s_x, s_f, s_a, s_b, s_d)$, and a list $\mathcal{L}$, and proceeds as follows:

1. if $N \in \mathcal{L}$, return 0
2. compute ${R_1}' := H^{s_f} \cdot D^{s_x} \cdot N^{-c}$
3. compute ${R_2}' := N^{s_a} \cdot H^{-s_d} \cdot D^{-s_b}$
4. compute ${R_3}' := e(T^{s_x} \cdot H^{-s_f - s_b} \cdot U^{-c}, G_2) \cdot e(H^{-s_a} \cdot T^c, Y_2)$
5. compute $c' := \mathtt{H}(D, N, T, {R_1}', {R_2}', {R_3}', m)$
6. if $c' = c$ return 1, otherwise return 0

Correctness of the verification. One might check that a valid signature is always accepted by the verification algorithm. Indeed, in that case we have:

$$
\begin{aligned}
{R_1}' &= H^{r_f + c \cdot f} \cdot D^{r_x + c \cdot x} \cdot H^{-c \cdot f} \cdot D^{-c \cdot x} \\
&= H^{r_f} \cdot D^{r_x} = R_1 \\
{R_2}' &= H^{f \cdot (r_a + c \cdot a)} \cdot D^{x \cdot (r_a + c \cdot a)} \cdot H^{-r_d - c \cdot a \cdot f} \\
&\qquad \cdot D^{-r_b - c \cdot a \cdot x} \\
&= H^{f \cdot r_a} \cdot D^{x \cdot r_a} \cdot H^{-r_d} \cdot D^{-r_b} = R_2 \\
{R_3}' &= e(T^{r_x + c \cdot x} \cdot H^{-r_f - c \cdot f - r_b - c \cdot a \cdot x} \cdot U^{-c}, G_2) \\
&\qquad \cdot e(H^{-r_a - c \cdot a} \cdot T^c, Y_2) \\
&= e(T^{r_x} \cdot H^{-r_f - r_b}, G_2) \cdot e(H^{-r_a}, Y_2) \\
&\qquad \cdot e(T^{c \cdot x} \cdot H^{-c \cdot (f - a \cdot x)} \cdot U^{-c}, G_2) \\
&\qquad \cdot e(H^{-c \cdot a} \cdot T^c, Y_2) \\
&= e(T^{r_x} \cdot H^{-r_f - r_b} \cdot (Y_1)^{-r_a}, G_2) \\
&\qquad \cdot e(T^{c \cdot x} \cdot H^{-c \cdot (f - a \cdot x)} \cdot U^{-c} \cdot A^{c \cdot y}, G_2) \\
&= R_3 \cdot e(A^{c \cdot (x + y)} \cdot (H^f \cdot U)^{-c}, G_2) \\
&= R_3 \cdot e((H^f \cdot U)^c \cdot (H^f \cdot U)^{-c}, G_2) = R_3
\end{aligned}
$$

Revocation. Given a revocation token $\mathsf{rt} = (F, x)$ of some user and a domain key $\mathsf{dpk} = D$, the revocation procedure returns the pseudonym $\mathsf{nym} := F \cdot D^x$. The domain updates its revocation list $\mathcal{L} := \mathcal{L} \cup \{\mathsf{nym}\}$.

Efficiency of the revocation in DSPS. Since a revocation list is a set of revoked pseudonyms in DSPS, the revocation test is a simple membership test. In practice, this can be done very

efficiently. It is far more efficient than revocation in the VLR group signature constructions. For instance, in [9], two pairings have to be computed for each token in the revocation list.

4 DSPS Security properties

Besides correctness, a DSPS scheme should satisfy the *cross-domain anonymity*, *seclusiveness* and *unforgeability* properties to be secure. Informally, a DSPS scheme is *cross-domain anonymous* if signatures are unlinkable but within a specific domain, *seclusive* if it is impossible to exhibit a valid signature without involving a single existing user, and *unforgeable* if nobody, including corrupted authority and domains owners, cannot sign on behalf of an honest user. The remaining of this section is dedicated to the formalization of these notions.

4.1 Game-based security proofs

Security properties are modelled by games between a challenger and an adversary. Given an adversary $\mathtt{A}$, a game G and a bit $b \in \{0,1\}$, we denote $\mathtt{A}^{\mathsf{G}} \Rightarrow b$ the fact that the challenger playing the game G outputs b after its interaction with $\mathtt{A}$. The following fundamental result about game-based security proofs is given in [4].

Lemma 2 ([4]) *Let* G *and* G' *be two games, identical until some internal flag* bad *is set to* true *in game* G. *Then*

$$\left|\Pr\left[\mathtt{A}^{\mathsf{G}} \Rightarrow 1\right] - \Pr\left[\mathtt{A}^{\mathsf{G}'} \Rightarrow 1\right]\right| \leq \Pr\left[\mathtt{A}^{\mathsf{G}} : \mathsf{bad} = \mathsf{true}\right].$$

4.2 Oracles

The global variables used by the challenger during a game are summarized in Table 1. Oracles may be available to the adversary. Below is given an informal description of each oracle. Complete definitions are given Figures 4 and 5 in Appendix A.

adding users and domains (U, D). A may dynamically add users and domains, through the U and D oracles.

corrupting users (C). A may corrupt users, with a query to C. The challenger returns the user's secret key (as well as the pseudonyms, but it may compute the pseudonyms with the user's key).

$\mathcal{U}$	set of honest users
$\mathcal{C}$	set of corrupted users
$\mathcal{D}$	set of domains
$\mathcal{L}$	domains' revocation lists
$\mathcal{S}$	set of messages signed by the challenger
$\mathcal{K}$	lists of leaked pseudonyms
$\mathcal{N}$	set of (user, domain) pairs the adversary may know the pseudonym
dpk	table of domains' public keys
usk	table of users' secret keys
nym	table of the pseudonyms
rt	table of revocation tokens
ro	table for the random oracle

Table 1: Variables managed by the challenger

revoking users (R). A may ask for the revocation of some users near some domains.

learning identities (N, I). A may ask for the pseudonyms corresponding to a given (user, domain) pair, with the oracle N. Given a pseudonym and a domain, A may also ask for the identity to which this pseudonym belongs within the domain, thanks to the oracle I.

pseudonyms leakage (L). A may collect pseudonyms without knowing the underlying identities.

signatures (S, DelS, V). A may ask for signatures on behalf of a given pseudonym for messages of its choice. Moreover, when part of the signature is delegated by a delegator to a delegatee, it can interacts with an honest delegator as a corrupted delegatee.

Since the verification of signatures involves revocation lists, A is given an oracle V for their verification.

Since the signatures are created and verified under the pseudonyms, the adversary does not ask for signatures on behalf of users. If the adversary wants a signature from a particular user, it may first ask for this user's pseudonym, then asks the signature oracle for a signature.

interactive issuing (StoI, StoU). A may interact as a corrupted user in front of a honest issuer (oracle StoI – Send to Issuer) or as a corrupted issuer in front of a honest user (oracle StoU – Send to User).

In the random oracle model, the adversary has access to a random oracle $\mathtt{H}_{\mathcal{R}}$ for values in the set $\mathcal{R}$ in addition to the oracles above; the set $\mathcal{R}$ being specified by the parameters of the scheme.

challenge oracle $\texttt{LoR}(b_0, b_1, i_0, i_1, j_0, j_1)$, game variables $\mathcal{U}, \mathcal{D}, \mathcal{N}, \mathcal{K}, \mathsf{nym}$:

1. if $i_0 \notin \mathcal{U}$ or $i_1 \notin \mathcal{U}$ or $j_0 \notin \mathcal{D}$ or $j_1 \notin \mathcal{D}$, return $\perp$
2. if $|\{j \in \{j_0, j_1\} : (i_0, j) \in \mathcal{N} \text{ or } (i_1, j) \in \mathcal{N}\}| > 1$, return $\perp$
   // step 2: there must be at most one domain where the users may be identified
3. if ($(\mathcal{K}[j_0] \cup \mathcal{K}[j_1]) \cap \{i_0, i_1\} \neq \emptyset$) and ($\exists j \in \{j_0, j_1\}$: $\{i_0, i_1\} \not\subseteq \mathcal{K}[j]$), return $\perp$
   // step 3: leakage of pseudonyms together with dynamic creation of users and domains should not help to trivially win
4. return $\{(j_0, \mathsf{nym}[i_{b_0}][j_0]), (j_1, \mathsf{nym}[i_{b_1}][j_1])\}$

cross-domain anonymity experiment $\texttt{Experiment}^{\mathsf{CDA}}_{\mathsf{param}}(\texttt{A})$

1. **initialisation**: $\texttt{InitializeGame}()$; $(\mathsf{gpk}, \mathsf{gsk}) \leftarrow \texttt{GroupKeyGen}(\mathsf{param})$; $b_0 \leftarrow \{0,1\}$; $b_1 \leftarrow \{0,1\}$;
2. **oracles**: $\mathcal{O} \leftarrow \{\texttt{U}(\mathsf{gsk}, \cdot), \texttt{D}(\cdot), \texttt{C}(\cdot), \texttt{R}(\cdot,\cdot), \texttt{N}(\cdot,\cdot), \texttt{S}_{\mathcal{M}}(\cdot,\cdot,\cdot), \texttt{DelS}_{\mathcal{M}}(\cdot,\cdot), \texttt{I}(\cdot,\cdot), \texttt{L}(\cdot), \texttt{H}_{\mathcal{R}}(\cdot), \texttt{V}_{\mathcal{M}}(\cdot,\cdot,\cdot,\cdot), \texttt{LoR}(b_0, b_1, \cdot, \cdot, \cdot, \cdot)\}$
3. **adversary phase**: $d \leftarrow \texttt{A}^{\mathcal{O}}(\mathsf{param}, \mathsf{gpk})$
4. **winning condition**: return 1 if $(d == (b_0 == b_1))$, otherwise return 0

Figure 1: The cross-domain anonymity experiment

4.3 Cross-domain anonymity

Figure 1 describes the cross-domain anonymity experiment. Informally, an adversary should not be able to link users across domains. This property is formalized by a challenge oracle: given two domains, an adversary should not be able to tell whether two pseudonyms belong to the same user or not, even if it knows the underlying user for one out of the two domains. Two bits are picked by the challenger, one for each domain queried to the challenge oracle. At the end of the experiment, the adversary $\texttt{A}$ returns a bit d and the game returns 1 if $(d == (b_0 == b_1))$ (where $(b_0 == b_1)$ equals 1 if the assertion is true and 0 otherwise). We denote $\mathbf{Adv}^{\mathsf{CDA}}_{\mathsf{param}}(\texttt{A})$ the advantage of the adversary, defined as:

$$\mathbf{Adv}^{\mathsf{CDA}}_{\mathsf{param}}(\texttt{A}) \stackrel{\text{def}}{=} \left| 2 \cdot \Pr\left[\texttt{A}^{\texttt{Experiment}^{\mathsf{CDA}}_{\mathsf{param}}} \Rightarrow 1\right] - 1 \right|.$$

Given a security parameter $\lambda \in \mathbb{N}$, parameters $\mathsf{param} \leftarrow \texttt{Setup}(1^\lambda)$ and a polynomial t, we say that a DSPS scheme achieves *cross-domain-anonymity* if, for all $\texttt{A}$ running in time less than $t(\lambda)$, the quantity $\mathbf{Adv}^{\mathsf{CDA}}_{\mathsf{param}}(\texttt{A})$ is negligible as a function of λ.

Corruption of the issuer. Contrary to group signatures (see [5]), the issuing authority IA is not corrupted in this game. This assumption is minimal since the IA may trace all honest users. Hence we give the adversary the ability to interact as a corrupted user with the honest issuer, through the $\texttt{StoI}$ oracle.

Discussion about anonymity. 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, add a user i_2, and win a challenge involving i_0, i_2 with non-negligible probability. To avoid such an attack, the challenger maintains a list of the (user, domain) pairs, for which the pseudonym might be known 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 $\texttt{LoR}$ oracle contain enough uncertainty for at least one domain. Note that the uncertainty is required for only one domain. A user queried to the $\texttt{LoR}$ might be known or revoked in a domain: the adversary has to guess whether the other pseudonym belongs to the same user.

Cross-domain anonymity w.r.t. several challenge calls. Let us assume that the standard notion of public key encryption and its semantic security IND-CPA (aka INDistinguishability under Chosen Plaintext Attack) is known to the reader. A popular argument says that if a public key encryption scheme is ϵ-IND-CPA secure with a single call to the challenge oracle, then it is $\ell \cdot \epsilon$-IND-CPA secure with ℓ calls to the challenge oracle. This argument does not work in the case of DSPS. However, we can still manage to show that if a DSPS scheme is CDA secure with a single call to the challenge oracle, then it is still CDA secure with a polynomial call to the

seclusiveness experiment $\mathtt{Experiment}^{\mathsf{SEC}}_{\mathsf{param}}(\mathtt{A})$

1. **initialisation**: $\mathtt{InitializeGame}()$; $(\mathsf{gpk}, \mathsf{gsk}) \leftarrow \mathtt{GroupKeyGen}(\mathsf{param})$;
2. **oracles**: $\mathcal{O} \leftarrow \{\mathtt{U}(\mathsf{gsk}, \cdot), \mathtt{StoI}(\mathsf{gsk}, \cdot, \cdot), \mathtt{D}(\cdot), \mathtt{C}(\cdot), \mathtt{R}(\cdot, \cdot), \mathtt{N}(\cdot, \cdot), \mathtt{I}(\cdot, \cdot), \mathtt{L}(\cdot), \mathtt{H}_{\mathcal{R}}(\cdot), \mathtt{S}_{\mathcal{M}}(\cdot, \cdot, \cdot), \mathtt{DelS}_{\mathcal{M}}(\cdot, \cdot), \mathtt{V}_{\mathcal{M}}(\cdot, \cdot, \cdot, \cdot)\}$
3. **adversary phase**: $(j^*, N^*, m^*, \sigma^*) \leftarrow \mathtt{A}^{\mathcal{O}}(\mathsf{param}, \mathsf{gpk})$
4. **winning condition**:
   (a) if $j^* \notin \mathcal{D}$, return 0
   (b) if $\mathtt{Verify}(\mathsf{gpk}, \mathsf{dpk}[j^*], N^*, m^*, \sigma^*, \{\mathsf{nym}[i][j^*]\}_{i \in \mathcal{U} \cup \mathcal{C}}) = 1$, return 1
   (c) return 0

Figure 2: The seclusiveness experiment

challenge oracle. A proof of this fact is given in Appendix C.

4.4 Seclusiveness

The Figure 2 describes the seclusiveness experiment. Informally, only valid enrolled users should be able to produce valid signatures. This property is similar with the traceability property of the group signatures [2, 5], where an adversary should be 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 properly define *tracing users*. Nevertheless, the management of the revocation tokens allows to correctly phrase the winning condition, as in VLR group signatures [9], providing that we take into account the presence of the pseudonyms. If at the end of the game, the challenger blacklists all the users. If the signature is valid, then the adversary has won the game, as an analogue of "the opener cannot conclude" in the group signature case. We denote $\mathbf{Adv}^{\mathsf{SEC}}_{\mathsf{param}}(\mathtt{A})$ the success probability of the adversary in this experiment:

$$\mathbf{Adv}^{\mathsf{SEC}}_{\mathsf{param}}(\mathtt{A}) \stackrel{\text{def}}{=} \Pr\left[\mathtt{A}^{\mathtt{Experiment}^{\mathsf{SEC}}_{\mathsf{param}}} \Rightarrow 1\right].$$

Given a security parameter $\lambda \in \mathbb{N}$, parameters $\mathsf{param} \leftarrow \mathtt{Setup}(1^\lambda)$ and a polynomial t, we say that a DSPS scheme achieves *seclusiveness* if, for all $\mathtt{A}$ running in time less than $t(\lambda)$, the quantity $\mathbf{Adv}^{\mathsf{SEC}}_{\mathsf{param}}(\mathtt{A})$ is negligible as a function of λ.

4.5 Unforgeability

The Figure 3 describes the unforgeability experiment. Nobody, including the issuer, should be able to produce a valid signature in the name of a honest user. We denote $\mathbf{Adv}^{\mathsf{UF}}_{\mathsf{param}}(\mathtt{A})$ the success probability of the adversary in this forgery experiment:

$$\mathbf{Adv}^{\mathsf{UF}}_{\mathsf{param}}(\mathtt{A}) \stackrel{\text{def}}{=} \Pr\left[\mathtt{A}^{\mathtt{Experiment}^{\mathsf{UF}}_{\mathsf{param}}} \Rightarrow 1\right].$$

Given a security parameter $\lambda \in \mathbb{N}$, parameters $\mathsf{param} \leftarrow \mathtt{Setup}(1^\lambda)$ and a polynomial t, we say that a DSPS scheme achieves *unforgeability* if, for all $\mathtt{A}$ running in time less than $t(\lambda)$, the quantity $\mathbf{Adv}^{\mathsf{UF}}_{\mathsf{param}}(\mathtt{A})$ is negligible as a function of λ.

4.6 Comparisons with precedent models

For sake of clarity, note that $(\mathsf{nym}_i, \mathsf{dsnym}_{ij})$ in [6] maps to (i, nym_{ij}) in our model.

The model of [6]. First of all, the model of [6] is static. All users and domains are created at the beginning of the games, while our security games are all dynamic. Then, 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 their correctness using the 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. Finally, the model of [6] does not

unforgeability experiment $\mathtt{Experiment}^{\mathsf{UF}}_{\mathsf{param}}(\mathtt{A})$

1. **initialisation**: $\mathtt{InitializeGame}()$ $(\mathsf{gpk}, \mathsf{gsk}) \leftarrow \mathtt{GroupKeyGen}(\mathsf{param})$;
2. **oracles**: $\mathcal{O} \leftarrow \{\mathtt{StoU}(\cdot), \mathtt{D}(\cdot), \mathtt{C}(\cdot), \mathtt{R}(\cdot,\cdot), \mathtt{N}(\cdot,\cdot), \mathtt{I}(\cdot,\cdot), \mathtt{L}(\cdot), \mathtt{H}_{\mathcal{R}}(\cdot), \mathtt{S}_{\mathcal{M}}(\cdot,\cdot,\cdot), \mathtt{DelS}_{\mathcal{M}}(\cdot,\cdot), \mathtt{V}_{\mathcal{M}}(\cdot,\cdot,\cdot,\cdot)\}$
3. **adversary phase**: $(i^*, j^*, m^*, \sigma^*) \leftarrow \mathtt{A}^{\mathcal{O}}(\mathsf{param}, \mathsf{gpk}, \mathsf{gsk})$
4. **winning condition**:
   (a) if $i^* \notin \mathcal{U}$ or $j^* \notin \mathcal{D}$ or $(i^*, j^*, m^*, \sigma^*) \in \mathcal{S}$, return 0
   (b) if $\mathtt{Verify}(\mathsf{gpk}, \mathsf{dpk}[j^*], \mathsf{nym}[i^*][j^*], m^*, \sigma^*, \{\}) = 1$, return 1
   (c) return 0

Figure 3: The unforgeability experiment

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).

The model of [12]. The model of [12] 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 [6]). This model 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.

5 Security of our construction

This section begins by stating several hard problems in prime order and bilinear groups. Then the construction from Section 3 is proven secure according to the model of Section 4 under these hard problems.

5.1 Hard problems

Discrete logarithm problem. Let $\mathbb{G}$ be a cyclic group of prime order p and G a generator of $\mathbb{G}$. Given an adversary $\mathtt{A}$, the success probability (or advantage) of $\mathtt{A}$ against the Discrete Logarithm (DL) problem is defined as

$$\mathbf{Adv}^{\mathsf{DL}}_{\mathbb{G}}(\mathtt{A}) \stackrel{\text{def}}{=} \Pr\left[a \leftarrow \mathbb{Z}_p : \mathtt{A}(G^a) \Rightarrow a\right].$$

We define $\mathbf{Adv}^{\mathsf{DL}}_{\mathbb{G}}(t) = \max_{\mathtt{A}}\{\mathbf{Adv}^{\mathsf{DL}}_{\mathbb{G}}(\mathtt{A})\}$, where the maximum is taken over all adversaries running in time at most t.

Decisional Diffie-Hellman problem [19]. Let $\mathbb{G}$ be a cyclic group of prime order p and G a generator of $\mathbb{G}$. Given an adversary $\mathtt{A}$, the success probability (or advantage) of $\mathtt{A}$ against the Decisional Diffie-Hellman (DDH) problem is defined as

$$\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}}(\mathtt{A}) \stackrel{\text{def}}{=} \Big|\Pr\left[a, b \leftarrow \mathbb{Z}_p : \mathtt{A}(G^a, G^b, G^{a\cdot b}) \Rightarrow 1\right] - \Pr\left[a, b, c \leftarrow \mathbb{Z}_p : \mathtt{A}(G^a, G^b, G^c) \Rightarrow 1\right]\Big|.$$

We define $\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}}(t) = \max_{\mathtt{A}}\{\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}}(\mathtt{A})\}$, where the maximum is taken over all adversaries running in time at most t.

Strong Diffie-Hellman problem [8]. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be three cyclic groups of prime order p, H_1 (respectively H_2) a generator of $\mathbb{G}_1$ (respectively $\mathbb{G}_2$), and e a non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Let $q \geq 1$. Given an adversary $\mathtt{A}$, the success probability (or advantage) of $\mathtt{A}$ against the Strong Diffie-Hellman (SDH) problem is defined as

$$\mathbf{Adv}^{q\text{-}\mathsf{SDH}}_{\mathbb{G}_1,\mathbb{G}_2}(\mathtt{A}) \stackrel{\text{def}}{=} \Pr\Big[a \leftarrow \mathbb{Z}_p^* : \mathtt{A}((H_1)^a, (H_2)^a, (H_1)^{(a^2)}, \ldots, (H_1)^{(a^q)}) \Rightarrow (c, (H_1)^{1/(a+c)}) \in \mathbb{Z}_p \setminus \{-a\} \times \mathbb{G}_1\Big].$$

We define $\mathbf{Adv}^{q\text{-}\mathsf{SDH}}_{\mathbb{G}_1,\mathbb{G}_2}(t) = \max_{\mathtt{A}}\{\mathbf{Adv}^{q\text{-}\mathsf{SDH}}_{\mathbb{G}_1,\mathbb{G}_2}(\mathtt{A})\}$, where the maximum is taken over all adversaries running in time at most t.

5.2 Cross-domain anonymity

The cross-domain anonymity of our scheme is proved by reduction to the decisional Diffie-

Hellman problem.

Theorem 3 *Let $\mathtt{A}$ be an adversary against the cross-domain anonymity of the pseudonymous signature scheme, running in time at most t, making at most q_h random oracle queries, at most q_s signature queries, adding q_u users and q_d domains. Then*

$$\mathbf{Adv}^{\mathsf{CDA}}_{\mathbb{G}_1,\mathbb{G}_2}(\mathtt{A}) \leq \frac{q_u \cdot q_d}{2} \cdot \left(\frac{q_s \cdot (q_s + q_h) + 1}{p} + \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(t) \right).$$

Proof. We proceed with a sequence of games.

Game 0. Game G_0 is $\mathtt{Experiment}^{\mathsf{CDA}}_{\mathsf{param}}(\mathtt{A})$ as given Figure 1, with $\mathcal{R} := \mathbb{Z}_p$ and $\mathcal{M} := \{0,1\}^*$, and where, in addition, G_0 uniformly guesses a user $\mathsf{i} \in [1, q_u]$ and a domain $\mathsf{j} \in [1, q_d]$, and hopes that i and j are asked to the challenge oracle. Since no information about i and j is available from the adversary's point of view, this happens with probability $4/(q_u \cdot q_d)$. So we have:

$$\Pr\left[\mathtt{A}^{\mathsf{G}_0} \Rightarrow 1\right] = \frac{\frac{4}{q_u \cdot q_d} \cdot \mathbf{Adv}^{\mathsf{CDA}}_{\mathbb{G}_1,\mathbb{G}_2}(\mathtt{A}) + 1}{2} \quad (1)$$

Game 1. In game G_1, we change the way the signature queries are handled. On input a pseudonym N, a domain key D, a user key (f, A, x), a message m, game G_1 produces a signature as follows. y is the group secret key. In the first phase, the following steps are carried out:

1. finds i and j such that $N = N_{i:j}$
2. draw $T \leftarrow \mathbb{G}_1$; $c \leftarrow \{0,1\}^\lambda$; $s_f, s_x, s_a, s_b \leftarrow \mathbb{Z}_p$
3. send $B := T^{s_x + y \cdot c} \cdot U^{-c} \cdot H^{-y \cdot s_a - s_f - s_b}$

In the second phase:

1. receive R_3
2. draw $s_d \leftarrow \mathbb{Z}_p$ and set $R_1 := H^{s_f} \cdot D^{s_x} \cdot N^{-c}$ and $R_2 := N^{s_a} \cdot H^{-s_d} \cdot D^{-s_b}$
3. if $\mathtt{H}(D, N, T, R_1, R_2, R_3, m) \neq \perp$, set bad to true
4. program $\mathtt{H}(D, N, T, R_1, R_2, R_3, m) := c$
5. return $\sigma := (T, c, s_x, s_f, s_a, s_b, s_d)$

Here bad is a flag, initially set to false. Games G_0 and G_1 are identical until bad is set to true. In particular, there exists a such that $T = A \cdot H^a$, and we have:

$$\begin{aligned} B &= T^{s_x + y \cdot c} \cdot U^{-c} \cdot H^{-y \cdot s_a - s_f - s_b} \\ &= T^{r_x + c \cdot x + y \cdot c} \cdot U^{-c} \\ &\quad \cdot H^{-y \cdot r_a - y \cdot a \cdot c - r_f - c \cdot f - r_b - c \cdot a \cdot x} \\ &= T^{r_x} \cdot A^{c \cdot x + y \cdot c} \cdot U^{-c} \\ &\quad \cdot H^{a \cdot c \cdot x + a \cdot y \cdot c - y \cdot r_a - y \cdot a \cdot c - r_f - c \cdot f - r_b - c \cdot a \cdot x} \\ &= A^{c \cdot (y+x)} \cdot U^{-c} \cdot H^{-c \cdot f} \cdot T^{r_x} \cdot H^{-r_f - r_b} \\ &\quad \cdot H^{-y \cdot r_a} \\ &= \left(U \cdot H^f \cdot U^{-1} \cdot H^{-f}\right)^c \\ &\quad \cdot T^{r_x} \cdot H^{-r_f - r_b} \cdot (Y_1)^{-r_a} \\ &= T^{r_x} \cdot H^{-r_f - r_b} \cdot (Y_1)^{-r_a}, \end{aligned}$$

as required. Moreover, the simulation remains valid even if the adversary gives an incorrect R_3. By Lemma 2, we have that:

$$\begin{aligned} &\Pr\left[\mathtt{A}^{\mathsf{G}_0} \Rightarrow 1\right] - \Pr\left[\mathtt{A}^{\mathsf{G}_1} \Rightarrow 1\right] \\ &\leq \Pr\left[\mathtt{A}^{\mathsf{G}_1} : \mathsf{bad} = \mathsf{true}\right] \leq \frac{q_s \cdot (q_s + q_h)}{p} \end{aligned} \quad (2)$$

where the last inequality follows from the fact for each signature query, $\mathtt{H}$ has been previously defined on at most $q_s + q_h$ points, so that the probability that bad is set to true is at most $(q_s + q_h)/p$. This is because R_2 is uniformly random in $\mathbb{G}_1$ and independent from R_3, whatever the adversary's choice for R_3 is. The important point here is that s_d may be drawn *after* the reception of R_3.

Game 2. The difference between game G_2 and game G_1 is in the way the parameters, the group public key and the user secret keys are computed. The game draws $G_1, F \leftarrow \mathbb{G}_1 \setminus \{\mathbf{1}_{\mathbb{G}_1}\}$; $G_2 \leftarrow \mathbb{G}_2 \setminus \{\mathbf{1}_{\mathbb{G}_2}\}$; $\beta, y, f_1, \ldots f_{q_u} \leftarrow \mathbb{Z}_p^*$, $x_1, \ldots, x_{q_u} \leftarrow \mathbb{Z}_p \setminus \{y\}$, and sets $U := F^{(y+x_1)\cdots(y+x_{q_u})}$, $H := U^\beta$, $Y_1 := H^y$, $Y_2 := (G_2)^y$ and $A_i := F^{(1+\beta \cdot f_i) \cdot \prod_{n \in [1,q_u] \setminus \{i\}} (y+x_n)}$ for all $i \in [1, q_u]$. G_1, H, U and G_2 are distributed as in Game G_1 (unless $\prod_{i \in [1,q_u]} (y + x_i) = 0 \bmod p$, in which case the flag bad is set to true). The keys are also identically distributed. For instance, for user i, one can check that:

$$\begin{aligned} (A_i)^{y+x_i} &= F^{(y+x_1)\cdots(y+x_{q_u}) \cdot (1+\beta \cdot f_i)} \\ &= U^{1+\beta \cdot f_i} = U \cdot H^{f_i}. \end{aligned}$$

Hence, G_1 and G_2 are identical until bad is set to true:

$$\Pr\left[\mathtt{A}^{\mathsf{G}_1} \Rightarrow 1\right] - \Pr\left[\mathtt{A}^{\mathsf{G}_2} \Rightarrow 1\right] \leq \frac{1}{p}. \quad (3)$$

Game 3. $\mathsf{G}_3(G_1, \mathsf{A}, \mathsf{B}, \mathsf{C})$ takes as input a generator G_1 and three group elements $\mathsf{A} = (G_1)^{\mathsf{a}}$, $\mathsf{B} = (G_1)^{\mathsf{b}}$ and $\mathsf{C} = (G_1)^{\mathsf{a}\cdot\mathsf{b}}$ such that a and b are uniform. The difference with game G_2 is in the way the pseudonyms and domain keys are computed. Game G_3 computes the keys and the pseudonyms such that $x_{\mathsf{i}} = \mathsf{a}$ and $\mathsf{dpk}_{\mathsf{j}} = \mathsf{B}$. More precisely, the game draws $G_2 \leftarrow \mathbb{G}_2 \setminus \{\mathbf{1}_{\mathbb{G}_2}\}$; $\alpha, \beta, y, f_1, \ldots f_{q_u} \leftarrow \mathbb{Z}_p^*$, $x_i \leftarrow \mathbb{Z}_p \setminus \{y\}$, and sets:

- $U := (\mathsf{A} \cdot (G_1)^y)^{\alpha \cdot \prod_{n \in [1,q_u]\setminus\{i\}}(y+x_n)}$
- $H := U^{\beta}$
- $Y_1 := H^y$
- $Y_2 := (G_2)^y$
- $A_i := (\mathsf{A} \cdot (G_1)^y)^{\alpha\cdot(1+\beta\cdot f_i)\cdot\prod_{n\in[1,q_u]\setminus\{i,\mathsf{i}\}}(y+x_n)}$
- $A_{\mathsf{i}} := (G_1)^{\alpha\cdot(1+\beta\cdot f_{\mathsf{i}})\cdot\prod_{n\in[1,q_u]\setminus\{\mathsf{i}\}}(y+x_n)}$
- $N_{\mathsf{i}:\mathsf{j}} := H^{f_{\mathsf{i}}} \cdot \mathsf{C}$
- $N_{i:\mathsf{j}} := H^{f_i} \cdot \mathsf{B}^{x_i}$
- $N_{\mathsf{i}:j} := H^{f_{\mathsf{i}}} \cdot \mathsf{A}^{r_j}$
- $N_{i:j} := H^{f_i} \cdot (G_1)^{r_j \cdot x_i}$

for $i \neq \mathsf{i}$, $j \neq \mathsf{j}$. Note that x_{i} is not used in the signature oracle simulation (cf. game G_1). If i were asked to the corruption oracle, the simulation would fail, but since i is asked to the challenge oracle (cf. game G_0), it is not corrupted. Game G_2 and G_3 are identically distributed since a and b are uniformly random.

$$\Pr\left[\mathtt{A}^{\mathsf{G}_2} \Rightarrow 1\right] = \Pr\left[\mathtt{A}^{\mathsf{G}_3} \Rightarrow 1\right]. \tag{4}$$

Game 4. The game $\mathsf{G}_4(G_1, \mathsf{A}, \mathsf{B}, \mathsf{C})$ is identical to G_3, except that it takes as input three uniformly distributed group elements $\mathsf{A} = (G_1)^{\mathsf{a}}$, $\mathsf{B} = (G_1)^{\mathsf{b}}$ and $\mathsf{C} = (G_1)^{\mathsf{c}}$. A distinguisher between G_3 and G_4 is a distinguisher for the DDH problem in $\mathbb{G}_1$.

$$\Pr\left[\mathtt{A}^{\mathsf{G}_3} \Rightarrow 1\right] - \Pr\left[\mathtt{A}^{\mathsf{G}_4} \Rightarrow 1\right] \leq \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(t). \tag{5}$$

Game 5. Let i_0, i_1, j_0, j_1 be the users and domains asked to the challenge oracle. In game G_5, the only difference with game G_4 is that the challenge oracle returns the pseudonym of the user i_0 for a domain $j \in \{j_0, j_1\}$, for which $\{(i_0, j), (i_1, j)\} \cap \mathcal{N} = \emptyset$. There exists at least such a j, otherwise the challenge query is invalid, by the condition on the set $\mathcal{N}$ in the challenge oracle definition. We have $\mathsf{i} \in \{i_0, i_1\}$, $\mathsf{j} \in \{j_0, j_1\}$. Except the pseudonyms, no information about the user's secret key is available (in particular, signatures are simulated without knowledge of the secret keys – cf. game G_1). Since G_1, A, B and C are independent uniform elements in $\mathbb{G}_1$, all pseudonyms in the set $\{\mathsf{nym}[i][j], \mathsf{nym}[\mathsf{i}][j], \mathsf{nym}[i][\mathsf{j}], \mathsf{nym}[\mathsf{i}][\mathsf{j}]\}$ are independent and uniform. Moreover, this set of pseudonyms cannot be discriminated by the leakage of pseudonyms, thanks to the condition on the set $\mathcal{K}$ in the challenge oracle query. So the view of the adversary is not affected.

$$\Pr\left[\mathtt{A}^{\mathsf{G}_4} \Rightarrow 1\right] = \Pr\left[\mathtt{A}^{\mathsf{G}_5} \Rightarrow 1\right]. \tag{6}$$

Finally, in game G_5, the view of $\mathtt{A}$ is independent of one challenge bit among b_0 and b_1, so the success probability of $\mathtt{A}$ is exactly $1/2$.

$$\Pr\left[\mathtt{A}^{\mathsf{G}_5} \Rightarrow 1\right] = \frac{1}{2}. \tag{7}$$

The theorem follows from Equations (1), (2), (3), (4), (5), (6), and (7). □

5.3 Seclusiveness

The seclusiveness of our DSPS scheme is proved by reduction to the strong Diffie-Hellman problem.

Theorem 4 *Let $\mathtt{A}$ be an adversary against the seclusiveness of our DSPS scheme, running in time at most t, making at most q_h random oracle queries and adding q_u users. Then*

$$\mathbf{Adv}^{\mathsf{SEC}}_{\mathbb{G}_1,\mathbb{G}_2}(\mathtt{A}) \leq \frac{q_u + q_h}{p} + \sqrt{2 \cdot q_u \cdot q_h \cdot \mathbf{Adv}^{q_u\text{-}\mathsf{SDH}}_{\mathbb{G}_1,\mathbb{G}_2}(2 \cdot t + {q_u}^2)}.$$

Proof. We proceed with a sequence of games.

Game 0. Game G_0 is simply the security experiment as described Figure 2. So we have:

$$\Pr\left[\mathtt{A}^{\mathsf{G}_0} \Rightarrow 1\right] = \mathbf{Adv}^{\mathsf{SEC}}_{\mathbb{G}_1,\mathbb{G}_2}(\mathtt{A}). \tag{8}$$

Game 1. Game G_1 takes as input $q_u + 3$ group elements $(H_1, H_2, \mathsf{F}_1, \mathsf{Q}, \mathsf{F}_2, \ldots, \mathsf{F}_{q_u}) \in (\mathbb{G}_1^* \times \mathbb{G}_2)^2 \times (\mathbb{G}_1)^{q_u}$ such that $\mathsf{F}_1 = (H_1)^{\mathsf{f}}$, $\mathsf{Q} = (H_2)^{\mathsf{f}}$, $\mathsf{F}_2 = (H_1)^{(\mathsf{f}^2)}$, …, and $\mathsf{F}_{q_u} = (H_1)^{(\mathsf{f}^{(q_u)})}$, for some uniform $\mathsf{f} \in \mathbb{Z}_p^*$. For sake of reading, let us set $q := q_u$. G_1 differs from G_0 in the way the parameters, the group key and the user keys are generated. G_1 picks $k \leftarrow \{1, \ldots, q\}$, $x_1, \ldots, x_q \leftarrow \mathbb{Z}_p$ (such that all x_i are distinct), and $s_1, \ldots, s_q \leftarrow \mathbb{Z}_p$; sets $G_2 := H_2$, $Y_1 :=$

$\mathsf{F}_1 \cdot (H_1)^{-x_k}$ and $Y_2 := \mathsf{Q} \cdot (H_2)^{-x_k}$. This implicitly sets the group secret key $y := \mathsf{f} - x_k$. For $\{x_1, \ldots, x_q\} \in \mathbb{Z}_p$, let P, P_m and P_m^- for $m \in [1, q]$ be the following polynomials on $\mathbb{F}_p[X]$:

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

Note that 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 f, we get $P(\mathsf{f}) = \sum_{n=0}^{q} a_n \mathsf{f}^n$ for some $\{a_n\}_{n=0}^{q}$ depending on the x_n. Game G_1 is able to compute $(H_1)^{P(\mathsf{f})}$ from the input without the knowledge of f. The same remark is equally true for P_m and P_m^-. Game G_1 picks $\alpha \leftarrow \mathbb{Z}_p$, $\beta, \delta \leftarrow \mathbb{Z}_p^*$, sets $U := (H_1)^{\beta(\alpha \cdot P(\mathsf{f}) - s_k \cdot P_k(\mathsf{f}))}$, $H := (H_1)^{\beta \cdot P_k(\mathsf{f})}$, and $G_1 := (H_1)^{\delta}$. If G_1, H, U, Y_1, or $\mathsf{F}_1 \cdot (H_1)^{-x_k + x_i}$ (for some $i \in [1, q]$) equals $\mathbf{1}_{\mathbb{G}_1}$, or G_2, or Y_2 equals $\mathbf{1}_{\mathbb{G}_2}$, game G_1 restarts drawing random values. $\mathtt{A}$ is given parameters $(e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, H, U, G_2)$ and group public key (Y_1, Y_2). Then game G_1 computes user secret keys as follows. When $\mathtt{A}$ asks for adding a new user $i \in [1, q]$, G_1 sets $A_i := (H_1)^{\beta \cdot (\alpha \cdot P_i(\mathsf{f}) + P_i^-(\mathsf{f}) \cdot (s_i - s_k))}$, and $\mathsf{usk}[i] := (s_i, A_i, x_i)$. One can check that $\mathsf{usk}[i]$ is a valid user secret key under (Y_1, Y_2):

$$
\begin{aligned}
&\left(U \cdot H^{s_i}\right)^{\frac{1}{y + x_i}} \\
&\quad = \big((H_1)^{\beta \cdot (\alpha \cdot P(\mathsf{f}) - s_k \cdot P_k(\mathsf{f}))} \\
&\qquad\qquad \cdot (H_1)^{\beta \cdot P_k(\mathsf{f}) \cdot s_i}\big)^{\frac{1}{\mathsf{f} - x_k + x_i}} \\
&\quad = (H_1)^{(\beta \cdot \alpha \cdot P_i(\mathsf{f}) - \beta \cdot s_k \cdot P_i^-(\mathsf{f}) + \beta \cdot s_i \cdot P_i^-(\mathsf{f})) \cdot \frac{\mathsf{f} + x_i - x_k}{\mathsf{f} - x_k + x_i}} \\
&\quad = (H_1)^{\beta \cdot (\alpha \cdot P_i(\mathsf{f}) + P_i^-(\mathsf{f}) \cdot (s_i - s_k))} = A_i.
\end{aligned}
$$

When the issuing process is interactive, the game receives an element $H^{f'}$ and a proof Π. From Π, thanks to the extraction key, game G_2 extracts f'. Then it computes A_i as above, and returns $s_i - f'$ to the user. Parameters and keys are distributed as in game G_0, except that some x_i could take the same value in the specification of the scheme. If the x_i are uniformly picked, this happens with negligible probability.

$$
\Pr\left[\mathtt{A}^{\mathsf{G}_1} \Rightarrow 1\right] - \Pr\left[\mathtt{A}^{\mathsf{G}_0} \Rightarrow 1\right] \leq \frac{q_u}{p}. \qquad (9)
$$

Solving a SDH challenge. Now, let $\mathtt{B}$ denote the following reduction algorithm, whose aim is to solve a SDH challenge $(\mathsf{F}_1, \mathsf{Q}, \mathsf{F}_2, \ldots, \mathsf{F}_q) \in \mathbb{G}_1 \times \mathbb{G}_2 \times (\mathbb{G}_1)^{q-1}$ with respect to generators H_1, H_2. $\mathtt{B}$ runs $\mathtt{A}$, simulating game G_1 on inputs $(H_1, H_2, \mathsf{F}_1, \mathsf{Q}, \mathsf{F}_2, \ldots, \mathsf{F}_q)$, until $\mathtt{A}$ returns a forgery $\sigma = (T, c, \vec{s})$ for some domain D, pseudonym N and message m (assuming $\mathtt{A}$ is successful; otherwise $\mathtt{B}$ aborts). Since the forgery is valid, the random oracle was programmed as $\mathtt{H}(D, N, T, R_1, R_2, R_3, m) := c$.

$\mathtt{B}$ replays the same experiment. When $\mathtt{A}$ makes the random oracle query corresponding to the forgery, it sends a fresh random value c'. If $\mathtt{A}$ returns a new forgery $\sigma' = (T', c', \vec{s'})$ for the same message m, then:

$$
\begin{aligned}
&H^{s_f} \cdot D^{s_x} \cdot N^{-c} = H^{s'_f} \cdot D^{s'_x} \cdot N^{-c'} \\
\Leftrightarrow &\begin{cases} s_f + c \cdot f = s'_f + c' \cdot f \mod p & \text{and} \\ s_x + c \cdot x = s'_x + c' \cdot x \mod p, \end{cases}
\end{aligned}
$$

from which some x and f such that $N = H^f \cdot D^x$ can be retrieved (assuming $c \neq c'$). Moreover, we have:

$$
H^{-s_d} \cdot D^{-s_b} \cdot N^{s_a} = H^{-s'_d} \cdot D^{-s'_b} \cdot N^{s'_a},
$$

from which we know (by substituting $N = H^f \cdot D^x$) that $(s_d - s'_d) = f \cdot (s_a - s'_a) \mod p$ and $(s_b - s'_b) = x \cdot (s_a - s'_a) \mod p$. Finally, we have:

$$
\begin{aligned}
&e(T^{s_x} \cdot H^{-s_f - s_b} \cdot U^{-c}, G_2) \cdot e(H^{-s_a} \cdot T^c, Y_2) = \\
&e(T^{s'_x} \cdot H^{-s'_f - s_b} \cdot U^{-c'}, G_2) \cdot e(H^{-s'_a} \cdot T^{c'}, Y_2),
\end{aligned}
$$

from which we know that $s_a + a \cdot c = s'_a + a \cdot c' \mod p$ and $e(T \cdot H^{-a}, (G_2)^x \cdot Y_2) = e(U \cdot H^f, G_2)$. $\mathtt{B}$ can then retrieve a and $A = T \cdot H^{-a}$ such that $A = (U \cdot H^f)^{\frac{1}{y+x}}$. Assuming that $\mathtt{A}$ is successful, there is no $n \in [1, q]$, such that $N = H^{f_n} \cdot D^{x_n}$. Hence:

$$
(f, x) \notin \{(f_1, x_1), \ldots, (f_q, x_q)\}. \qquad (10)
$$

Let us now distinguish two cases (Ⅰ) and (Ⅱ).

(Ⅰ). $x \in \{x_1, \ldots, x_q\}$. If $x \neq x_k$, $\mathtt{B}$ returns $\perp$ and aborts. Let us now assume that $x = x_k$. We have $f \neq s_k$ – since $f = s_k$ contradicts (10) – and

$$
\begin{aligned}
&(A^{s_k} \cdot (A_k)^{-f})^{\frac{1}{s_k - f}} \\
&\quad = \big((H_1)^{s_k \cdot \beta \cdot P_k(\mathsf{f}) \cdot (\alpha \cdot \mathsf{f} - s_k + f) \cdot \frac{1}{\mathsf{f} - x_k + x}} \\
&\qquad\quad \cdot (H_1)^{-f \cdot \beta \cdot \alpha \cdot P_k(\mathsf{f})}\big)^{\frac{1}{s_k - f}} \\
&\quad = (H_1)^{s_k \cdot \beta \cdot P_k(\mathsf{f}) \cdot (\alpha \cdot \mathsf{f} - s_k + f) \cdot \frac{1}{\mathsf{f}} \cdot \frac{1}{s_k - f}} \\
&\qquad\quad \cdot (H_1)^{-f \cdot \beta \cdot \alpha \cdot \mathsf{f} \cdot P_k(\mathsf{f}) \cdot \frac{1}{\mathsf{f}} \cdot \frac{1}{s_k - f}} \\
&\quad = (H_1)^{\beta \cdot (\alpha \cdot P(\mathsf{f}) - s_k \cdot P_k(\mathsf{f})) \cdot \frac{1}{\mathsf{f}}}
\end{aligned}
$$

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

$$C := R(0) = -\beta \cdot s_k \cdot \left[\prod_{n=1, n\neq k}^{q} (x_n - x) \right] \text{ and}$$
$$(A^{s_k} \cdot (A_k)^{-f})^{\frac{1}{s_k - f}} = (H_1)^{\frac{C}{\mathsf{f}} + Q(\mathsf{f})}$$

where $C \neq 0$. $\mathtt{B}$ computes

$$(H_1)^{1/\mathsf{f}} := ((A^{s_k} \cdot (A_k)^{-f})^{\frac{1}{s_k - f}} \cdot (H_1)^{-Q(\mathsf{f})})^{1/C},$$

sets $c := 0$ and returns $(0, (H_1)^{1/\mathsf{f}})$ as a solution to the SDH challenge.

(II). $x \notin \{x_1, \ldots, x_q\}$. This implies:

$$x_n - x \neq 0 \mod p \text{ for all } n \in [1, q]. \quad (11)$$

Let us now consider the quantity $\beta \cdot P_k(\mathsf{f}) \cdot (\alpha \cdot \mathsf{f} + f - s_k)$ as a polynomial S in f. If we carry out the Euclidean division of S by $(\mathsf{f} + x - x_k)$, we get Q and R such that $S(\mathsf{f}) = (\mathsf{f} + x - x_k) \cdot Q(\mathsf{f}) + R(\mathsf{f})$. Since $(\mathsf{f} + x - x_k)$ is a first degree polynomial $X - (x_k - x)$, we know that $R(\mathsf{f}) = S(x_k - x)$, so $\mathtt{B}$ can compute

$$C := R(\mathsf{f}) = S(x_k - x) = \beta \cdot \left[\prod_{n=1, n\neq k}^{q} (x_n - x) \right] \cdot (\alpha \cdot (x_k - x) + f - s_k).$$

Since (f, A, x) is a valid user key, we have

$$A = (H_1)^{\beta \cdot P_k(\mathsf{f}) \cdot (\alpha \cdot \mathsf{f} - s_k + f) \cdot \frac{1}{\mathsf{f} - x_k + x}}$$

so $A = (H_1)^{\frac{(\mathsf{f}+x-x_k)\cdot Q(\mathsf{f}) + R(\mathsf{f})}{\mathsf{f} - x_k + x}} = (H_1)^{Q(\mathsf{f}) + \frac{C}{\mathsf{f}+x-x_k}}$.

$\mathtt{B}$ can compute $(H_1)^{Q(\mathsf{f})}$ from the SDH challenge. If $(f - s_k) = \alpha \cdot (x - x_k)$, $\mathtt{B}$ returns $\perp$ and aborts. If $(f - s_k) \neq \alpha \cdot (x - x_k)$, then $C \neq 0$ by (11) and by the choice of β, so $\mathtt{B}$ computes $(H_1)^{\frac{1}{\mathsf{f}+x-x_k}} = \left(A \cdot (H_1)^{-Q(\mathsf{f})}\right)^{\frac{1}{C}}$, set $c = x - x_k$, and return $(c, (H_1)^{1/(\mathsf{f}+c)})$ as a solution to the SDH challenge.

This achieves the description of the reduction $\mathtt{B}$. Let us now consider its probability of success and its running time. Let us denote ϵ the probability $\Pr\left[\mathtt{A}^{\mathsf{G}_1} \Rightarrow 1\right]$ and ϵ' the probability that $\mathtt{B}$ manages to extract a valid SDH solution from the forge. From standard forking techniques (cf. [3, Lemma 1]), we have that:

$$\epsilon' \geq \epsilon \cdot \left(\frac{\epsilon}{q_h} + \frac{1}{p} \right). \quad (12)$$

Then, let us denote ϵ'' the probability that $\mathtt{B}$ does not abort, once a secret key is extracted. No information is available about k from $\mathtt{A}$'s point of view. As a result, in case (I), $\mathtt{B}$ succeeds with probability $1/q_u$; and in case (II), $\mathtt{B}$ succeeds with probability at least $1 - 1/q_u$. Hence, considering the more pessimistic case, $\epsilon' \leq 2 \cdot q_u \cdot \epsilon''$. The running time of $\mathtt{B}$ is essentially twice the running time of $\mathtt{A}$ (to extract a SDH challenge) plus $O({q_u}^2)$ operations in $\mathbb{G}_1$. Therefore, we get:

$$\Pr\left[\mathtt{A}^{\mathsf{G}_1} \Rightarrow 1\right] \leq \frac{q_h}{p} + \sqrt{2 \cdot q_u \cdot q_h \cdot \mathbf{Adv}^{q_u\text{-}\mathsf{SDH}}_{\mathbb{G}_1, \mathbb{G}_2}(2 \cdot t + {q_u}^2)}, \quad (13)$$

which concludes the proof. □

5.4 Unforgeability

The unforgeability of our DSPS scheme is proved by reduction to the discrete logarithm problem.

Theorem 5 *Let $\mathtt{A}$ be an adversary against the unforgeability of the pseudonymous signature scheme, running in time at most t, making at most q_h random oracle queries, at most q_s signature queries, and adding q_u users. Then*

$$\mathbf{Adv}^{\mathsf{UF}}_{\mathbb{G}_1, \mathbb{G}_2}(\mathtt{A}) \leq \frac{q_u \cdot (q_s \cdot (q_s + q_h) + q_h)}{p} + q_u \cdot \sqrt{q_h \cdot \mathbf{Adv}^{\mathsf{DL}}_{\mathbb{G}_1}(2 \cdot t)}.$$

Proof. We proceed with a sequence of games.

Game 0. Game G_0 is the security experiment as described Figure 3 where, in addition, G_0 uniformly guesses the user $\mathsf{i} \in [1, q_u]$ returned by the adversary, and fail if the guess is wrong. So we have:

$$\Pr\left[\mathtt{A}^{\mathsf{G}_0} \Rightarrow 1\right] = \frac{1}{q_u} \cdot \mathbf{Adv}^{\mathsf{UF}}_{\mathbb{G}_1, \mathbb{G}_2}(\mathtt{A}). \quad (14)$$

Game 1. In game G_1, the signature queries are simulated as in Game G_1 of Theorem 3. The fact that the adversary knows the group secret key has no effect on the simulation.

$$\Pr\left[\mathtt{A}^{\mathsf{G}_0} \Rightarrow 1\right] - \Pr\left[\mathtt{A}^{\mathsf{G}_1} \Rightarrow 1\right] \leq \frac{q_s \cdot (q_s + q_h)}{p}. \quad (15)$$

Game 2. The difference between game G_2 and game G_1 is in the way the parameters, the group public key and the user secret keys are computed. The game draws $G_1, F \leftarrow \mathbb{G}_1 \setminus \{\mathbf{1}_{\mathbb{G}_1}\}$; $G_2 \leftarrow \mathbb{G}_2 \setminus \{\mathbf{1}_{\mathbb{G}_2}\}$; $\beta, y, f_1, \ldots f_{q_u} \leftarrow \mathbb{Z}_p^*$, $x_1, \ldots, x_{q_u} \leftarrow \mathbb{Z}_p \setminus \{y\}$, and sets $U := F^{(y+x_1)\cdots(y+x_{q_u})}$, $H := U^{\beta}$, $Y_1 := H^y$, $Y_2 := (G_2)^y$, and $A_i := F^{(1+\beta \cdot f_i) \cdot \prod_{n \in [1, q_u] \setminus \{i\}} (y + x_n)}$ for all $i \in [1, q_u]$. When the game interacts with the

adversary as a corrupted issuer, the game simply sends a value $H^{f'}$, the corresponding proof, receives (f'', A, x), and sets $f = f' + f''$. G_1, H, U and G_2 are distributed as in Game G_1, and we have:

$$\Pr\left[\mathtt{A}^{\mathsf{G}_1} \Rightarrow 1\right] = \Pr\left[\mathtt{A}^{\mathsf{G}_2} \Rightarrow 1\right]. \quad (16)$$

Game 3. $\mathsf{G}_3(F, \mathsf{A})$ takes as input two group elements such that $\mathsf{A} = F^{\mathsf{a}}$ for some uniform $\mathsf{a} \in \mathbb{Z}_p$. The difference with game G_2 is in the way the secret key and the pseudonyms of user i are computed. G_2 sets $A := F \cdot \mathsf{A}^{\beta}$ and $\mathsf{nym}_{\mathsf{i}j} := \mathsf{A} \cdot (D_j)^{x_{\mathsf{i}}}$. This implicitly sets $f_{\mathsf{i}} := \mathsf{a}$. If the user i was created through an interactive process, the challenger simulates a proof of equality (in the first round), and receives A_{i} from the corrupted issuer (in the second round). In this case we have, $H^{f_{\mathsf{i}}} = \mathsf{A} \cdot H^{f''}$ for some adversarially chosen f''. It implicitly sets $f_{\mathsf{i}} := \mathsf{a} + f''$ instead of $f_{\mathsf{i}} := \mathsf{a}$. The signature oracle is not affected since the users' secret keys are no longer used in the simulation of the signatures (cf. game G_1). No information about i leaks from the adversary's point of view, unless i is corrupted.

$$\Pr\left[\mathtt{A}^{\mathsf{G}_2} \Rightarrow 1\right] = \Pr\left[\mathtt{A}^{\mathsf{G}_3} \Rightarrow 1\right]. \quad (17)$$

Solving a DL challenge. Now, let $\mathtt{B}$ denote the following reduction algorithm, whose aim is to solve a Discrete Logarithm challenge $\mathsf{A} \in \mathbb{G}_1$ with respect to a generator F. $\mathtt{B}$ runs $\mathtt{A}$, simulating game $\mathsf{G}_3(F, \mathsf{A})$, until $\mathtt{A}$ returns a forgery $\sigma = (T, c, \vec{s})$ for some domain D, and message m (assuming $\mathtt{A}$ is successful; otherwise $\mathtt{B}$ aborts). Since the forgery is valid, the random oracle was programmed as $\mathtt{H}(D, N, T, R_1, R_2, R_3, m) := c$. As in Theorem 4, $\mathtt{B}$ replays the same experiment, and then sends a fresh random value c' for the corresponding random oracle query. If $\mathtt{A}$ returns a new forgery $\sigma' = (T', c', \vec{s'})$ for the same message m, then:

$$\begin{aligned} H^{s_f} \cdot D^{s_x} \cdot N^{-c} &= H^{s'_f} \cdot D^{s'_x} \cdot N^{-c'} \\ &\Rightarrow s_f + c \cdot f = s'_f + c' \cdot f \mod p, \end{aligned}$$

from which $f = \mathsf{a}$ (or $f = \mathsf{a} + f''$) can be retrieved, assuming that the adversary produces a forge for the user i and assuming $c \neq c'$. Let us denote ϵ the probability $\Pr\left[\mathtt{A}^{\mathsf{G}_3} \Rightarrow 1\right]$ and ϵ' the probability that $\mathtt{B}$ manages to extract a valid DL solution from the forge. As in Theorem 4, forking techniques [3, Lemma 1] allows to show that:

$$\Pr\left[\mathtt{A}^{\mathsf{G}_3} \Rightarrow 1\right] \leq \frac{q_h}{p} + \sqrt{q_h \cdot \mathbf{Adv}^{\mathsf{DL}}_{\mathbb{G}_1}(2 \cdot t)}. \quad (18)$$

The theorem follows from Equations (14), (15), (16), (17), and (18). □

6 Conclusion

In this work, we study *dynamic domain-specific pseudonymous signatures*, a recent privacy-preserving primitive, which brings many interesting applications. We supply clean definitions for their security properties. We highlight the fact that, in some sense, using the full power of group signatures is somehow "too strong" for constructing DSPS signatures. Following this intuition, we provide a new, optimized, construction that is more efficient than the one given in [12], while achieving the same strong security and privacy properties. Moreover, motivated by the main application of DSPS – the authentication of eID tokens –, we study the interactions between an entity, as a chip, embedding a user's secret key, and its reader. Taking advantage of the computational power of the reader, our DSPS scheme may be implemented on existing chips, without deploying dedicated hardware or pairing implementations on the chip side.

A Definition of the oracles in the security games

Figures 4 and 5 give the details of the oracles available to the adversary in the security games of Section 4.

B A proof of knowledge of a valid certificate

The signature procedure of our scheme is obtained from a proof of knowledge of a valid certificate. This proof is explicitly given Figure 6. It is indeed a honest-verifier zero-knowledge proof of knowledge, as shown by the following lemmas.

Lemma 6 (Completeness) *The protocol of Figure 6 is complete.*

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

$$\begin{aligned} & H^{s_f} \cdot \mathsf{dpk}^{s_x} \cdot \mathsf{nym}^{-c} \\ &= H^{r_f + c \cdot f} \cdot \mathsf{dpk}^{r_x + c \cdot x} \cdot (H^f \cdot \mathsf{dpk}^x)^{-c} \\ &= H^{r_f + c \cdot f - f \cdot c} \cdot \mathsf{dpk}^{r_x + c \cdot x - x \cdot c} \\ &= H^{r_f} \cdot \mathsf{dpk}^{r_x} = R_1 \end{aligned}$$

initialization $\texttt{InitializeGame}()$

$\mathcal{U} := \{\}$; $\mathcal{C} := \{\}$; $\mathcal{D} := \{\}$; $\mathcal{N} := \{\}$; $\mathcal{S} := \{\}$; $sig\_state := 0$; $\mathcal{L} := \perp$; $\mathcal{K} := \perp$;
$\mathsf{ro} := \perp$; $\mathsf{usk}[i] := \perp$, $\mathsf{rt}[i] := \perp$, $\mathsf{dpk}[j] := \perp$, $\mathsf{nym}[i][j] := \perp$, for all i, j

users $\texttt{U}(\mathsf{gsk}, i)$, $i \in \mathbb{N}$

1. if $i \in \mathcal{U} \cup \mathcal{C}$, return $\perp$; otherwise set $\mathcal{U} := \mathcal{U} \cup \{i\}$; generate $\mathsf{usk}[i], \mathsf{rt}[i]$ with gsk
2. set $\mathsf{nym}[i][j] \leftarrow \texttt{NymGen}(\mathsf{gpk}, \mathsf{dpk}[j], \mathsf{usk}[i])$ for all $j \in \mathcal{D}$; return ε

domains $\texttt{D}(j)$, $j \in \mathbb{N}$

1. if $j \in \mathcal{D}$, return $\perp$; otherwise set $\mathsf{dpk}[j] \leftarrow \texttt{DomainKeyGen}()$; for all $i \in \mathcal{U} \cup \mathcal{C}$, do:
   (a) if $\mathsf{usk}[i] \neq \perp$, set $\mathsf{nym}[i][j] \leftarrow \texttt{NymGen}(\mathsf{gpk}, \mathsf{dpk}[j], \mathsf{usk}[i])$
   (b) otherwise if $\mathsf{rt}[i] \neq \perp$, set $\mathsf{nym}[i][j] \leftarrow \texttt{Revoke}(\mathsf{gpk}, \mathsf{rt}[i], \mathsf{dpk}[j])$
2. set $\mathcal{D} := \mathcal{D} \cup \{j\}$; $\mathcal{N} := \mathcal{N} \cup \{(i,j)\}_{i \in \mathcal{C}}$; $\mathcal{L}[j] \leftarrow \{\}$; $\mathcal{K}[j] \leftarrow \{\}$; $\texttt{Check}(j)$; return $\mathsf{dpk}[j]$

corruption oracle $\texttt{C}(i)$, $i \in \mathbb{N}$

1. if $i \notin \mathcal{U}$ (or if i was queried to the $\texttt{LoR}$ oracle), return $\perp$
2. otherwise set $\mathcal{U} := \mathcal{U} \setminus \{i\}$; $\mathcal{C} := \mathcal{C} \cup \{i\}$; $\mathcal{N} := \mathcal{N} \cup \{(i,j)\}_{j \in \mathcal{D}}$; $\forall j \in \mathcal{D}$: $\texttt{Check}(j)$
3. return $(\mathsf{usk}[i], \mathsf{rt}[i])$ *// pseudonyms might be computed from* $\mathsf{usk}[i]$

revocation oracle $\texttt{R}(i,j)$, $i, j \in \mathbb{N}$

1. if $i \notin \mathcal{U}$ or $j \notin \mathcal{D}$ (or if i and j were queried together to the $\texttt{LoR}$ oracle), return $\perp$
2. otherwise set $\mathcal{L}[j] := \mathcal{L}[j] \cup \{\mathsf{nym}[i][j]\}$; $\mathcal{N} := \mathcal{N} \cup \{(i,j)\}$; $\texttt{Check}(j)$; return ε

nym oracle: $\texttt{N}(i,j)$, $i, j \in \mathbb{N}$

1. if $i \notin \mathcal{U}$ or $j \notin \mathcal{D}$ (or if i and j were queried together to the $\texttt{LoR}$ oracle), return $\perp$
2. otherwise set $\mathcal{N} := \mathcal{N} \cup \{(i,j)\}$; $\texttt{Check}(j)$; return $\mathsf{nym}[i][j]$

identification oracle: $\texttt{I}(N,j)$, $N \in \{0,1\}^*$, $j \in \mathbb{N}$

1. find $i \in \mathcal{U}$ such that $N = \mathsf{nym}[i][j]$
2. if none is found or $j \notin \mathcal{D}$ (or if i and j were queried together to the $\texttt{LoR}$ oracle), return $\perp$
3. otherwise set $\mathcal{N} := \mathcal{N} \cup \{(i,j)\}$; $\texttt{Check}(j)$; and return i

nym leakage: $\texttt{L}(j)$, $j \in \mathbb{N}$

1. if $j \notin \mathcal{D}$, return $\perp$; otherwise set $\mathcal{K}[j] := \mathcal{K}[j] \cup \mathcal{U}$; return $\{(j, \mathsf{nym}[i][j])\}_{i \in \mathcal{U}}$

random oracle $\texttt{H}_{\mathcal{R}}(string)$, $string \in \{0,1\}^*$

1. if $\mathsf{ro}[string] \neq \perp$, return $\mathsf{ro}[string]$
2. otherwise draw $r \leftarrow \mathcal{R}$; set $\mathsf{ro}[string] := r$; return r

$\texttt{Check}(j)$ for $j \in \mathcal{D}$ is defined as: [if $\exists i \in \mathcal{U}$: $\{i' : (i',j) \in \mathcal{N}\} \setminus \mathcal{U} = \{i\}$, then set $\mathcal{N} := \mathcal{N} \cup \{(i,j)\}$; return (i,j)]
it ensures that the anonymity is not trivially broken by dynamic and adaptive identification

Figure 4: Oracles for the experiments – I

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

Lemma 7 (Zero-knowledge) *The protocol of Figure 6 is honest-verifier zero-knowledge.*

Proof. 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 simu-

signature oracle $\mathtt{S}_{\mathcal{M}}(N, j, m)$, $N \in \{0,1\}^*$, $j \in \mathbb{N}$, $m \in \mathcal{M}$

1. find a user $i \in \mathcal{U}$ such that $N = \mathsf{nym}[i][j]$ (return $\perp$ if none is found)
2. compute $\sigma \leftarrow \mathtt{Sign}(\mathsf{gpk}, \mathsf{dpk}[j], \mathsf{usk}[i], N, m)$; set $\mathcal{S} := \mathcal{S} \cup \{(i, j, m, \sigma)\}$; return σ

signature oracle with delegation $\mathtt{DelS}_{\mathcal{M}}(\mathsf{in}_0, \mathsf{in}_1)$

1. if $sig\_state = 0$
   (a) parse in_0 as $(N, j, m) \in \{0,1\}^* \times \mathbb{N} \times \mathcal{M}$; find $i \in \mathcal{U}$ such that $N = \mathsf{nym}[i][j]$
   (b) compute $(del, aux) \leftarrow \mathtt{Delegate}(\mathsf{gpk}, \mathsf{dpk}[j], \mathsf{usk}[i], N, m)$
   (c) set $sig\_state := 1 - sig\_state$; store (i, j, m, aux); return del
2. if $sig\_state = 1$
   (a) parse in_1 as $resp \in \{0,1\}^*$; retrieve (i, j, m, aux)
   (b) $\sigma \leftarrow \mathtt{Finalize}(\mathsf{gpk}, aux, resp)$
   (c) $sig\_state := 1 - sig\_state$; $\mathcal{S} := \mathcal{S} \cup \{(i, j, m, \sigma)\}$; return σ

verification $\mathtt{V}_{\mathcal{M}}(N, j, m, \sigma)$, $N \in \{0,1\}^*$, $j \in \mathbb{N}$, $m \in \mathcal{M}$, $\sigma \in \{0,1\}^*$

1. if $j \notin \mathcal{D}$, return $\perp$; otherwise return $\mathtt{Verify}(\mathsf{gpk}, \mathsf{dpk}[j], N, m, \sigma, \mathcal{L}[j])$

interactive issuing with a corrupted issuer $\mathtt{StoU}(M_{in})$

1. if M_{in} is parsed as $i \in \mathbb{N}$ *// the adversary asks for a session with a new user*
   (a) if $i \in \mathcal{U} \cup \mathcal{C}$, return $\perp$; otherwise initiate an issuing session on behalf of i
   (b) compute a protocol first message M_{out} and keep the internal state as aux
2. otherwise if M_{in} is parsed as $(i, resp)$
   (a) retrieve the internal state aux of i and finalize $\mathsf{usk}[i]$ with internal state aux and response $resp$
   (b) set $\mathcal{U} := \mathcal{U} \cup \{i\}$; $\mathsf{nym}[i][j] \leftarrow \mathtt{NymGen}(\mathsf{gpk}, \mathsf{dpk}[j], \mathsf{usk}[i])$ for all $j \in \mathcal{D}$

interactive issuing with a corrupted user $\mathtt{StoI}(\mathsf{gsk}, i, M_{in})$

1. the oracle simulates the issuer in front of a new user $i \notin \mathcal{U} \cup \mathcal{C}$; it computes a response to the protocol message M_{in} thanks to the group secret key gsk, sends the response, and records a revocation token $\mathsf{rt}[i]$. Then, it sets $\mathcal{C} := \mathcal{C} \cup \{i\}$, $\mathcal{N} \cup \{(i, j)\}_{j \in \mathcal{D}}$, and $\mathsf{nym}[i][j] \leftarrow \mathtt{Revoke}(\mathsf{gpk}, \mathsf{rt}[i], \mathsf{dpk}[j])$ for all $j \in \mathcal{D}$.

In the oracles for the interactive issuing, we assume for simplicity that the protocol contains two rounds, as in our scheme.

Figure 5: Oracles for the experiments – II

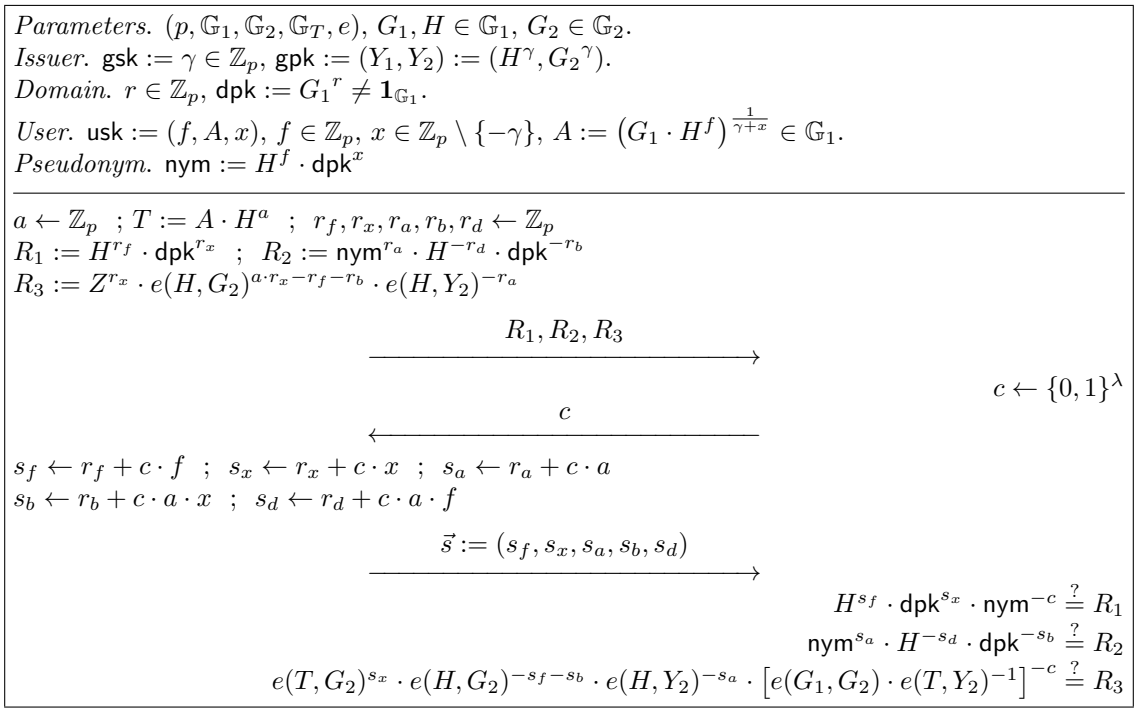
late T, which can be done by picking $T \leftarrow \mathbb{G}_1$. This element is indistinguishable from the output of any prover since, given $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, $G_1, H \leftarrow \mathbb{G}_1 \setminus \{\mathbf{1}\}$, $G_2 \leftarrow \mathbb{G}_2 \setminus \{\mathbf{1}\}$,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

$$\Delta := \{T \mid a \leftarrow \mathbb{Z}_p \ ; \ T := A \cdot H^a\}$$
$$\Delta' := \{T \mid T \leftarrow \mathbb{G}_1\}$$

Then, we pick $c \leftarrow \{0,1\}^\lambda$ and s_f, s_x, s_a, s_b, $s_d \leftarrow \mathbb{Z}_p$. The simulator computes (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. □

Lemma 8 (Proof of knowledge) *There exists an extractor for the protocol of Figure 6.*

Proof. 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 $H^{s_f} \cdot \mathsf{dpk}^{s_x} = R_1 \cdot \mathsf{nym}^c$ by $H^{s'_f} \cdot \mathsf{dpk}^{s'_x} = R_1 \cdot \mathsf{nym}^{c'}$, 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 $\mathsf{nym}^{s_a} \cdot H^{-s_d} \cdot \mathsf{dpk}^{-s_b} = R_2$ by $\mathsf{nym}^{s'_a} \cdot H^{-s'_d} \cdot \mathsf{dpk}^{-s'_b} = R_2$, 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



Scalar operations are carried out modulo p

Figure 6: A proof of knowledge of a valid certificate

dpk are generators of $\mathbb{G}_1$). Finally by dividing

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

we obtain:

$$\begin{aligned} &e(T, G_2)^{s_x - s'_x} \cdot e(H, G_2)^{-s_f + s'_f - s_b + s'_b} \\ &\cdot e(H, Y_2)^{-s_a + s'_a} = \left[e(G_1, G_2) \cdot e(T, Y_2)^{-1}\right]^{c - c'}. \end{aligned}$$

If we set $\tilde{a} := (s_a - s'_a)/(c - c')$ and use the previously obtained equality $(s_b - s'_b) = \tilde{x} \cdot (s_a - s'_a)$, we have:

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

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. □

C Cross-domain anonymity w.r.t. several challenge calls

In this section, we show that, if a DSPS scheme is cross-domain-anonymous with a single call to the challenge oracle, then it is still cross-domain-anonymous with a polynomial call to the challenge oracle.

Let us first begin by extending the definition of the cross-domain anonymity experiment to allow ℓ queries to the challenge oracle. The only difference with the experiment of Figure 1 is that the following non-triviality conditions are added in the LoR oracle.

4. If $\exists (i,j) \in \{i_0, i_1\} \times \{j_0, j_1\}$ such that (i,j) was previously queried to the LoR oracle, return $\perp$.
5. For $b \in \{0,1\}$, let us define $\bar{i} = i_{1-b}$ (resp. $\bar{j} = j_{1-b}$) if $i = i_b$ (resp. $j = j_b$). If $\exists (i,j) \in \{i_0, i_1\} \times \{j_0, j_1\}$: $((i,j) \in \mathcal{N}$ and (either $(i, \bar{j}) \in \mathcal{N}$ or $(\bar{i}, \bar{j}) \in \mathcal{N}$)), return $\perp$.

The condition (4.) states that different calls to the oracle should not involve the same users. Indeed the adversary can easily win if the sets of queried users overlap.

The condition (5.) is due to the fact that we allow the users in the challenge call not to be

anonymous in some domain, provided that they are anonymous in the other domain. An adversary would trivially win the game by guessing both bits through this leakage, so the anonymity is broken in one domain, it is always in the same side – the right or the left – in the different left-or-right calls.

We now state an intermediate result. Without loss of generality, we can assume that, if the anonymity is broken in one domain, it is in the left domain. To show this, we construct, from any ℓ-CDA adversary $\mathtt{A}$, a ℓ-CDA adversary $\mathtt{A}'$ that achieves this property.

Lemma 9 *Let $\mathtt{A}$ be a ℓ-CDA adversary, running in time at most t. There exists a ℓ-CDA adversary $\mathtt{A}'$, running in nearly the same time, such that, for all valid query (i_0, i_1, j_0, j_1) to $\mathtt{LoR}$, information about the pseudonym for (i_0, j_1) nor for (i_1, j_1) has leaked (through a call to the revocation $\mathtt{R}$, identification $\mathtt{I}$ or pseudonym $\mathtt{N}$ oracle, or through a $\mathtt{Check}$), and such that:*

$$\mathbf{Adv}^{\ell\text{-CDA}}_{\mathsf{param}}(\mathtt{A}) \leq 2 \cdot \mathbf{Adv}^{\ell\text{-CDA}}_{\mathsf{param}}(\mathtt{A}').$$

Note: a valid query is a query such that the conditions required by the oracle are fulfilled.

Proof. We construct $\mathtt{A}'$ as a ℓ-CDA challenger for $\mathtt{A}$, $\mathtt{A}'$ having its own ℓ-CDA challenger, say $\mathtt{C}$. Basically, $\mathtt{A}'$ only transfers the messages between $\mathtt{A}$ and $\mathtt{C}$, except for the $\mathtt{LoR}$ oracle. We restrict ourselves to the valid $\mathtt{LoR}$ queries; $\mathtt{A}'$ simply returns $\perp$ for invalid queries.

First we ensure that, for each information $\mathtt{A}$ might get about the pseudonym for a pair (i, j), $\mathtt{A}'$ knows the corresponding pseudonym. That is, each time $\mathtt{A}$ calls the revocation oracle $\mathtt{R}(i, j)$, $\mathtt{A}'$ transfers the query, and calls in addition $\mathtt{N}(i, j)$; and each time $\mathtt{Check}$ returns a pair (i, j), $\mathtt{A}'$ calls $\mathtt{N}(i, j)$.

Let (i_0, i_1, j_0, j_1) be the first query from $\mathtt{A}$ to its $\mathtt{LoR}$ oracle for which i_0 or i_1 is not anonymous in j_0 or j_1; *i.e.*, for which $\exists(\mathsf{i},\mathsf{j}) \in \{i_0, i_1\} \times \{j_0, j_1\}$ such that (i,j) was involved in a call to the revocation $\mathtt{R}$, identification $\mathtt{I}$, pseudonym $\mathtt{N}$ oracle, or a leakage through $\mathtt{Check}$.

If $\mathsf{j} == j_0$, then $\mathtt{A}$ already has the requested property. Indeed, by condition (5.), users will be anonymous in j_1 in all future $\mathtt{LoR}$ calls. In this case, $\mathtt{A}'$ only transfers messages between $\mathtt{A}$ and $\mathtt{C}$ during the whole game.

If $\mathsf{j} == j_1$, $\mathtt{A}'$ makes a call $(i_0,\ i_1,\ j_1,\ j_0)$ to its challenger, and gets $\{(j_1,\ \mathsf{nym}_{i_{b_0} j_1}),\ (j_0,\ \mathsf{nym}_{i_{b_1} j_0})\}$. Since the pseudonym for the user i is known in the domain j_1, $\mathtt{A}'$ can deduce b_0. Then it makes a guess, whether $(b_0 == b_1)$ or not. According to this guess, it computes the response to the future $\mathtt{LoR}$ calls from $\mathtt{A}$, possibly with the help of its $\mathtt{N}$ oracle if needed.

At the end of the game, $\mathtt{A}'$ returns $\mathtt{A}$'s output. The simulation of $\mathtt{A}'$ is correct at least with probability $\frac{1}{2}$. The lemma follows. □

Theorem 10 *Let $\mathtt{A}$ be an adversary against the cross-domain anonymity of a pseudonymous signature scheme, running in time at most t. Then*

$$\mathbf{Adv}^{\ell\text{-CDA}}_{\mathsf{param}}(\mathtt{A}) \leq 2 \cdot \ell \cdot \mathbf{Adv}^{1\text{-CDA}}_{\mathsf{param}}(t).$$

Proof. We proceed with a sequence of $\ell + 1$ games. Game G_n, for $n \in [0, \ell]$, is the ℓ-CDA game, except for the $\mathtt{LoR}$ oracle. The game G_n maintains a counter $c \in [1, \ell]$ for each call to the $\mathtt{LoR}$ oracle. At the c-th call, if $(n < c)$, then the oracle $\mathtt{LoR}$ returns $\{(j_0,\ \mathsf{nym}_{i_{b_0} j_0}),\ (j_1,\ \mathsf{nym}_{i_{b_0} j_1})\}$, otherwise (i.e. if $(n \geq c)$), it returns $\{(j_0,\ \mathsf{nym}_{i_{b_0} j_0}),\ (j_1,\ \mathsf{nym}_{i_{b_0} j_1})\}$. Game G_0 is simply the ℓ-CDA experiment, so we have:

$$\Pr\left[\mathtt{A}^{\mathsf{G}_0} \Rightarrow 1\right] = \frac{\mathbf{Adv}^{\ell\text{-CDA}}_{\mathsf{param}}(\mathtt{A}) + 1}{2}. \tag{19}$$

In game G_ℓ, no information on b_1 is available, so we have:

$$\Pr\left[\mathtt{A}^{\mathsf{G}_\ell} \Rightarrow 1\right] = \frac{1}{2}. \tag{20}$$

From (19) and (20), we have:

$$\Pr[\mathtt{A}^{\mathsf{G}_0} \Rightarrow 1] - \Pr[\mathtt{A}^{\mathsf{G}_\ell} \Rightarrow 1] = \frac{1}{2} \cdot \mathbf{Adv}^{\ell\text{-CDA}}_{\mathsf{param}}(\mathtt{A}). \tag{21}$$

We now restrict ourselves to an adversary $\mathtt{A}$ for which users in each left-or-right $\mathtt{LoR}$ call are never identified in the right side of the call. Then we apply Lemma 9.

We describe ℓ solvers $\mathtt{B}_n$, for $n \in [0, \ell - 1]$, for the 1-CDA property. $\mathtt{B}_n$ runs $\mathtt{A}$ as ℓ-CDA challenger, and is itself a 1-CDA adversary against its own challenger. $\mathtt{B}$ transfers all queries from $\mathtt{A}$ to its own challenger, except for the $\mathtt{LoR}$ oracle, in which case it proceeds as follows. Let c be a counter for the queries. If $(n < c)$, then $\mathtt{B}_n$ uses its own oracle $\mathtt{N}$ to determine $\mathsf{nym}_{i_{b_0} j_0}$ and $\mathsf{nym}_{i_{b_1} j_1}$, and returns $\{(j_0, \mathsf{nym}_{i_{b_0} j_0}), (j_1, \mathsf{nym}_{i_{b_1} j_1})\}$. If $(n > c)$, then $\mathtt{B}_n$ uses its own oracle $\mathtt{N}$ to determine $\mathsf{nym}_{i_{b_0} j_0}$ and $\mathsf{nym}_{i_{b_0} j_1}$, and returns $\{(j_0, \mathsf{nym}_{i_{b_0} j_0}), (j_1, \mathsf{nym}_{i_{b_0} j_1})\}$. If $(c == n)$, then $\mathtt{B}_n$ uses its own oracle $\mathtt{N}(i_{b_0}, j_0)$, transfers $\mathtt{A}$'s call (i_0, i_1, j_0, j_1) to its own $\mathtt{LoR}$ oracle, gets the response $\{(j_0, \mathsf{nym}'), (j_1, \mathsf{nym}'')\}$, and returns $\{(j_0, \mathsf{nym}_{i_{b_0} j_0}), (j_1, \mathsf{nym}'')\}$.

$\mathtt{B}_n$ calls the nym oracle $\mathtt{N}$ on some identities before its call to the challenge oracle. This implies some restrictions on the challenge, and $\mathtt{B}_n$ might fail to comply with these restrictions if it asks its challenge oracle for a disallowed identity. However, the calls originate from $\mathtt{A}$. $\mathtt{B}_n$ only transfers them. When $\mathtt{A}$ supplies some identities to the challenge, these identities do not appear in a future call, by condition (4.). In particular, they do not appear in the call that leads to $\mathtt{B}_n$'s challenge call. In addition, when $(c == n)$, $\mathtt{B}_n$ performs some identification (i_{b_0}, j_0) through a call to $\mathtt{N}$. Since $\mathtt{A}$ is a restricted adversary for which users are always anonymous in j_1, the $\mathtt{LoR}$ call of $\mathtt{B}_n$ is safe.

Let now $\mathsf{b}_0, \mathsf{b}_1$ be the bits flipped by the challenger of $\mathtt{B}_n$. Recall that b_0, b_1 are the bits flipped by $\mathtt{B}_n$. If $b_0 == b_1$, then G_n and G_{n+1} are identical and $\mathtt{B}_n$ simulates this game for $\mathtt{A}$. Let us now assume that $b_0 \neq b_1$. If $\mathsf{b}_1 == b_0$, then $\mathtt{B}_n$ simulates game G_n for $\mathtt{A}$. If $\mathsf{b}_1 == b_1$, then $\mathtt{B}_n$ simulates game G_{n+1} for $\mathtt{A}$.

The running time of $\mathtt{B}_n$ is roughly the running time of $\mathtt{A}$. We conclude that, for all $n \in [0, \ell-1]$, we have:

$$\Pr[\mathtt{A}^{\mathsf{G}_n} \Rightarrow 1] - \Pr[\mathtt{A}^{\mathsf{G}_{n+1}} \Rightarrow 1] \leq \frac{1}{2} \cdot \mathbf{Adv}^{\mathsf{1\text{-}CDA}}_{\mathsf{param}}(t). \quad (22)$$

From (21) and (22) we get

$$\mathbf{Adv}^{\ell\text{-}\mathsf{CDA}}_{\mathsf{param}}(\mathtt{A}) \leq \ell \cdot \mathbf{Adv}^{\mathsf{1\text{-}CDA}}_{\mathsf{param}}(t).$$

From Lemma 9, we get the theorem. □

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 work. The authors would also like to thanks Jacques Traoré for useful feedbacks and discussions about an earlier version of this work. This work has been partially funded by the European FP7 FIDELITY (SEC-2011-284862) and EKSISTENZ (SEC-2013-607049) projects. 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] Paulo Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography, SAC'05*, *LNCS* 3897, pages 319–331. Springer, 2006.

[2] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Advances in Cryptology – EUROCRYPT'03*, *LNCS* 2656, pages 614–629. Springer, 2003.

[3] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *ACM Conference on Computer and Communications Security – CCS'06*, pages 390–399. ACM Press, 2006.

[4] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *Advances in Cryptology, EUROCRYPT'06*, *LNCS* 4004, pages 409–426. Springer, 2006.

[5] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In *Topics in Cryptology – CT-RSA'05*, volume 3376 of *LNCS*, pages 136–153. Springer, 2005.

[6] Jens Bender, Özgür Dagdelen, Marc Fischlin, and Dennis Kügler. Domain-specific pseudonymous signatures for the German identity card. In *Information Security Conference – ISC'12*, volume 7483 of *LNCS*, pages 104–119. Springer, 2012.

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

[8] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.

[9] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In *ACM Conference on Computer and Communications Security – CCS'04*, pages 168–177. ACM Press, 2004.

[10] Ernest Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *ACM Conference on Computer and Communications Security – CCS'04*, pages 132–145. ACM Press, 2004.

[11] Julien Bringer, Hervé Chabanne, Roch Lescuyer, and Alain Patey. Efficient and strongly secure dynamic domain-specific pseudonymous signatures for ID documents. In *FC'14*, *LNCS* 8437, pages 255–272. Springer, 2014.

[12] Julien Bringer, Hervé Chabanne, and Alain Patey. Collusion-resistant domain-specific pseudonymous signatures. In *Network and System Security Conference, NSS'13*, *LNCS* 7873, pages 649–655. Springer, 2013.

[13] Julien Bringer, Hervé Chabanne, and Alain Patey. Cross-unlinkable hierarchical group signatures. In *Public Key Infrastructures, Services and Applications, EuroPKI'12*, volume 7868 of *LNCS*, pages 161–177. Springer, 2013.

[14] Bundesamt für Sicherheit in der Informationstechnik (BSI). Advanced security mechanisms for machine readable travel documents and eIDAS token, part 2 – protocols for electronic identification, authentication and trust services (eIDAS), technical guideline TR-03110-2, v2.20, February 2015.

[15] Jan Camenisch, Maria Dubovitskaya, Robert Enderlein, Anja Lehmann, Gregory Neven, Christian Paquin, Franz-Stefan Preiss. Concepts and languages for privacy-preserving attribute-based authentication. *J. Inf. Sec. Appl.*, 19(1):25–44, 2014.

[16] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology – CRYPTO'04*, volume 3152 of *LNCS*, pages 56–72. Springer, 2004.

[17] Ivan Damgård. On Sigma-protocols v.2, 2010. `http://www.daimi.au.dk/~ivan/Sigma.pdf` consulted the 2015/09/01.

[18] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In *VIETCRYPT'06*, volume 4341 of *LNCS*, pages 193–210. Springer, 2006.

[19] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. on Information Theory*, 22(6):644–654, 1976.

[20] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology – CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, 1987.

[21] Steven Galbraith, Kenneth Paterson, Nigel Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[22] Miroslaw Kutylowski and Jun Shao. Signing with multiple ID's and a single key. In *38th CCNC*, pages 519–520. IEEE, 2011.

[23] Anna Lysyanskaya, Ronald Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *Selected Areas in Cryptography – SAC'99*, *LNCS* 1758, pages 184–199. Springer, 2000.