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

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

[1] Morpho, Issy-Les-Moulineaux, France
[2] Télécom ParisTech, Paris, France
Identity and Security Alliance (The Morpho and Télécom ParisTech Research Center)
{julien.bringer|herve.chabanne|roch.lescuyer|alain.patey}@morpho.com

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

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

## 1 Introduction

*Authentication with ID documents.* Recently, the German BSI agency introduced several security mechanisms regarding the use of ID documents for authentication purposes [10]. In such situations, a *Machine Readable Travel Document* (MRTD) connects to a Service Provider (SP) through a reader (for concreteness, one might see the MRTD as a passport). The security mechanisms of [10] can be summarized as follows. First of all, during the PACE protocol (*Password Authenticated Connection Establishment*), the MRTD and the reader establish a secure channel. Then, during the EAC protocol (*Extended Access Control*), the MRTD and the SP authenticate each other through another secure channel.

The reader transfers the exchanged messages. At last, during the (optional) RI protocol (*Restricted Identification*), the MRTD gives its pseudonym for the service to the SP. This pseudonym enables the SP to link users inside its service. However, across the services, users are still unlinkable. The latter property is called *cross-domain anonymity*. This property is interesting for many applications, since it offers at the same time privacy for the users and usability for the service provider, who might not want to have fully anonymous users, but might want them to use an account to give them more personal services (*e.g.* bank accounts, TV subscriptions, *etc.*).

For authentication purposes, giving pseudonyms is insufficient since the authenticity of the pseudonym is not guaranteed. For this reason, subsequent works [6, 5] adopt a "signature mode" for the RI protocol. This signature mode can be described as follows.

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

The contribution of [6] is to propose this signature mode and to present an efficient construction based on groups of prime order (without pairings). Their construction relies on a very strong hypothesis regarding the tamperproofness of the MRTD. In fact, recovering two users' secrets enables to compute the key of the certification authority. To deal with this concern, the authors of [5] propose to introduce group signatures into this signature mode. In addition to providing strong privacy properties, group signatures provide collusion resistance even if several users' secrets do leak.

*Our contributions.* The authors of [5] claim that the security model of group signatures directly gives a security model for DSPS, and, in fact, leave imprecise the definition of the DSPS security properties. Moreover, the model of [6] only concerns the static case, and their anonymity definition is flawed. So a security model for dynamic DSPS as such has to be supplied. Our first contribution is then a clean security model for *dynamic domain-specific pseudonymous signatures*.

This first contribution highlights the fact that, in some sense, using group signatures is "too strong" for constructing DSPS signatures. Following this intuition, we provide a new construction that is more efficient than the one of [5], while achieving the same strong security and privacy properties. Our second contribution is then an efficient proven secure dynamic DSPS with short signatures.

Finally, we concentrate on the use of our DSPS scheme in the RI protocol for MRTD private authentication. Our construction is based on bilinear pairings, but, as a first advantage, no pairing computation is necessary during the signature. However, we can go a step further, by taking advantage of the computational power of the reader. If some computations are delegated to the reader, then the chip only performs computation in a group of prime order. This is a

valuable practical advantage since existing chips might be used. Otherwise, one needs to deploy *ad hoc* chips, which has an industrial cost.

*Related notions.* As a privacy-preserving cryptographic primitive, a DSPS scheme shares some properties with other primitives. We now discuss common points and differences. DSPS schemes share some similarities with *group signatures with verifier local revocation* (VLR) [9] in the sense that, in both primitives, the revocation is done on the verifier's side. However, the anonymity properties are not the same: group signatures are always unlinkable, whereas DSPS achieve some partial linkability. Moreover, one can establish a parallel with the notion of *cross-unlinkable VLR group signatures* [4], where users employ several group signatures for several domains such that the signatures are unlinkable across domains. Within a domain, the group signatures are however unlinkable, which is too strong for the context of DSPS.

The difference between DSPS and *pseudonym systems* [14] or *anonymous credential systems* [11] is that DSPS-pseudonyms are deterministic whereas anonymous credentials pseudonyms must be unlinkable. In a DSPS scheme, the unlinkability is required across domains only, which is a weaker notion compared to anonymity in anonymous credentials. In fact, the anonymity of DSPS is a weaker notion compared to the anonymity of group signatures, as noticed above, and (multi-show) anonymous credentials are often constructed through group signatures techniques [11].

A point of interest is to clarify the relation between *pseudonymous signatures* and *direct anonymous attestations* (DAA) [2]. A DAA scheme might be seen (cf. [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 pseudonyms, and where the underlying group signature is replaced by a VLR group signature. The VLR group signatures introduce revocation concerns that are away from DAA. Moreover, in the ID document use-case, the MRTD/reader pair might be seen as the TPM/host pair of DAA scheme. However, both primitives remain distinct. The choice of pseudonyms in DSPS is more restrictive than the choice of the basename in DAA. Moreover, the host always embeds the same chip, but a MRTD is not linked to a specific reader, and might authenticate in front of several readers. Both differences impact the DSPS notion of anonymity.

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

## 2 Definition and security properties of dynamic DSPS

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`, `DomainKeyGen`, `Join`, `Issue`, `NymGen`, `Sign`, `Verify`, `DomainRevoke`, `Revoke`} as described below. By convention, users are enumerated here with indices $i \in \mathbb{N}$ and domains with indices $j \in \mathbb{N}$.

`Setup`. On input a security parameter $\lambda$, this algorithm computes global parameters $\mathsf{gpk}$ and an issuing secret key $\mathsf{isk}$. A message space $\mathcal{M}$ is specified. The sets $\mathcal{U}$ and $\mathcal{D}$ are initially empty. The global parameters $\mathsf{gpk}$ are implicitly given to all algorithms, if not explicitly specified. We note $(\mathsf{gpk}, \mathsf{isk}) \leftarrow \mathtt{Setup}(1^\lambda)$.

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

`Join` $\leftrightarrow$ `Issue`. This protocol involves a user $i \in \mathcal{U}$ and the issuing authority $IA$. `Join` takes as input the global parameters $\mathsf{gpk}$. `Issue` takes as input the global parameters $\mathsf{gpk}$ and the issuing secret key $\mathsf{isk}$. At the end of the protocol, the user $i$ gets a secret key $\mathsf{usk}_i$ and the issuing authority $IA$ gets a revocation token $\mathsf{rt}_i$. We note $\mathsf{usk}_i \leftarrow \mathtt{Join}(\mathsf{gpk}) \leftrightarrow \mathtt{Issue}(\mathsf{gpk}, \mathsf{isk}) \rightarrow \mathsf{rt}_i$.

`NymGen`. On input the global parameters $\mathsf{gpk}$, a public key $\mathsf{dpk}_j$ for a domain $j \in \mathcal{D}$ and a secret key $\mathsf{usk}_i$ of a user $i \in \mathcal{U}$, this *deterministic* algorithm outputs a pseudonym $\mathsf{nym}_{ij}$ for the user $i$ usable in the domain $j$. We note $\mathsf{nym}_{ij} \leftarrow \mathtt{NymGen}(\mathsf{gpk}, \mathsf{dpk}_j, \mathsf{usk}_i)$.

`Sign`. On input the global parameters $\mathsf{gpk}$, a public key $\mathsf{dpk}_j$ of a domain $j \in \mathcal{D}$, a user secret key $\mathsf{usk}_i$ of a user $i \in \mathcal{U}$, a pseudonym $\mathsf{nym}_{ij}$ for the user $i$ and the domain $j$ and a message $m \in \mathcal{M}$, this algorithm outputs a signature $\sigma$. We note $\sigma \leftarrow \mathtt{Sign}(\mathsf{gpk}, \mathsf{dpk}_j, \mathsf{usk}_i, \mathsf{nym}_{ij}, m)$.

`Verify`. On input the global parameters $\mathsf{gpk}$, a public key $\mathsf{dpk}_j$ of a domain $j \in \mathcal{D}$, a pseudonym $\mathsf{nym}_{ij}$, a message $m \in \mathcal{M}$, a signature $\sigma$ and the revocation list $RL_j$ of the domain $j$, this algorithm outputs a decision $d \in \{\mathsf{accept}, \mathsf{reject}\}$. We note $d \leftarrow \mathtt{Verify}(\mathsf{gpk}, \mathsf{dpk}_j, \mathsf{nym}_{ij}, m, \sigma, RL_j)$.

`DomainRevoke`. On input the global parameters $\mathsf{gpk}$, a public key $\mathsf{dpk}_j$ of a domain $j \in \mathcal{D}$, an auxiliary information $\mathsf{aux}_j$ and the revocation list $RL_j$ of the domain $j$, this algorithm outputs an updated revocation list $RL'_j$. We note $RL'_j \leftarrow \mathtt{DomainRevoke}(\mathsf{gpk}, \mathsf{dpk}_j, \mathsf{aux}_j, RL_j)$.

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

We consider the dynamic case where both users and domains may be added to the system. Users might also be revoked. Moreover, the global revocation may concern all the domains at a given point, or a subset of them. A global revocation protocol enabling to revoke the user $i$ from every domain is implicit here: it suffices to publish $\mathsf{rt}_i$. Using $\mathsf{rt}_i$ and public parameters, anyone can revoke user $i$, even for domains that will be added later. Pseudonyms are deterministic. This implies the existence of an implicit `Link` algorithm to link signatures inside a specific domain. On input a domain public key $\mathsf{dpk}$ and two triples $(\mathsf{nym}, m, \sigma)$ and $(\mathsf{nym}', m', \sigma')$, this algorithm outputs 1 if $\mathsf{nym} = \mathsf{nym}'$ and outputs 0 otherwise. This also gives implicit procedures for the service providers to put the users on a white list or a black list, without invoking the `Revoke` or `DomainRevoke` algorithms: it suffices to publish the pseudonym of the concerned user.

**Security definitions**. To be secure, a DSPS scheme should satisfy the *correctness*, *cross-domain anonymity*, *seclusiveness* and *unforgeability* properties. Informally, a DSPS scheme is (i) correct if honest and non-revoked users are accepted (signature correctness) and if the revocation of honest users effectively blacklists them (revocation correctness), (ii) cross-domain anonymous if signatures are unlinkable but within a specific domain, (iii) seclusive if it is impossible to exhibit a valid signature without involving a single existing user, and (iv) unforgeable if corrupted authority and domains owners cannot sign on behalf of an honest user. Let us now formalize each of these intuitions. The definition of correctness does not make difficulties and is postponed to the full version [3].

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

*Seclusiveness.* Informally, a DSPS scheme achieves seclusiveness if, by similarity with the traceability property of the group signatures, an adversary $A$ is unable to forge a valid signature that cannot "trace" to a valid user. In the group signature case, there is an opening algorithm, which enables to check if a valid user produced a given signature. However, there is no opening here, so one might ask how to define "tracing" users. Nevertheless, the management of the revocation tokens allows to correctly phrase the gain condition, as in VLR group signatures [9], providing that we take into account the presence of the pseudonyms. At the end of the game, we revoke all users on the domain supplied by the adversary. If the signature is still valid, then the adversary has won the game. Indeed, in this case, the signature does not involve any existing user. (This is an analogue of "the opener cannot conclude" in the group signature case).

$\mathsf{Seclusiveness}_A^{\mathrm{DSPS}}(\lambda)$
- $(\mathsf{gpk}, \mathsf{isk}) \leftarrow \mathrm{DSPS.}\mathtt{Setup}(1^\lambda)$ ; $\mathcal{D}, \mathcal{HU}, \mathcal{CU} \leftarrow \{\}$
- $\mathcal{O} \leftarrow \{\mathtt{AddDomain}(\cdot), \mathtt{AddUser}(\cdot), \mathtt{CorruptUser}(\cdot), \mathtt{UserSecretKey}(\cdot), \mathtt{Sign}(\cdot,\cdot,\cdot), \mathtt{ReadRegistrationTable}(\cdot), \mathtt{SendToIssuer}(\cdot,\cdot)\}$
- $(\mathsf{dpk}_*, \mathsf{nym}_*, m_*, \sigma_*) \leftarrow A^{\mathcal{O}}(\mathsf{gpk})$
- Find $j \in \mathcal{D}$ such that $\mathsf{dpk}_* := \mathbf{dpk}[j]$. If no match is found, then return 0.
- Return 1 if for all $i \in \mathcal{U}$, either $\mathbf{rt}[i] = \perp$ or $\mathrm{DSPS.}\mathtt{Verify}(\mathsf{gpk}, \mathsf{dpk}_*, \mathsf{nym}_*, m_*, \sigma_*, RL) = \mathsf{accept}$ where $RL := \mathrm{DSPS.}\mathtt{DomainRevoke}\big(\mathsf{gpk}, \mathsf{dpk}_*, aux, \mathbf{RL}[j]\big)$ and $aux := \mathrm{DSPS.}\mathtt{Revoke}(\mathsf{gpk}, \mathbf{rt}[i], \{\mathsf{dpk}_*\})$.

A DSPS scheme achieves *seclusiveness* if the probability for a polynomial adversary $A$ to win the $\mathsf{Seclusiveness}_A^{\mathrm{DSPS}}$ game is negligible (as a function of $\lambda$).

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

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

A DSPS scheme achieves *unforgeability* if the probability for a polynomial adversary $A$ to win the $\mathsf{Unforgeability}_A^{\mathrm{DSPS}}$ game is negligible (as a function of $\lambda$).

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

$\mathsf{Anonymity}_A^{\mathrm{DSPS}}(\lambda)$
- $(\mathsf{gpk}, \mathsf{isk}) \leftarrow \mathrm{DSPS.}\mathtt{Setup}(1^\lambda)$ ; $\mathcal{D}, \mathcal{HU}, \mathcal{CU}, \mathcal{CH} \leftarrow \{\}$ ; $b_A, b_B \xleftarrow{\$} \{0,1\}$
- $\mathcal{O} \leftarrow \{\mathtt{AddDomain}(\cdot), \mathtt{AddUser}(\cdot), \mathtt{CorruptUser}(\cdot), \mathtt{UserSecretKey}(\cdot), \mathtt{Revoke}(\cdot,\cdot), \mathtt{DomainRevoke}(\cdot,\cdot), \mathtt{Nym}(\cdot,\cdot), \mathtt{NymDomain}(\cdot), \mathtt{NymSign}(\cdot,\cdot,\cdot), \mathtt{SendToIssuer}(\cdot,\cdot), \mathtt{Challenge}(b_A, b_B, \cdot,\cdot,\cdot,\cdot)\}$
- $b' \leftarrow A^{\mathcal{O}}(\mathsf{gpk})$
- Return 1 if $b' == (b_A == b_B)$, and return 0 otherwise.


**Fig. 1.** Oracles provided to adversaries

A DSPS scheme achieves *cross-domain anonymity* if the probability for a polynomial adversary $A$ to win the $\mathsf{Anonymity}_A^{\mathrm{DSPS}}$ game is negligible[3][4].

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

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

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

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

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

*Comparison to previous security models.* First, the model of [6] is static: all users and domains are created at the beginning of the games, while our security games are all dynamic. Second, let us focus on the cross-domain anonymity and show that their definition is flawed. The adversary is given all pseudonyms and all domain parameters. The left-or-right challenge takes as input two pseudonyms for the same domain and a message and outputs a signature on this message by one of the corresponding users. A simple strategy to win the game, independently of the construction, is to verify this signature using both pseudonyms: it will be valid for only one of them. This observation motivates our choice for our challenge output to be a pair of pseudonyms and not a pair of signatures, since it is easy to verify correctness using pseudonyms. Moreover, in their game, both pseudonyms queried to the challenge oracle are in the same domain, which does not fit the *cross-domain* anonymity, while our challenge involving two different domains does. Third, the model of [6] does not allow for collusions: the adversary can be given at most one user secret key (indeed, with their construction, using two users' secret keys, one can recover the issuing keys)[5].

The model of [5] 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

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

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]). The model also lacks from a precise description of the oracles, thus leaving looseness on what are the exact inputs and outputs. Our model is more precise and separated from the model of group signatures, which leads, as we will see in the following, to a more efficient construction.

## 3 An efficient construction of dynamic DSPS

In this section, we present an efficient construction of dynamic DSPS we call the D scheme and prove it secure in the sense of the previous Section in the random oracle model. Our construction makes use of bilinear pairings. A bilinear environment is given by a tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ where $p$ is a prime number, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are three groups of order $p$ (in multiplicative notation) and $e$ is a bilinear and non-degenerate application $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The property of bilinearity states that for all $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$, $a, b \in \mathbb{Z}_p$, we have $e(g^a, h^b) = e(g, h)^{ab} = e(g^b, h^a)$. The property of non-degeneracy states that for all $g \in \mathbb{G}_1 \setminus \{\mathbf{1}_{\mathbb{G}_1}\}$, $h \in \mathbb{G}_2 \setminus \{\mathbf{1}_{\mathbb{G}_2}\}$, $e(g, h) \neq \mathbf{1}_{\mathbb{G}_T}$. Bilinear environments may be symmetric if $\mathbb{G}_1 = \mathbb{G}_2$ or asymmetric if $\mathbb{G}_1 \neq \mathbb{G}_2$. Let us now describe our scheme.

$\texttt{Setup}(1^\lambda)$

1. Generate an asymmetric bilinear environment $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$
2. Pick generators $g_1, h \xleftarrow{\$} \mathbb{G}_1 \setminus \{\mathbf{1}_{\mathbb{G}_1}\}$ and $g_2 \xleftarrow{\$} \mathbb{G}_2 \setminus \{\mathbf{1}_{\mathbb{G}_2}\}$
3. Pick $\gamma \in \mathbb{Z}_p$ ; Set $w := {g_2}^\gamma$
4. Choose a hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^\lambda$
5. Return $\mathsf{gpk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, h, g_2, w, \mathcal{H})$ ; $\mathsf{isk} := \gamma$

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

1. Pick $r \xleftarrow{\$} \mathbb{Z}_p^*$ ; Set $RL_j \leftarrow \{\}$ ; Return $\mathsf{dpk}_j := {g_1}^r$ ; $RL_j$

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

1. $[i]$ Pick $f' \xleftarrow{\$} \mathbb{Z}_p$ ; Set $F' := h^{f'}$
2. $[i]$ Compute $\Pi := \mathsf{PoK}\{C = \mathsf{Ext\text{-}Commit}(f') \wedge \mathsf{NIZKPEqDL}(f', C, F', h)\}$[6]
3. $[U \to IA]$ Send $F, \Pi$ $[IA]$ Check $\Pi$
4. $[IA]$ Pick $x, f'' \in \mathbb{Z}_p$ ; Set $F := F' \cdot h^{f''}$ ; $A := \left(g_1 \cdot F\right)^{\frac{1}{\gamma + x}}$ ; $Z := e(A, g_2)$
5. $[U \leftarrow IA]$ Send $f'', A, x, Z$
6. $[i]$ Set $f := f' + f''$ ; Check $e(A, {g_2}^x \cdot w) \stackrel{?}{=} e(g_1 \cdot h^f, g_2)$

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

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

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

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

The `Sign` procedure is obtained by applying the Fiat-Shamir heuristic [13] to a proof of knowledge of a valid user's certificate (we explicitly give this proof of knowledge in Appendix A.1). More precisely, a signer proves knowledge of $(f,(A,x))$ such that $A = \left(g_1 \cdot h^f\right)^{\frac{1}{\gamma+x}}$ and $\mathsf{nym} = h^f \cdot \mathsf{dpk}^x$.

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

1. Parse $\mathsf{usk}$ as $(f, A, x, Z)$
2. Pick $a, r_a, r_f, r_x, r_b, r_d \xleftarrow{\$} \mathbb{Z}_p$ ; Set $T := A \cdot h^a$
3. Set $R_1 := h^{r_f} \cdot \mathsf{dpk}^{r_x}$ ; $R_2 := \mathsf{nym}^{r_a} \cdot h^{-r_d} \cdot \mathsf{dpk}^{-r_b}$
4. Set $R_3 := Z^{r_x} \cdot e(h, g_2)^{a \cdot r_x - r_f - r_b} \cdot e(h, w)^{-r_a}$
5. Compute $c := \mathcal{H}(\mathsf{dpk} \| \mathsf{nym} \| T \| R_1 \| R_2 \| R_3 \| m)$
6. Set $s_f := r_f + c \cdot f$ ; $s_x := r_x + c \cdot x$ ; $s_a := r_a + c \cdot a$ ; $s_b := r_b + c \cdot a \cdot x$ ; $s_d := r_d + c \cdot a \cdot f$
7. Return $\sigma := (T, c, s_f, s_x, s_a, s_b, s_d)$

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

1. If $\mathsf{nym} \in RL$, then return $\mathsf{reject}$ and abort.
2. Parse $\sigma$ as $(T, c, s_f, s_x, s_a, s_b, s_d)$
3. Set $R'_1 := h^{s_f} \cdot \mathsf{dpk}^{s_x} \cdot \mathsf{nym}^{-c}$ ; $R'_2 := \mathsf{nym}^{s_a} \cdot h^{-s_d} \cdot \mathsf{dpk}^{-s_b}$
4. Set $R'_3 := e(T, g_2)^{s_x} \cdot e(h, g_2)^{-s_f - s_b} \cdot e(h, w)^{-s_a} \cdot \left[e(g_1, g_2) \cdot e(T, w)^{-1}\right]^{-c}$
5. Compute $c' := \mathcal{H}(\mathsf{dpk} \| \mathsf{nym} \| T \| R'_1 \| R'_2 \| R'_3 \| m)$
6. Return $\mathsf{accept}$ if $c = c'$, otherwise return $\mathsf{reject}$.

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

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

$\mathtt{DomainRevoke}(\mathsf{gpk}, \mathsf{dpk}_j, \mathsf{aux}_j, RL_j)$[7]

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

We now sketch a proof of the following theorem. A full proof can be found in [3].

**Theorem 1.** *The* `D` *scheme achieves seclusiveness, unforgeability and cross-domain anonymity in the sense of Section 2 in the random oracle model under the* DL*,* $q$-SDH *and* DDH *assumptions.*

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

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

$q$-Strong Diffie-Hellman $q-$SDH. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a bilinear environment, $h_1 \xleftarrow{\$} \mathbb{G}_1$, $h_2 \xleftarrow{\$} \mathbb{G}_2$ and $\theta \xleftarrow{\$} \mathbb{Z}_p$. Given $(h_1, {h_1}^{\theta}, {h_1}^{\theta^2}, \ldots, {h_1}^{\theta^q}, h_2, {h_2}^{\theta}) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2$, find a pair $(c, {g_1}^{1/(\theta+c)}) \in \mathbb{Z}_p \setminus \{-\theta\} \times \mathbb{G}_1$.

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

We first show that, under a chosen-message attack, in the random oracle model, it is computationally impossible to produce a valid $\mathtt{D}$ signature $\sigma := (T, c, s_f, s_x, s_a, s_b, s_d)$ without the knowledge of a valid certificate $(f, A, x, Z)$. In other words, from a valid signature, we can extract a valid certificate. This "extraction step" is standard when signature schemes are built by applying the Fiat-Shamir heuristic [13] to a given $\Sigma$-protocol (cf. [16, 15, 12]).

**Proof of seclusiveness**. In the random oracle model, the $\mathtt{D}$ scheme achieves seclusiveness in the sense of Section 2 if the $\mathsf{SDH}$ problem is hard. Let $(h_1, {h_1}^{\theta}, {h_1}^{\theta^2}, \ldots, {h_1}^{\theta^q}, h_2, {h_2}^{\theta}) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2$ be a $\mathsf{SDH}$ instance on a bilinear environment $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$. We build an algorithm $B$ that outputs $(c, {g_1}^{1/(\theta+c)})$, for a $c \in \mathbb{Z}_p \setminus \{-\theta\}$, from an adversary $A$ against the seclusiveness of our scheme.

*Parameters.* $B$ picks $k \xleftarrow{\$} [1, q]$, $x_1, \ldots, x_q, s_1, \ldots, s_q \xleftarrow{\$} \mathbb{Z}_p$, computes $g_2 := h_2$, $w := ({h_2}^{\theta}) \cdot {h_2}^{-x_k}$. For $\{x_1, \ldots, x_q\} \in \mathbb{F}_p$, define polynomials $P$, $P_m$ and $P_m^-$ for $m \in [1, q]$ on $\mathbb{F}_p[X]$ by $P := \prod_{n=1}^{q}(X + x_n - x_k)$, $P_m := \prod_{n=1, n\neq m}^{q}(X + x_n - x_k)$, $P_m^- := \prod_{n=1, n\neq m, n\neq k}^{q}(X + x_n - x_k)$. Expanding $P$ on $\theta$, we get $P(\theta) = \sum_{n=0}^{q} a_n \theta^n$ for some $\{a_n\}_{n=0}^{q}$ depending on the $x_n$. Since $B$ knows ${h_1}^{\theta^n}$ from the $q$-$\mathsf{SDH}$ challenge, $B$ is able to compute ${h_1}^{P(\theta)}$ without the knowledge of $\theta$. $B$ picks $\alpha \xleftarrow{\$} \mathbb{Z}_p$, $\beta \xleftarrow{\$} \mathbb{Z}_p^*$, sets $g_1 := {h_1}^{\beta(\alpha P(\theta) - s_k P_k(\theta))}$, $h := {h_1}^{\beta P_k(\theta)}$ and gives $(e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, h, g_2, w, \mathcal{H})$ to $A$.

*Simulating the issuing algorithm.* Let $\mathtt{Aux}$ be the following sub-routine, taking as input $(f', \mathsf{ctr}) \in \mathbb{Z}_p \times \mathbb{N}$ and outputting $(f'', A, x, Z)$ as in the fourth step of the $\mathtt{D.Issue}$ algorithm. $\mathsf{ctr}$ is a counter for the queries. $B$ sets $A_{\mathsf{ctr}} := {h_1}^{\beta(\alpha P_{\mathsf{ctr}}(\theta) + P_{\mathsf{ctr}}^-(\theta)(s_{\mathsf{ctr}} - s_k))}$ and returns $(s_{\mathsf{ctr}} - f', A_{\mathsf{ctr}}, x_{\mathsf{ctr}}, e(A_{\mathsf{ctr}}, g_2))$.

*Simulating the oracles.* A counter is set $\mathsf{ctr} := 0$. When $A$ asks for adding a new honest user, $B$ sets $\mathsf{ctr} := \mathsf{ctr} + 1$, picks $f' \xleftarrow{\$} \mathbb{Z}_p$, calls the $\mathtt{Aux}$ procedure on input $(f', \mathsf{ctr})$, gets $(f''_{\mathsf{ctr}}, A_{\mathsf{ctr}}, x_{\mathsf{ctr}}, Z_{\mathsf{ctr}})$, records $\mathbf{usk}[\mathsf{ctr}] := (f' + f''_{\mathsf{ctr}}, A_{\mathsf{ctr}}, x_{\mathsf{ctr}}, Z_{\mathsf{ctr}})$ and $\mathbf{rt}[\mathsf{ctr}] := (h^{f' + f''_{\mathsf{ctr}}}, x_{\mathsf{ctr}})$. When $A$ interacts with the issuer as a corrupted user, $B$ sets $\mathsf{ctr} := \mathsf{ctr} + 1$ and extracts $f'$ such that $F := h^{f'}$ thanks to the extraction key $\mathsf{ek}$. $B$ then calls the $\mathtt{Aux}$ procedure on the input $(f', \mathsf{ctr})$, and gets $(f''_{\mathsf{ctr}}, A_{\mathsf{ctr}}, f_{\mathsf{ctr}}, Z_{\mathsf{ctr}})$ back, which $B$ transfers to $A$. $B$ records $\mathbf{usk}[\mathsf{ctr}] := (f' + f''_{\mathsf{ctr}}, A_{\mathsf{ctr}}, x_{\mathsf{ctr}}, Z_{\mathsf{ctr}})$ and $\mathbf{rt}[\mathsf{ctr}] := (h^{f' + f''_{\mathsf{ctr}}}, x_{\mathsf{ctr}})$.

*Response.* $A$ eventually outputs $(\mathsf{dpk}_*, \mathsf{nym}_*, m_*, \sigma_*)$. If this is a non trivial response, then there exists $j \in \mathcal{D}$ such that $\mathsf{dpk}_* = \mathbf{dpk}[j]$. At this point, $B$ blacklists all users near $j$, by updating $\mathbf{RL}[j]$. For all $i \in \mathcal{U}$, we have (i) $\mathbf{usk}[i] \neq \perp$ and (ii) $\mathbf{rt}[i] \neq \perp$. If the response is valid, then $\mathtt{Verify}(\mathsf{gpk}, \mathsf{dpk}_*, \mathsf{nym}_*, m, \sigma, \mathbf{RL}[j]) = \mathsf{accept}$. This means that $B$ can extract a new certificate $(f_*, A_*, x_*, Z_*)$ in reasonable expecting time.

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

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

and $({A_*}^{s_k} \cdot {A_k}^{-f_*})^{\frac{1}{s_k - f_*}} = {h_1}^{\beta(\alpha P(\theta) - s_k P_k(\theta))\frac{1}{\theta}}$. By dividing $\beta(\alpha P(\theta) - s_k P_k(\theta))$ by $\theta$ we get $R$ and $Q$ such that $C := R(0) = -\beta s_k \prod_{n=1, n\neq k}^{q} (x_n - x_*)$ and $({A_*}^{s_k} \cdot {A_k}^{-f_*})^{\frac{1}{s_k - f_*}} = {h_1}^{\frac{C}{\theta} + Q(\theta)}$ where $C \neq 0$. $B$ computes ${h_1}^{1/\theta} := (({A_*}^{s_k} \cdot {A_k}^{-f_*})^{\frac{1}{s_k - f_*}} \cdot {h_1}^{-Q(\theta)})^{1/C}$, sets $c := 0$ and returns $(0, {h_1}^{1/\theta})$.

(B) $x_* \notin \{x_1, \ldots, x_q\}$. In particular, we have (iv) $x_n - x_* \neq 0$ for all $n \in [1, q]$. Let us now consider the quantity $\beta P_k(\theta)(\alpha\theta + f_* - s_k)$ as a polynomial $D$ in $\theta$. If we carry out the Euclidean division of $D$ by $(\theta + x_* - x_k)$, we get $Q$ and $R$ such that $D(\theta) = (\theta + x_* - x_k)Q(\theta) + R(\theta)$. As $(\theta + x_* - x_k)$ is a first degree polynomial $X - (x_k - x_*)$, we know that $R(\theta) = D(x_k - x_*)$, so $B$ can compute $C := R(\theta) = D(x_k - x_*) = \beta \left[\prod_{n=1, n\neq k}^{q} (x_n - x_*)\right] (\alpha(x_k - x_*) + f_* - s_k)$. We have $A_* = {h_1}^{Q(\theta) + \frac{C}{\theta + x_* - x_k}}$. $B$ can compute ${h_1}^{Q(\theta)}$ from the SDH challenge.

(B.I) $(f_* - s_k) \neq \alpha(x_* - x_k)$. In this case, $C \neq 0$ by (iv) and by the choice of $\beta$, so $B$ can compute ${g_1}^{\frac{1}{\theta + x_* - x_k}} = \left(A_* \cdot {g_1}^{-Q(\theta)}\right)^{\frac{1}{C}}$, set $c = x_* - x_k$, and return $(c, {g_1}^{1/(\theta + c)})$. (B.II) $(f_* - s_k) = \alpha(x_* - x_k)$. $B$ returns $\perp$ and aborts.

In [3] we show that $A$ outputs a valid forgery with probability $\epsilon$, then $B$ solves the SDH challenge with probability at least $\epsilon/2q$. □

**Proof of unforgeability**. In the random oracle model, the D scheme achieves unforgeability in the sense of the Section 2 if the DL problem is hard. Let $A$ be an adversary against the unforgeability of the D scheme. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a bilinear environment and $(g, \mathsf{H})$ be a discrete logarithm instance in $\mathbb{G}_1$. We construct an algorithm $B$ that computes $\theta := \log_g \mathsf{H}$.

*Parameters.* $B$ picks $g_1 \xleftarrow{\$} \mathbb{G}_1$, $g_2 \xleftarrow{\$} \mathbb{G}_2$, $\gamma \xleftarrow{\$} \mathbb{Z}_p$, sets $h := g$ and $w := {g_2}^{\gamma}$. $B$ gives parameters $\mathsf{gpk} := (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, h, g_2, w)$ to $A$. $B$ picks a random user $\mathsf{i} \in [1, q_U]$. In addition, $B$ generates parameters for the extractable commitment scheme Ext-Commit and the non-interactive proof system NIZKPEqDL.

*Simulating the oracles.* At each time $B$ interacts (as an honest user) with $A$ (as the corrupted issuing authority), $B$ follows the Join procedure, but for the i-th user. In the latter case, $B$ sets $F' := \mathsf{H}$, simulates $\Pi$ and gets $(f''_{\mathsf{i}}, A_{\mathsf{i}}, x_{\mathsf{i}}, Z_{\mathsf{i}})$ where $A_{\mathsf{i}} = (g_1 \cdot \mathsf{H} \cdot h^{f''_{\mathsf{i}}})^{\frac{1}{x_{\mathsf{i}} + \gamma}}$ for some $f''_{\mathsf{i}}$. $B$ does not know $f_{\mathsf{i}}$, but can compute $\mathsf{nym}_{\mathsf{i}j} := \mathsf{H} \cdot h^{f''_{\mathsf{i}}} \cdot {\mathsf{dpk}_j}^{x_{\mathsf{i}}}$ for all $j \in \mathcal{D}$. When $A$ asks for a signature, $B$ simulates a signature for i, other signatures are normally computed.

*Response.* A play of $A$ gives a valid and non trivial $(\mathsf{dpk}_*, \mathsf{nym}_*, m_*, \sigma_*)$. Then (i) we can find a domain $j$ such that $\mathsf{dpk}_* = \mathbf{dpk}[j]$ and an honest user $i$ with consistent values $\mathsf{nym}_{ij} \in \mathbf{nym}[i][j]$, $(F_i, x_i) \in \mathbf{rt}[i]$ and $(*, A_i, x_i, Z_i) \in \mathbf{usk}[i]$ such that $\mathsf{nym}_* = \mathsf{nym}_{ij} = F_i \cdot (\mathsf{dpk}_*)^{x_i}$, and (ii) we are able to extract a valid certificate $(f_*, A_*, x_*, Z_*)$ where, in particular, $\mathsf{nym}_* = h^{f_*} \cdot (\mathsf{dpk}_*)^{x_*}$. Since discrete representations in $\mathbb{G}_1$ are unique modulo $p$, then we have that $f_* = \log_g F_i$ (the pseudonym must be valid in a non trivial forgery) and $x_* = x_i$. With probability $\frac{1}{|\mathcal{U}|}$ we have $i = \mathsf{i}$, since i is independent of the view of $A$. This implies that $A_{\mathsf{i}} = A_*$ (a value $A$ is determined by $f$, $x$ and $\gamma$). Thus $A_* = (g_1 \cdot g^{f_*})^{\frac{1}{x_* + \gamma}} = (g_1 \cdot \mathsf{H} \cdot h^{f''_{\mathsf{i}}})^{\frac{1}{x_* + \gamma}}$ and we obtain $\theta = f_* - f''_{\mathsf{i}}$. □

**Proof of anonymity**. The D scheme achieves anonymity in the sense of Section 2 if the DDH problem is hard in $\mathbb{G}_1$. Let $q_U$ be the number of queries to `AddUser` and `SendToIssuer` and $q_D$ to `AddDomain`. Let $A$ be an $\epsilon$-adversary against the unforgeability of the D scheme. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a bilinear environment and $(g, \mathsf{A}, \mathsf{B}, \mathsf{C})$ a Diffie-Hellman instance in $\mathbb{G}_1$. We construct $B$ that decides whether $\mathsf{C}$ is the Diffie-Hellman of $\mathsf{A}$ and $\mathsf{B}$ *w.r.t.* $g$.

*Parameters.* The parameters $\mathsf{gpk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, h, g_2, w)$ for the D scheme are computed honestly, knowing $\mathsf{isk} = \gamma$, except that $g_1 := g$. $B$ picks two bits $b_A, b_B \xleftarrow{\$} \{0,1\}$, a random user $\mathsf{i} \xleftarrow{\$} [1, q_U]$ and a random domain $\mathsf{j} \xleftarrow{\$} [1, q_D]$.

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

`SimNym`$(i, j)$.
- (I) $i \neq \mathsf{i}$ and $j \neq \mathsf{j}$: $B$ gets $(f_i, x_i)$, $(\mathsf{dpk}_j, r_j)$ and sets $\mathsf{nym}_{ij} := h^{f_i} \cdot {g_1}^{r_j x_i}$.
- (II) $i = \mathsf{i}$ and $j \neq \mathsf{j}$: $B$ gets $f_i$, $(\mathsf{dpk}_j, r_j)$ and sets $\mathsf{nym}_{ij} := h^{f_i} \cdot \mathsf{A}^{r_j}$.
- (III) $i \neq \mathsf{i}$ and $j = \mathsf{j}$: $B$ gets $(f_i, x_i)$ and sets $\mathsf{nym}_{ij} := h^{f_i} \cdot \mathsf{B}^{x_i}$.
- (IV) $i = \mathsf{i}$ and $j = \mathsf{j}$: $B$ gets $f_i$ and sets $\mathsf{nym}_{ij} := h^{f_i} \cdot \mathsf{C}$.

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

Let us now estimate the advantage that $B$ has of solving the DDH challenge.

$$\begin{aligned}\mathbf{Adv}_B^{\mathsf{DDH}} &= \left|\Pr[B \Rightarrow \mathsf{true} | \mathsf{C} = \mathsf{DH}_g(\mathsf{A}, \mathsf{B})] - \Pr[B \Rightarrow \mathsf{true} | \mathsf{C} \text{ is random}]\right| \\ &= \left|\Pr[\mathbf{abort}] \cdot \mathbf{P}_1 + \Pr[\overline{\mathbf{abort}}] \cdot \mathbf{P}_2 - \Pr[\mathbf{abort}] \cdot \mathbf{P}_3 - \Pr[\overline{\mathbf{abort}}] \cdot \mathbf{P}_4\right|\end{aligned}$$

where $\mathbf{P}_1 := \Pr[B \Rightarrow \mathsf{true} | \mathbf{abort} \wedge \mathsf{C} = \mathsf{DH}_g(\mathsf{A}, \mathsf{B})]$, $\mathbf{P}_2 := \Pr[B \Rightarrow \mathsf{true} | \overline{\mathbf{abort}} \wedge \mathsf{C} = \mathsf{DH}_g(\mathsf{A}, \mathsf{B})]$, $\mathbf{P}_3 := \Pr[B \Rightarrow \mathsf{true} | \mathbf{abort} \wedge \mathsf{C} \text{ is random}]$ and $\mathbf{P}_4 := \Pr[B \Rightarrow \mathsf{true} | \overline{\mathbf{abort}} \wedge \mathsf{C} \text{ is random}]$. Due to the lack of space, we only give a bound and postpone its analysis to the full version of our paper [3]. We obtain:

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

where $q_C$, $q_S$ and $q_H$ are the number of queries to (resp.) `UserSecretKey`, `Sign` and $\mathcal{H}$. □

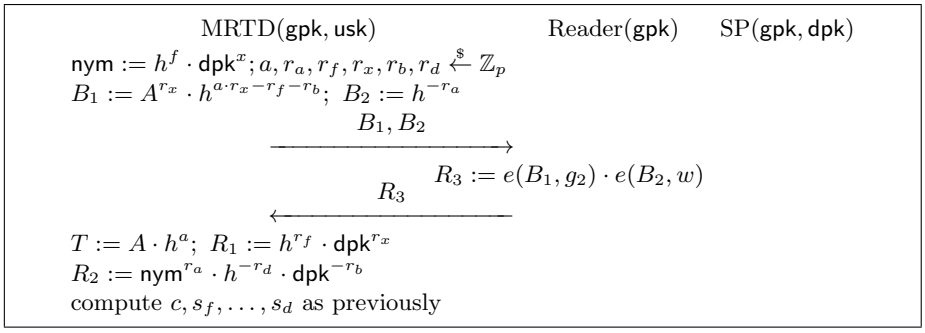


**Fig. 2.** Delegation of computation from the MRTD to the reader

## 4 Implementation considerations

*Signature size.* A signature $\sigma := (T, c, s_f, s_x, s_a, s_b, s_d)$ is composed of 1 element in $\mathbb{G}_1$, a challenge of size $\lambda$ and five scalars, which is particularly short for this level of security. By comparison, a signature of [5] is of the form $(B, J, K, T, c, s_f, s_x, s_a, s_b) \in {\mathbb{G}_1}^4 \times \{0,1\}^\lambda \times {\mathbb{Z}_p}^4$. The short group signature of [12] lies in $\in {\mathbb{G}_1}^4 \times \{0,1\}^\lambda \times {\mathbb{Z}_p}^4$ as well, which highlights the fact that we do not need the whole power of group signatures here.

*Pre-computations and delegation of computation.* In the `D` scheme, the issuer computes the element $Z := e(A, g_2)$ and adds it to the user secret key. Thanks to this pre-computation, the user avoids to compute any pairing. In the signature procedure, the user only computes (multi)-exponentiations in $\mathbb{G}_1$ and $\mathbb{G}_T$. This is an advantage if we consider that the user is a smart-card, as in the ID document use-case.

But we can go a step further by delegating some computation from the card to the reader. The MRTD interacts with the SP through the reader but, in the RI protocol, even in signature mode, the reader just transfers the messages. In our case however, we take advantage of the computational power of the reader. A proposal for this kind of delegation is given Figure 2. We obtain a piece of valuable advantages since there is no need to implement large groups operations (like operations in $\mathbb{G}_T$) in the MRTD. As a consequence, we do not need to develop specific chips for achieving those heavy computations, and existing chips can be used. We implemented our protocol on a PC. Following first estimations of a partial implementation on a chip, the overall signature and communication (including delegation) between the reader and the passport cost around 890ms, for equipment currently in use.

*Security of the delegation.* Of course, this delegation of computation must be done without compromising the security. In the DAA analysis of [7], a DAA scheme (with distinct host and TPM) is built upon a pre-DAA scheme (where TPM and host are not separated). However, our analysis differs, because the

MRTD is not linked to a single reader. Therefore we adapt our model. We add a pair of successive oracles (with a lock mechanism between their calls): $\mathsf{GetPreComp}(i, j, m)$, enabling a corrupted reader to obtain pre-computations from an honest user, and $\mathtt{Sign}'(i, j, D)$, where the same user produces a signature given a delegated computation $D$ supplied by the adversary. Formal definition are given in [3].

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

Finally, we adapt our proofs. First, in the anonymity proof, the challenger honestly computes signatures for all users, but $\mathsf{i}$, for which signatures are simulated. Then, we must show that, in each game, the challenger can simulate $B_1$, $B_2$ and $\sigma$ (a proof of this fact is given in Appendix A.2). In our construction, the adversary can compute $A$ from $B_2$ and $\sigma$. The fact that we can simulate signatures even in the cross-domain anonymity game shows that the knowledge of $A$ does not help linking users across domains.

## 5 Conclusion

In this paper, we supplied a clean security model for *dynamic domain-specific pseudonymous signatures*, and compared this notion with other privacy-friendly cryptographic primitives. We then highlighted the fact that, in some sense, using group signatures is "too strong" for constructing DSPS signatures. Following this intuition, we provided a new construction that is more efficient than the one of [5], while achieving the same strong security and privacy properties. Finally, we concentrated on the use of our DSPS scheme in the RI protocol for MRTD private authentication. Our construction might be implemented on existing chips if one takes advantage of the computational power of the reader. We supplied an analysis of such a delegation of computation.

**Acknowledgements.** The authors would like to thanks the anonymous reviewers for their valuable comments. This work has been partially funded by the European FP7 FIDELITY project (SEC-2011-284862). The opinions expressed in this document only represent the authors' view. They reflect neither the view of the European Commission nor the view of their employer.

## References


1. Dan Boneh, Xavier Boyen, Short signatures without random oracles, the SDH assumption in bilinear groups, *J. of Crypt.*, 21(2), pp. 149-177, 2008.
2. Ernest Brickell, Jan Camenisch, Liqun Chen, Direct anonymous attestation, *CCS'04*, pp. 132-145, ACM, 2004.
3. Julien Bringer, Hervé Chabanne, Roch Lescuyer, Alain Patey, Efficient and strongly secure dynamic domain-specific pseudonymous signatures for ID documents, *Full version available at* http://eprint.iacr.org/2014/067.
4. Julien Bringer, Hervé Chabanne, Alain Patey, Cross-unlinkable hierarchical group signatures, *EuroPKI 2012*, LNCS 7868, pp. 161-177, Springer, 2013.
5. Julien Bringer, Hervé Chabanne, Alain Patey, Collusion-resistant domain-specific pseudonymous signatures, *NSS'13*, LNCS 7873, pp. 649-655, Springer, 2013.
6. Jens Bender, Özgür Dagdelen, Marc Fischlin, Dennis Kügler, Domain-specific pseudonymous signatures for the German identity card, *ISC'12*, LNCS 7483, pp. 104-119, Springer, 2012.
7. David Bernhard, Georg Fuchsbauer, Essam Ghadafi, Nigel Smart, Bogdan Warinschi, Anonymous attestation with user-controlled linkability, *Int. J. Inf. Sec.*, 12(3), pp. 219-249, 2013.
8. Paulo Barreto, Michael Naehrig, Pairing-friendly elliptic curves of prime order, *SAC'05*, LNCS 3897, pp. 319-331, Springer, 2006.
9. Dan Boneh, Hovav Shacham, Group signatures with verifier-local revocation, *CCS'04*, pp. 168-177, ACM, 2004.
10. Bundesamt für Sicherheit in der Informationstechnik (BSI), *Advanced Security Mechanisms for Machine Readable Travel Documents, Part 2 – Extended Access Control Version 2 (EACv2), Password Authenticated Connection Establishment (PACE),, Restricted Identification (RI)*, TR-03110-2, March 2012.
11. Jan Camenisch, Anna Lysyanskaya, Signature schemes, anonymous credentials from bilinear maps, *CRYPTO'04*, LNCS 3152, pp. 56-72, Springer, 2004.
12. Cécile Delerablée, David Pointcheval, Dynamic fully anonymous short group signatures, *VIETCRYPT'06*, LNCS 4341, pp. 193-210, Springer, 2006.
13. Amos Fiat, Adi Shamir, How to prove yourself: practical solutions to identification, signature problems, *CRYPTO'86*, LNCS 263, pp. 186-194, Springer, 1987.
14. Anna Lysyanskaya, Ronald Rivest, Amit Sahai, Stefan Wolf, Pseudonym systems, *SAC'99*, LNCS 1758, pp. 184-199, Springer, 2000.
15. David Pointcheval, Jacques Stern, Security arguments for digital signatures, blind signatures, *J. of Crypt.*, 13(3), pp. 361-396, 2000.
16. Claus-Peter Schnorr, Efficient identification, signatures for smart cards, *CRYPTO'89*, LNCS 435, pp. 239-252, Springer, 1989.


## A Appendix

### A.1 A proof of knowledge of a valid certificate

Let $\mathtt{P}$ be the protocol Figure 3 for proving knowledge of $(f,(A,x))$ such that $A = \left(g_1 \cdot h^f\right)^{\frac{1}{\gamma+x}}$ and $\mathsf{nym} = h^f \cdot \mathsf{dpk}^x$. In [3], we show that (i) 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, and that (ii) there exists an extractor for the protocol $\mathtt{P}$.

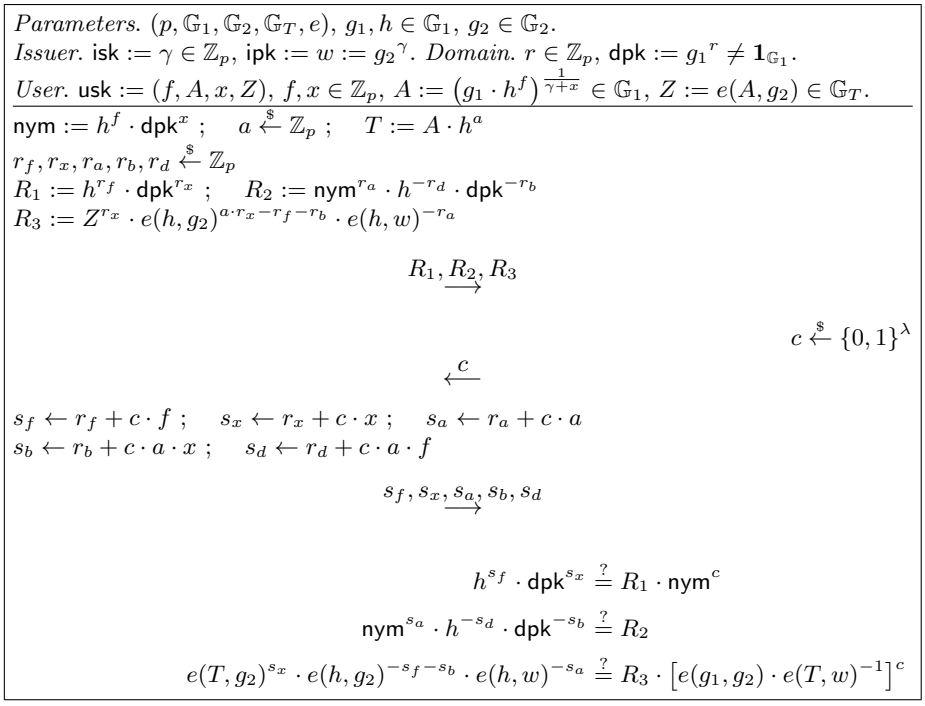


**Fig. 3.** The P protocol

### A.2 Simulation of signatures with delegated computation

We now adapt the proofs of our main scheme to the extended model of Section 4. We first simulate the `GetPreComp` step. In the *seclusiveness* proof, all signatures are honestly computed. In the *unforgeability* proof, if $i \neq \mathsf{i}$, then all signatures are honestly computed. If $i = \mathsf{i}$, then, given $\mathsf{H}$ (from the $\mathsf{DL}$ challenge), $A_\mathsf{i}$, $x_\mathsf{i}$ and $f_\mathsf{i}''$, $B$ picks $a, c, s_f, s_x, s_a, s_b \xleftarrow{\$} \mathbb{Z}_p$ and computes $B_1 := (A_\mathsf{i}^{-x_\mathsf{i}} \cdot \mathsf{H} \cdot h^{f_\mathsf{i}''})^c \cdot A^{s_x} \cdot h^{a \cdot s_x - s_f - s_b}$ and $B_2 := h^{a \cdot c - s_a}$. In the *cross-domain anonymity* proof, the challenger honestly computes signatures for all users, but $\mathsf{i}$, for which signatures are simulated. Given $\mathsf{A}$ (from the $\mathsf{DDH}$ challenge) and $f_\mathsf{i}$, $B$ picks $\alpha \xleftarrow{\$} \mathbb{Z}_p$. The same $\alpha$ is used in each signature, for consistency. Then, for each signature query, $B$ picks fresh values $a, c, s_f, s_x, s_a, s_b$ and computes $B_1 := (\mathsf{A}^{-\alpha} \cdot h^{f_\mathsf{i}})^c \cdot T^{s_x} \cdot h^{-s_f - s_b}$ and $B_2 := h^{a \cdot c - s_a}$. (The simulation is done as if $A_\mathsf{i} := {g_1}^{\alpha}$.)

We now simulate the `Sign'` oracle (identically in the three proofs). $B$ retrieves $m, B_1, B_2, c, s_a, s_x, s_a, s_b$ from the `GetPreComp` step. Whatever $D$ is ($D$ may not equal $e(B_1, g_2) \cdot e(B_2, w)$), $B$ picks $s_d \xleftarrow{\$} \mathbb{Z}_p$, computes $T$, $R_1$ and $R_2$ as usual and sets $c$ as the random oracle's value for the input $\mathsf{dpk} \| \mathsf{nym} \| T \| R_1 \| R_2 \| D \| m$. If $D$ is correct *w.r.t.* $B_1$ and $B_2$, then $B$ returns a valid signature. If not, then the signature is no longer valid but the response remains consistent *w.r.t.* $B_1$ and $B_2$. □