Towards Practical Security of Pseudonymous Signature on the BSI eIDAS Token*

Mirosław Kutyłowski, Lucjan Hanzlik**, Kamil Kluczniak* * *

Faculty of Fundamental Problems of Technology, Wrocław University of Technology
{firstname.secondname}@pwr.edu.pl

Abstract. In this paper we present an extension of Pseudonymous Signature introduced by the German Federal BSI authority as a part of technical recommendations for electronic identity documents.
Without switching to pairing friendly groups we enhance the scheme so that: (a) the issuer does not know the private keys of the citizen (so it cannot impersonate the citizen), (b) a powerful adversary that breaks any number of ID cards created by the Issuer cannot forge new cards that could be proven as fake ones, (c) deanonymization of the pseudonyms used by a citizen is a multi-party protocol, where the consent of each authority (e.g. a court) is necessary to reveal the identity of a user. (d) we propose extended features concerning fully anonymous signatures and a pragmatic revocation approach. (e) we present an argument for unlinkability (cross-domain anonymity) of the presented schemes.
In this way we make a step forwards to overcome the substantial weaknesses of the Pseudonymous Signature scheme. Moreover, the extension is on top of the original scheme with relatively small number of changes, following the strategy of reusing the previous schemes – thereby reducing the costs of potential technology update.

1 Introduction

Introduction of strict personal data protection rules in the European regulation GDPR [13] creates a market for technical solutions that attempt to satisfy the requirements by-design. One of the solutions is to change the status from personal data to non-personal data. It suffices to ensure that the data does not concern *identifiable person*. So changing explicit identification data by pseudonymous identity may solve the problem as long as the pseudonym cannot be linked with the pseudonym owner using available data. On the other hand, as already reflected by GDPR, the pseudonymization process should be reversible: in certain circumstances, like law enforcement, it might be necessary to link the pseudonym back to the original identity.

* This paper is an extended raport, where an initial version, "Pseudonymous Signature on eIDAS Token Implementation Based Privacy Threats", has been presented at ACISP 2016. New results contained in this paper have been supported by National Science Centre (Poland) under grant OPUS no 2014/15/B/ST6/02837

** current affiliation: Stanford University, USA

* * * current affiliation: CISPA, Saarbrücken, Germany

Domains specific pseudonyms.

There might be many strategies to replace the real identity with pseudonyms. Replacing the ID with a single pseudonym is very likely to be totally ineffective: high amount of data related to a single pseudonym enables eliminating all candidates but the pseudonym owner. So the best solution would be to use the same pseudonym only when it is necessary to indicate that the same (anonymous) person is concerned. These leads to the concept of a *domain*, which is a virtual environment where all activities of a physical person should be linked together (so a single pseudonym per person should be used), while activities within domain should not be linked with activities outside the domain.

The concept of domains appeared probably for the first time in Austria together with the concept of *Bürgerkarte*. The domains have been defined as different areas of public administration. The solution has been based on symmetric cryptography and enabled the user to create a domain specific password. The main invention was to prevent using the password from one domain in another domain. Otherwise the solution is not stronger than one could achieve with symmetric cryptography.

The Austrian concept of domains have been reused by German authorities with introduction of Restricted Identification on electronic personal identity documents [7]. A single (asymmetric) key stored on an ID card enables to create exactly one cryptographic pseudonym per domain, which serves as a password authenticating the request.

Domain specific pseudonymous signatures.

The main advantage and disadvantage of Restricted Identification is that the scheme creates a volatile proof. The advantage is that as a volatile proof it cannot be misused as it does not create an undeniable digital data. The disadvantage is that one the password created by Restricted Identification has been learned by a malicious party (e.g. via leaking from the target system), it can be used to authenticate as the legitimate user. Moreover, it cannot authenticate any digital data contributed by the user.

Domain specific pseudonymous signature is a concept that:

- a user hold exactly one signing key (or set of keys of a fixed size),
- for any given domain, the user may create its pseudonym using their signing key; the pseudonyms acts simultaneously as a public of this person,
- given domain D and digital data M, a user can create its signature for M; the signature points to D as well as the pseudonym of the signer in domain D,
- the verification process requires the name of domain, signer's pseudonym in this domain, the signature and the signed data. The verification result is negative if, among others, there is mismatch in the domain name or the pseudonym.

While this concept has appeared in the technical recommendation [7], it leaves some details unspecified or as options. For instance, it need to be specified what domain data is required to compute a domain specific pseudonym. In the simplest case (ID based solution) only a domain name is required. The other case is that the domain itself has a public key that has to be generated by a multiparty-protocol in cooperation with an appropriate authority. Both options (together with a technical solution) have been briefly

mentioned in [7]; the second one enables deanonymization of the signer. For more about the formal concept of domain signatures see [11].

Applications. There is a large number of potential applications for domain specific pseudonymous signatures. For typical cases and a classification of such schemes see [3], but let us name just a few ones:

anonymous authentication: domain specific pseudonymous signature might be a part of a challenge-response authentication mechanism, where the user authenticates themselves under domain specific pseudonym. This option has been used for Chip Authentication v.3 from [7] – this scheme is an improvement over the former method based on Chip Authentication with Restricted Identification.

authenticating database entries: if a database consists from records that need to be strongly authenticated, then a good way is to attach a pseudonymous signature of the author for each record. The domain in this case is a database.

self-organization in ad hoc groups: suppose that a self-organization algorithm has to be executed in a group of users that has emerged ad hoc. If the domain is an ad hoc one, related to the group and the situation in which the self-organization algorithm is run, then we may use the domain specific pseudonyms of the users as a source of randomness for the algorithm, thereby derandomizing it in a verifiable way. The signatures can be used to authenticate the data presented by the participants as well as the pseudonyms. For further details see [8].

Previous schemes.

While Pseudonymous Signature presented in [1, 2, 7] is the initial scheme developed in Germany, a few other schemes have followed it, but all of them have been based on pairings friendly groups. The first scheme is due to a French team ([4, 5], see also [10]). Then a few improved schemes for different scenarios have been developed by Kluczniak [9]. A version based on self-created certificates has been presented by M.Wszoła and Marcin Słowik during CECC 2017 (see [3] for a concise description of the scheme). A scheme dealing with "two-dimensional" domains and deanonymizations along dimensions has been presented in [8].

As far as Pseudonymous Signature from [1] are concerned, the only following security report has been published in [12]. Later in this paper we recall its conclusions and countermeasures proposed.

1.1 Paper contribution

The goal of this paper is to improve the Pseudonymous Signature by eliminating its weaknesses – both known before as well as a few ones presented in this paper. We work under the assumption that all countermeasures need to work for standard groups, without retreating to pairings friendly groups. This is motivated by the fact that there are some concerns (economic and technical ones) against wide scale transition to pairing friendly groups.

The goals achieved are the following ones:

- the private keys of the user are known only for the user, the Issuer has no longer access to them (we follow the approach from [12]),
- the Issuer has no longer possibility to create trapdoor information for tracing links between pseudonyms in different domains (we defer the attacks presented in [12]),
- despite the ability of the user to create own private keys, the Issuer together with a Control Authority has possibility to deanonymize the domain specific pseudonym of the user, deanonymization by the Issuer alone is infeasible,
- there is a limited seclusiveness property: if a given user has not been enrolled to the system by the Issuer, then it can be proved during a deanonymization procedure, (However, detection is not automatic during a standard signature verification.)
- we propose a framework of certificates-of-health as an effective solution for Pseudonymous Signature revocation (as well as an ultimate solution for the seclusiveness problem) – lack of a pragmatic revocation procedure have been limiting the scope of applications of Pseudonymous Signature to low risk areas, even in case of schemes based on pairing friendly groups where seclusiveness problem has been solved, (according to [1] the Issuer together with each domain authority creates a blacklist of revoked user – which is not practical if the number of domains is large),
- we present an proof for unlinkability, thereby filling the crucial gap due to problems regarding the argument from [1],
- we extend the scheme so that if only part of the pseudonym is given, then the signatures of a user in a domain remain linkable, but deanonymization is infeasible.

2 Pseudonymous Signature

2.1 Pseudonymous Signature from BSI Technical Recommendation [7]

Let us recall the Pseudonymous Signature Protocol described in the BSI Technical Recommendation [7]. Note that an almost the same protocol has been published by the same authors in the paper [1]; we shall discuss the differences shortly and the end of this section.

Setup. The scheme uses a group $\mathcal{G}$ of a prime order q. The DDH Problem for $\mathcal{G}$ must be hard. Moreover, specification of $\mathcal{G}$ must point to a generator $g \in \mathcal{G}$.
During the setup process the Issuer creates its system keys:

- secret keys $SK_{ICC}, SK_M \in \mathbb{Z}_q$,
- public keys $PK_{ICC}, PK_M \in \mathcal{G}$, where $PK_{ICC} = g^{SK_{ICC}}$ and $PK_M = g^{SK_M}$.

One of peculiar properties of the scheme described in [7] is that the keys PK_M, SK_M are long term keys, while PK_{ICC}, SK_{ICC} change periodically.

User's keys. The user gets a signing device with the following keys installed there by the Issuer:

- private key (x_0, x_1) such that $SK_{ICC} = x_0 + x_1 \cdot SK_M \bmod q$,
- the corresponding public keys $P_0 = g^{x_0}$ and $P_1 = g^{x_1}$.

Setup of a domain. For a domain D, a public key PK_D is defined. There are two ways to do it: according to the first option the discrete logarithm of PK_D is not known to any party. A way to realize it is to compute PK_D by hashing the domain's official name (the hashing function has to map into $\mathcal{G}$). According to the second option, $PK_D = g^d$, where d is a secret key. The key d can be divided into parts: $d = r \cdot R$, where r is known only to the Issuer and R is known only to the domain's authority.

Domain pseudonyms of a user. In a domain D with the public key PK_D, a user holding the keys x_0, x_1 has identifiers I_0^D, I_1^D, where

$$I_0 := (PK_D)^{x_0} \quad \text{and} \quad I_1 := (PK_D)^{x_1} \ .$$

According to the specification, it is possible to use the scheme with just one of the pseudonym components or even to create a signature without indicating the pseudonym.

Creating a signature. A device holding the keys x_0, x_1 creates a signature for m and domain with the public key PK_D in the following way:

- choose k_0, $k_1 < q$ at random,
- compute $Q := g^{k_0} \cdot PK_M^{k_1}$,
- compute the pseudonyms $I_0 := (PK_D)^{x_0}$, $I_1 := (PK_D)^{x_1}$,
- [optional?] compute the commitments $A_0 := (PK_D)^{k_0}$, $A_1 := (PK_D)^{k_1}$,
- compute

$$c := \mathsf{Hash}(Q, I_0, A_0, I_1, A_1, PK_D, m) \ , \tag{1}$$

- compute $s_0 := k_0 - c \cdot x_0 \bmod q, \quad s_1 := k_1 - c \cdot x_1 \bmod q$,
- output the signature (c, s_0, s_1) together with the pseudonyms I_0, I_1.

Verification of a signature. Given the signature (c, s_0, s_1) and pseudonyms I_0, I_1, the following steps are executed by a verifier:

- reconstruct Q as $Q' := (PK_{ICC})^c \cdot g^{s_0} \cdot (PK_M)^{s_1}$,
- reconstruct A_0 as $A_0' := (PK_D)^{s_0} \cdot (I_0)^c$,
- reconstruct A_1 as $A_1' := (PK_D)^{s_1} \cdot (I_1)^c$,
- accept the signature if and only if

$$c = \mathsf{Hash}(Q', I_0, A_0', I_1, A_1', PK_D, m)$$

Note: There are slight differences between the version of the algorithm presented in [1] and in [7]. In [1] only I_0 and A_0 are used and I_1, A_1 are skipped. The differences have been neither explained nor justified.

3 Problems

3.1 Securing against the Issuer

- According to [7], the Issuer creates the private keys for each user and during this process has full access to them. As the Issuer may retain these keys (or modify the generation process so that the domain key of a user can be recovered later), there is a silent assumption that the Issuer is unconditionally honest.
- The above problem has been solved in [12]. According to the modified scheme a user personalizes the private keys, so that they is protected against Issuer aiming to forge signatures of the user. However, this is a double edged sword: the technique applied in [12] prevents (controlled) deanonymization of the user. On the other hand, such a special deanonymization procedure is necessary for most practical applications.

The goal of this paper is to enable personalization such that deanonymization remains possible on a cryptographic way even if the user does not register the personalized public keys.

3.2 External Hash Attack

Below we discuss an attack that may happen, if an implementation of the scheme is not careful enough and does not take care of some details that are not included in the original documentation. Namely, we show that a likely implementation of Pseudonymous Signature can lead to a situation where creating users gets out of control and a PC communicating with 2 different electronic identity documents can create Pseudonymous Signatures attributed to a new identity. Therefore, in such a case seclusiveness property does not really hold, despite that there is no physical attack against the identity documents.

The attack has the following background:

- One of the operations performed during signature creation is computing the value $c = \mathsf{Hash}(Q, I_0, A_0, I_1, A_1, PK_D, m)$ for a message m to be signed. However, m might be a very long document and sending it to the signing device and computing the hash c there would be quite tedious. If for example computing a hash value of a 256-bit block on a smart card takes 5ms, then signing a 4MB document (a photo) would take over 10 minutes due to the hashing effort.
- None of the arguments used for computing c is confidential. What is more, they are explicitly used during a standard signature verification. Therefore, it seems to be harmless to outsource this computation to the computer, from which the signing request originates. Definitely, this would solve the problem of computing a hash value for a long document m.
- One can also argue that even if PC is malicious and sends a manipulated value c, then no secrets are revealed, since in fact in this case the PC executes Schnorr identification protocol with the electronic identity document. Therefore the security arguments about the Schnorr identification protocol apply here as well.

Given protocol specification and the arguments raised above, it seems acceptable to **outsource computing the hash c to the machine interacting with the identity document**. Sometimes, to be on the safe side ("random" properties of the hash output) it is required that the final stage of computing the hash value is performed by the signing device.

Attack description. For the attack described below we assume that the implementation of Pseudonymous Signature outsources computation of the hash c, at least up to the moment when A_1 is fully processed.

Steps of the attack performed by a computer C:

1. C opens communication with two different identity documents, say U and V. Let $x_{0,U}, x_{1,U}$ and $x_{0,V}, x_{1,V}$ denote their private keys.
2. C sends signing requests to U and V.
3. U returns the values Q, I_0, A_0, I_1, A_1, while device V returns $Q', I'_0, A'_0, I'_1, A'_1$.
4. C chooses α at random and computes:
   - $\hat{Q} := Q^{\alpha} \cdot {Q'}^{1-\alpha}$
   - $\hat{A}_0 := A_0^{\alpha} \cdot {A'_0}^{1-\alpha}$
   - $\hat{A}_1 := A_1^{\alpha} \cdot {A'_1}^{1-\alpha}$
   - $\hat{I}_0 := I_0^{\alpha} \cdot {I'_0}^{1-\alpha}$
   - $\hat{I}_1 := I_1^{\alpha} \cdot {I'_1}^{1-\alpha}$
   - $c := \mathsf{Hash}(\hat{Q}, \hat{I}_0, \hat{A}_0, \hat{I}_1, \hat{A}_1, PK_D, m)$

   and returns c both to U and V.
5. U computes the signature values s_0, s_1 and returns them to C, analogously, V computes the signature values s'_0, s'_1 and returns them to C.
6. C computes $\hat{s}_0 := \alpha \cdot s_0 + (1-\alpha) \cdot s'_0$ and $\hat{s}_1 := \alpha \cdot s_1 + (1-\alpha) \cdot s'_1$.
7. Finally, C outputs the signature $(c, \hat{s}_0, \hat{s}_1)$ for an (artificial) identity $\hat{I}_0, \hat{I}_1$.

Fact 1 *$(c, \hat{s}_0, \hat{s}_1)$ is a valid Pseudonymous Signature for pseudonymous identity $\hat{I}_0, \hat{I}_1$ and the message m.*

Proof. The verification procedure yields the positive result if

$$\begin{aligned} \hat{Q} &= (PK_{ICC})^c \cdot g^{\hat{s}_0} \cdot (PK_M)^{\hat{s}_1} \\ \hat{A}_0 &= (PK_D)^{\hat{s}_0} \cdot (\hat{I}_0)^c \\ \hat{A}'_1 &= (PK_D)^{\hat{s}_1} \cdot (\hat{I}_1)^c \end{aligned}$$

We have

$$\begin{aligned} &(PK_{ICC})^c \cdot g^{\hat{s}_0} \cdot (PK_M)^{\hat{s}_1} = (PK_{ICC})^c \cdot g^{\alpha \cdot s_0 + (1-\alpha) \cdot s'_0} \cdot (PK_M)^{\alpha \cdot s_1 + (1-\alpha) \cdot s'_1} \\ &= (PK_{ICC})^c \cdot g^{\alpha \cdot s_0} \cdot g^{(1-\alpha) \cdot s'_0} \cdot (PK_M)^{\alpha \cdot s_1} \cdot (PK_M)^{(1-\alpha) \cdot s'_1} \\ &= \left((PK_{ICC})^{c \cdot \alpha} \cdot g^{\alpha \cdot s_0} \cdot (PK_M)^{\alpha \cdot s_1} \right) \cdot \left((PK_{ICC})^{c \cdot (1-\alpha)} g^{(1-\alpha) \cdot s'_0} \cdot (PK_M)^{(1-\alpha) \cdot s'_1} \right) \\ &= Q^{\alpha} \cdot {Q'}^{1-\alpha} = \hat{Q} \,. \end{aligned}$$

Moreover,

$$\begin{aligned}
&(PK_D)^{\hat{s}_0} \cdot (\hat{I}_0)^c = (PK_D)^{\alpha \cdot s_0 + (1-\alpha)\cdot s'_0} \cdot (I_0^{\alpha} \cdot {I'_0}^{1-\alpha})^c \\
&= (PK_D)^{\alpha \cdot s_0} \cdot (I_0^{\alpha})^c \cdot (PK_D)^{(1-\alpha)\cdot s'_0} \cdot (I'_0)^{1-\alpha})^c \\
&= (PK_D^{s_0} \cdot I_0^c)^{\alpha} \cdot (PK_D)^{s'_0} \cdot (I'_0)^c)^{1-\alpha} = A_0^{\alpha} \cdot (A'_0)^{1-\alpha} = \hat{A}_0 \, .
\end{aligned}$$

Analogously, one can check that $\hat{A}'_1 = (PK_D)^{\hat{s}_1} \cdot (\hat{I}_1)^c$. □

One might hope that the identities created by the attack are in some sense specific and that some test (different from the standard verification procedure) might indicate the attack. Unfortunately, this is not the case:

- The user private keys x_0, x_1 satisfy the linear equation $SK_{ICC} = x_0 + x_1 \cdot SK_M \bmod q$, so all valid keys form a 1-dimensional affine subspace of $\mathbb{Z}_q^2$.
- The keys $\hat{x}_0, \hat{x}_1$ created by the attack form an affine subspace of dimension one. Therefore, each private key pair can be obtained in this way.

Attack countermeasures. A simple change that disables the above mentioned attack is to compute the hash c via the following equation.

$$c = \mathsf{Hash}(Q, I_0, A_0, I_1, A_1, PK_D, \mathsf{Hash}(m))$$

In this scenario the signing device receives the input $\mathsf{Hash}(m)$ and therefore can easily perform the whole computation itself. Another approach could be to take m as the first argument for hashing:

$$c = \mathsf{Hash}(m, Q, I_0, A_0, I_1, A_1, PK_D)$$

In this approach the computation can be outsourced to a PC until the time when the whole m is consumed.

4 Pseudonymous Signature 2.0

In this section we present an enhancement of the scheme presented in [12] (as well as [7]). The current solution eliminates the following problems:

- the Issuer does not know the secret keys of the user (as in [12], but not [7]),
- a user is not forced to register after private key personalization in order to enable deanonymization,
- enables detection of forged identities created after breaking into some number of identity documents,
- each pseudonym can be linked with an entry in a database of legitimate participants, in this process cooperation of the domain authority, the Issuer and the Control Authority is necessary.

Setup. We use a group $\mathcal{G}$ of a prime order q satisfying the same properties as for the original Pseudonymous Signature. Let g be a chosen generator of $\mathcal{G}$.

- The Issuer holds the following system keys used for generating private keys of the users:
  - secret keys $SK_{ICC}, SK_M, SK_L \in \mathbb{Z}_q$ chosen at random,
  - the corresponding public keys PK_{ICC}, PK_M, PK_L, where

$$PK_{ICC} = g^{SK_{ICC}}, \quad PK_M = g^{SK_M}, \quad PK_L = g^{SK_L} .$$

- For deanonymization purposes, the Issuer holds the following keys:
  - private keys δ, γ chosen at random,
  - the corresponding public keys Δ, Γ, where

$$\Delta = g^{\delta} \quad \text{and} \quad \Gamma = g^{\gamma} .$$

- For auxiliary purposes (signing database entries), the Issuer also holds keys for creating standard electronic signatures.

User's pre-keys. The Issuer prepares a signing device with the following keys:

- a private key (x'_0, x'_1, x'_2) such that

$$\begin{cases} SK_{ICC} = x'_0 + x'_1 \cdot SK_M + x'_2 \cdot SK_L \bmod q , \\ \mathsf{ID}_A \quad = x'_1 \cdot \gamma + x'_2 \cdot \delta \bmod q , \end{cases}$$

  where ID_A is an ID token of the user (see below for the ID token specification),
- a twin private key (x''_0, x''_1, x''_2) satisfying the same system of linear equations:

$$\begin{cases} SK_{ICC} = x''_0 + x''_1 \cdot SK_M + x''_2 \cdot SK_L \bmod q , \\ \mathsf{ID}_A \quad = x''_1 \cdot \gamma + x''_2 \cdot \delta \bmod q , \end{cases}$$

- and the corresponding public keys (P'_0, P'_1, P'_2) and (P''_0, P''_1, P''_2), where

$$(P'_0, P'_1, P'_2) = (g^{x'_0}, g^{x'_1}, g^{x'_2}), \quad (P''_0, P''_1, P''_2) = (g^{x''_0}, g^{x''_1}, g^{x''_2}) .$$

For this purpose $x'_1 \in \mathbb{Z}_q$ (respectively, x''_1) is chosen at random, and x'_2, x'_0 (respectively, x''_2, x''_0) are derived based on the linear equations presented above.

The signing device containing $(x'_0, x'_1, x'_2), (x''_0, x''_1, x''_2)$ is given the the user. At the same time the public keys are given directly to the user. A mutual authentication of the user and the Issuer is required at this moment – any effective procedure can be applied.

The public keys given to the user are used only for checking personalization process by the user. This optional procedure aims to to prevent cheating the user (e.g. preinstalling the keys known to the Issuer and running a fake personalization process by the signing device).

Creating ID_A. The following procedure is executed by the Issuer when enrolling a user A. Among others, during the execution an entry concerning A is stored in a database DB_{ID} run by the Issuer.

1. choose $s \in \mathbb{Z}_q$ independently at random, set $\mathsf{ID}_A = s$,
2. store a signed record containing (g^{ID_A}, A) in the database DB_{ID}. (The value ID_A is not published and should be destroyed.)

If the identity of users that have the signing devices is not to be published, then the publicly available part of DB_{ID} may contain the entries $(g^{\mathsf{ID}_A}, \mathrm{Sign}_{\mathrm{Issuer}}(g_A^{\mathsf{ID}}))$.

Note that the construction opens room for a further improvements, where the Issuer chooses numbers s from some subset of all possible exponents, but recognizing whether g^s has been created in this way requires a trapdoor from the Issuer.

Signing key personalization. The user personalizes the pre-keys so that the Issuer loses control over the private keys in a signing device, even if the pre-keys are retained by the Issuer. We follow the approach presented in [12]:

- the user chooses α at random and sets $\beta := 1 - \alpha \bmod q$, the value α is presented to the signing device,
- the signing device sets the signing key to (x_0, x_1, x_2), where

$$(x_0, x_1, x_2) = \alpha \cdot (x_0', x_1', x_2') + \beta \cdot (x_0'', x_1'', x_2'') \bmod q$$

  Note that (x_0, x_1, x_2) satisfies the same system of linear equations:

$$\begin{cases} SK_{ICC} = x_0 + x_1 \cdot SK_M + x_2 \cdot SK_L \bmod q \\ \mathsf{ID}_A \quad = x_1 \cdot \gamma + x_2 \cdot \delta \bmod q \end{cases} \tag{2}$$

- the public keys (P_0, P_1, P_2) corresponding to the personalized private keys are computed according to the formula

$$P_i = (P_i')^{\alpha} \cdot (P_i'')^{\beta} \quad \text{for} \quad i = 0, 1, 2\,. \tag{3}$$

  Note that this computation can be done by the user outside the signing device.

The user may verify that the signing device has installed the private key according to the method described above. The detailed procedure based on the equalities (3) is presented below after describing the pseudonym creation and signing processes.

Domain's parameters. For each domain there are assigned three public keys $\underline{PK}_{\mathrm{dom}}$, PK_{dom}, $\overline{PK}_{\mathrm{dom}}$. The public key $\underline{PK}_{\mathrm{dom}}$ is created by hashing the domain name (so its discrete logarithm is not known with very high probability). The two other public keys fulfil the following equalities:

$$PK_{\mathrm{dom}} = g^{d_1 \cdot d_2}, \qquad \overline{PK}_{\mathrm{dom}} = \Delta^{d_1 \cdot d_2}$$

where the exponent d_1 is chosen at random by the Issuer and the exponent d_2 is chosen at random by the target domain authority.

Creating a signature. Assume that $\underline{PK}_{\text{dom}}$, PK_{dom}, $\overline{PK}_{\text{dom}}$ are the domain parameters of a domain for which a user holding (x_0, x_1, x_2) has to sign m. The signing procedure is analogous to the procedure from [7]:

- choose k_0, k_1, k_2 random,
- compute $Q = g^{k_0} \cdot PK_M^{k_1} \cdot PK_L^{k_2}$,
- compute domain specific pseudonyms of the signer:

$$I_0 = (\underline{PK}_{\text{dom}})^{x_0}, \quad I_1 = (PK_{\text{dom}})^{x_1}, \quad I_2 = (\overline{PK}_{\text{dom}})^{x_2} ,$$

- [optional[1]] compute commitments to random elements k_0, k_1, k_2:

$$A_0 = (\underline{PK}_{\text{dom}})^{k_0}, A_1 = (PK_{\text{dom}})^{k_1}, A_2 = (\overline{PK}_{\text{dom}})^{k_2} ,$$

- for a message m to be signed compute

$$c = \mathsf{Hash}(Q, I_0, A_0, I_1, A_1, PK_{\text{dom}}, I_2, A_2, \mathsf{Hash}(m)) ,$$

- compute the second components of Schnorr signatures:

$$s_0 = k_0 - c \cdot x_0 \bmod q, \quad s_1 = k_1 - c \cdot x_1 \bmod q, \quad s_2 = k_2 - c \cdot x_2 \bmod q ,$$

- output signature (c, s_0, s_1, s_2) together with pseudonyms I_0, I_1, I_2 and the identity of the domain concerned.

Verification of a signature. The verification is a simple extension of the verification procedure of the Pseudonymous Signature [7]:

- recompute Q as

$$Q' = (PK_{ICC})^c \cdot g^{s_0} \cdot (PK_M)^{s_1} \cdot (PK_L)^{s_2}$$

- reconstruct A_0 as $A'_0 = (\underline{PK}_{\text{dom}})^{s_0} \cdot (I_0)^c$,
- reconstruct A_1 as $A'_1 = (PK_{\text{dom}})^{s_1} \cdot (I_1)^c$,
  reconstruct A_2 as $A'_2 = (\overline{PK}_{\text{dom}})^{s_2} \cdot (I_2)^c$,
- accept the signature iff

$$c = \mathsf{Hash}(Q', I_0, A'_0, I_1, A'_1, PK_{\text{dom}}, I_2, A'_2, \mathsf{Hash}(m))$$

[1] creating the pseudonyms and including them in the computation of c is optional according to [7]. If the pseudonyms are not included, then the signature indicates that it has been created by a party holding keys fulfilling the equations (2), so presumably a legitimate user. Note that if only I_0 is given in the signature, then it is possible to link the signatures of the same user in this domain, but it will be impossible to deanonymize him.

Verifying initialization of the keys. Potentially, the signing device might pretend to follow the procedure of device initialization, but instead of initializing the private keys according to the equality (2), it might install some predefined values known to the Issuer, thereby enabling it to create signature on behalf of the user. The following procedure enables to check that the keys are set according to (2):

1. a user chooses r_0, r_1, r_2 at random and ask the device for a signature for a domain with parameters $\underline{PK}_{\text{dom}}$, PK_{dom}, $\overline{PK}_{\text{dom}}$, where
$$\underline{PK}_{\text{dom}} = g^{r_0}, \quad PK_{\text{dom}} = g^{r_1}, \quad \overline{PK}_{\text{dom}} = g^{r_2},$$
2. when a signature with pseudonyms I_0, I_1, I_2 is returned, then the user checks that:
$$I_0 = ((P'_0)^{\alpha} \cdot (P''_0)^{\beta})^{r_0}, \quad I_1 = ((P'_1)^{\alpha} \cdot (P''_1)^{\beta})^{r_1}, \quad I_2 = ((P'_2)^{\alpha} \cdot (P''_2)^{\beta})^{r_2}$$

A user can also check that his ID_A has been correctly marked in the database DB_{ID}. That is, the entry g^{ID_A} should be contained there. For this purpose the following steps are executed:

1. the user chooses r at random and creates artificial domain parameters: $PK_{\text{dom}} = \Delta^r$, $\overline{PK}_{\text{dom}} = \Gamma^r$,
2. the user asks the signing device for a domain signature for such domain parameters,
3. the signing device returns a signature with pseudonyms I_1, I_2,
4. the user computes $S = (I_1 \cdot I_2)^{r^{-1} \bmod q}$ and check whether S is on the publicly available part of DB_{ID}.

Deanonymization. Let us assume that in a domain with parameters $\underline{PK}_{\text{dom}}$, PK_{dom}, $\overline{PK}_{\text{dom}}$ there is a misbehaving user with a pseudonym I_0, I_1, I_2. Then the following steps are executed:

- The domain authority (holding d_2) computes
$$I'_1 := I_1^{d_2^{-1} \bmod q}, \quad I'_2 := I_2^{d_2^{-1} \bmod q}$$
- The Control Authority (holding d_1) computes
$$I''_1 := (I'_1)^{d_1^{-1} \bmod q}, \quad I''_2 := (I'_2)^{d_1^{-1} \bmod q}$$
The resulting values are $I''_1 = g^{x_1}$, $I''_2 = \Delta^{x_2}$.
- Finally, the Issuer computes $I = (I''_1)^{\gamma} \cdot I''_2$, which should be equal to g^{ID_A}.
- the Issuer looks for an entry I in DB_{ID} and finds the identity A of the misbehaving user.

Note that all computations (raising to powers) can be accompanied by some proof of correctness.

Remark 1. One may observe that the Control Authority and the domain authority may derive g^{x_1} (by performing analogous steps for the pseudonym I_0). So linking between domains is possible once the authorities of the domains concerned and the Control Authority collude. However, the real identity of the user will remain hidden.

Fully anonymous version. Let us recall that in our construction we have used the domain parameter $\underline{PK}_{\text{dom}}$, for which the discrete logarithm is not known. Introducing such domain parameter has been possible since PK_{dom} and $\overline{PK}_{\text{dom}}$ enable deanonymization. On the other hand, this opens a option for creating signatures, where only the pseudonym $I_0 = (\underline{PK}_{\text{dom}})^{x_0}$ is presented. Such signatures are bound to the given domain, all signatures of the same user in this domain can be immediately linked, but it is infeasible to recover the identity of the user via deanonymization procedure as long as I_0 is not used together with I_1, I_2 in any signature in this domain.

A similar possibility has been indicated for [7], but then we had two choices: First, the domain parameters could be generated so that the discrete logarithm is not known. In this case every signature in this domain is anonymous. The other case is that no pseudonym is given in the signature (it does not appear as an argument of the hash in (1). In this case the signatures created by the same user in this domain cannot be linked.

5 Comparison

Table 1 presents the proposed algorithm and its basic version from [7] in a tabular form. The new or changed elements in our scheme are presented in the red color.

In Table 2 we compare some features of three versions of Pseudonymous Signature scheme published in [7], [12] and this paper.

BSI Pseudonymous Signature [7]	Pseudonymous Signature 2.0
System setup	
$PK_{ICC} = g^{SK_{ICC}}$ $PK_M = g^{SK_M}$	$PK_{ICC} = g^{SK_{ICC}}$ $PK_M = g^{SK_M}$ $PK_L = g^{SK_L}$ $\Delta = g^{\delta}, \quad \Gamma = g^{\gamma}$
Domain setup	
$PK_{\text{dom}} = g^{d_1 \cdot d_2}$	$\underline{PK}_{\text{dom}}, \quad PK_{\text{dom}} = g^{d_1 \cdot d_2}, \quad \overline{PK}_{\text{dom}} = \Delta^{d_1 \cdot d_2}$
User keys	
x_0, x_1 where: $SK_{ICC} = x_0 + x_1 \cdot SK_M$	x_0, x_1, x_2 where: $SK_{ICC} = x_0 + x_1 \cdot SK_M + x_2 \cdot SK_L$ $\mathsf{ID}_A = x_1 \cdot \gamma + x_2 \cdot \delta$
Signature creation	
choose k_0, k_1 at random $Q = g^{k_0} \cdot PK_M^{k_1}$ $I_0 = (PK_{\text{dom}})^{x_0}$, $I_1 = (PK_{\text{dom}})^{x_1}$ $A_0 = (PK_{\text{dom}})^{k_0}$, $A_1 = (PK_{\text{dom}})^{k_1}$, $c = \mathsf{Hash}(Q, I_0, A_0, I_1, A_1, PK_{\text{dom}}, m)$ $s_0 = k_0 - c \cdot x_0$, $s_1 = k_1 - c \cdot x_1$	choose k_0, k_1, k_2 at random $Q = g^{k_0} \cdot PK_M^{k_1} \cdot PK_L^{k_2}$ $I_0 = (\underline{PK}_{\text{dom}})^{x_0}$, $I_1 = (PK_{\text{dom}})^{x_1}$, $I_2 = (\overline{PK}_{\text{dom}})^{x_2}$ $A_0 = (\underline{PK}_{\text{dom}})^{k_0}$, $A_1 = (PK_{\text{dom}})^{k_1}$, $A_2 = (\overline{PK}_{\text{dom}})^{k_2}$ $c = \mathsf{Hash}(Q, I_0, A_0, I_1, A_1, PK_{\text{dom}}, I_2, A_2, \mathsf{Hash}(m))$ $s_0 = k_0 - c \cdot x_0$, $s_1 = k_1 - c \cdot x_1$, $s_2 = k_2 - c \cdot x_2$
Signature verification	
$Q' = (PK_{ICC})^c \cdot g^{s_0} \cdot (PK_M)^{s_1}$ $A'_0 = (PK_{\text{dom}})^{s_0} \cdot (I_0)^c$ $A'_1 = (PK_{\text{dom}})^{s_1} \cdot (I_1)^c$ $c \stackrel{?}{=} \mathsf{Hash}(Q', I_0, A'_0, I_1, A'_1, PK_{\text{dom}}, m)$	$Q' = (PK_{ICC})^c \cdot g^{s_0} \cdot (PK_M)^{s_1} \cdot (PK_L)^{s_2}$ $A'_0 = (\underline{PK}_{\text{dom}})^{s_0} \cdot (I_0)^c$ $A'_1 = (PK_{\text{dom}})^{s_1} \cdot (I_1)^c$ $A'_2 = (\overline{PK}_{\text{dom}})^{s_2} \cdot (I_2)^c$ $c \stackrel{?}{=} \mathsf{Hash}(Q', I_0, A'_0, I_1, A'_1, PK_{\text{dom}}, I_2, A'_2, \mathsf{Hash}(m))$
Deanonymization / checking ID validity	
$I'_1 := I_1^{d_2^{-1}}$ $I := (I'_1)^{d_1^{-1}}$ check for I in the database of the Issuer	$I'_1 := I_1^{d_2^{-1}}$, $I'_2 := I_2^{d_2^{-1}}$ (domain authority) $I''_1 := (I'_1)^{d_1^{-1}}$, $I''_2 := (I'_2)^{d_1^{-1}}$ (Control Authority) $I := (I''_1)^{\gamma} \cdot (I''_2)$ (Issuer) check for I in the database DB_{ID}

Table 1: Pseudonymous Signature from [7] and our scheme in a tabular form

Properties:	[7]	[12]	this paper
the Issuer can derive user's pseudonyms in any domain	yes	no	no
the Issuer can create valid signatures of the user	yes	no	no
the Issuer can enable tracing a user	yes	no	no
size of the private key	2 exponents	2 exponents	3 exponents
signing device personalization	no	yes	yes
registration necessary to enable deanonymization	no	required	no
deanonymization executed by	Issuer+ domain auth.	Issuer+ domain auth. (if registered)	Issuer+ domain auth. + Control Auth.
recognizing forged ID's (after deanonym.)	by comparison with retained keys	by comparison with registered keys	by checking database
external hash attack	possible	possible	impossible
consequences of breaking into eID	creating fake eID's	creating fake eID's	creating fake eID's
proof of forgery	by Issuer's declaration declaration	by inspecting registration records registration records	by inspecting DB_{ID} created at issuing time (potential extensions)

Table 2: Comparison of properties of Pseudonymous Signature schemes

6 Security

6.1 Seclusiveness

As for the scheme [7] breaking into a small number of signing devices may create problems. Indeed, after breaking into three signing devices the adversary gets three triples (x_0, x_1, x_2) that satisfy

$$SK_{ICC} = x_0 + x_1 \cdot SK_M + x_2 \cdot SK_L$$

Thereby the adversary may derive SK_{ICC}, SK_M, SK_L. The equality

$$\mathsf{ID}_A = x_1 \cdot \gamma + x_2 \cdot \delta$$

is more problematic, since each new broken device brings one new equality but also a new unknown ID_A (note that ID_A is not stored inside the device and not available in DB_{ID}). Therefore it is infeasible to derive γ and δ.

It follows that an adversary may create signing devices that create signatures accepted via the regular verification procedure. However, in order to escape detection of a forgery after executing the deanonymization procedure the adversary would have to use x_1, x_2 such that

$$g^s = \Delta^{x_1} \cdot \Gamma^{x_2} \tag{4}$$

for some entry g^s from the database, where s is secret. For modelling the attack we have to assume that the adversary knows some number of entries v and keys x_1' and x_2' such that $v = \Delta^{x_1'} \cdot \Gamma^{x_2'}$.

First assume that an adversary knows x_1, x_2 such that (4) holds and can find different keys x_1', x_2', which point to the same g^s through running the deanonymization procedure. We shall see that in this case it would be possible to break the discrete logarithm problem. Namely, assume that for an input U we aim to find u such that $U = g^u$. First we choose r_1 and r_2 at random and set $\Delta = g^{r_1}$, $\Gamma = U^{r_2}$. Then we choose at random x_1, x_2 and create an entry (g^s from (4)) equal to $\Delta^{x_1}\Gamma^{x_2}$. We feed these data to the adversary and obtain back different keys x_1', x_2' such that

$$\Delta^{x_1'}\Gamma^{x_2'} = \Delta^{x_1}\Gamma^{x_2} \ .$$

Then

$$\Delta^{(x_1'-x_1)/(x_2-x_2')} = \Gamma, \quad \text{so} \quad g^{r_1\cdot(x_1'-x_1)/(x_2-x_2')} = U^{r_2}$$

and the sought discrete logarithm equals $\frac{r_1\cdot(x_1'-x_1)}{(x_2-x_2')\cdot r_2}$ modulo the group order.

If an adversary can find x_1', x_2' such that $\Delta^{x_1'}\Gamma^{x_2'}$ equals another entry in the database, then we provide a similar argument. We set such an entry as g^{r_0}, and provide the system parameters of the form $\Delta = g^{r_1}$ and $\Gamma = U^{r_2}$, where r_0, r_1, r_2 are chosen at random. Then a solution x_1', x_2' given by the adversary satisfies

$$g^{r_0} = g^{r_1\cdot x_1'} \cdot U^{r_2\cdot x_2'} \ .$$

Consequently, the discrete logarithm of U equals $\frac{r_0 - r_1\cdot x_1'}{r_2\cdot x_2'}$.

6.2 Malicious Issuer

We concern here the problem of effectiveness of personalization of the signing keys. The Issuer knows the pre-keys, so the question is whether he can link the set of keys before personalization with the keys after personalization. The second question is whether the Issuer can impersonate the user once his public keys after personalization are given (in principle, this may not require to learn the identity of the user).

Theorem 2. *If the Issuer can link the private keys before personalization to the public keys after personalization, then the Issuer can solve the DDH problem as well.*

Proof. Assume that the Issuer learns $I_0 = \underline{PK}_{\text{dom}}^{x_0}$, $I_1 = PK_{\text{dom}}^{x_1}$, $I_2 = \overline{PK}_{\text{dom}}^{x_2}$ and has to decide whether they correspond to $x'_0, x'_1, x'_2, x''_0, x''_1, x''_2$. It corresponds to the question whether there is α such that

$$x_i = \alpha x'_i + (1-\alpha)x''_i \quad \text{for } i = 0, 1, 2.$$

These equations are equivalent to

$$(x_i - x'_i) = \alpha(x'_i - x'_i) + (1-\alpha)(x''_i - x'_i) \quad \text{for } i = 0, 1, 2.$$

which in turn reduces to the question whether there is β such that

$$(x_i - x'_i) = \beta(x''_i - x'_i) \quad \text{for } i = 0, 1, 2.$$

This corresponds to existence of β fulfilling the equalities

$$I_0/\underline{PK}_{\text{dom}}^{x'_0} = \underline{PK}_{\text{dom}}^{(x''_0 - x'_0)\beta}, \;\; I_1/PK_{\text{dom}}^{x'_1} = PK_{\text{dom}}^{(x''_1 - x'_1)\beta}, \;\; I_2/\overline{PK}_{\text{dom}}^{x'_2} = \overline{PK}_{\text{dom}}^{(x''_2 - x'_2)\beta} \quad (5)$$

Let $\mathcal{A}$ be an algorithm that enables to check if there is such β. We use it to solve the Problem of Equality of 3 Discrete Logarithms. That is, given an instance (A, B, C, D, E, F) we have to decide whether there is ζ such that $A^\zeta = B$, $C^\zeta = D$, and $E^\zeta = F$.

Note that the DDH problem is the problem of equality of 2 logarithms: given an instance (A, B, C, D) we have to decide whether there is ζ such that $A^\zeta = B$, $C^\zeta = D$. Obviously, solving DDH implies solving Problem of Equality of 3 Discrete Logarithms. Also solving the later problem implies solving problem of equality of 2 logarithms: given (A, B, C, D) we choose r, r' at random and solve the question for $(A, B, C, D, A^r C^{r'}, B^r D^{r'})$ (one can see that if for (A, B, C, D) the answer is negative, then $A^r C^{r'}, B^r D^{r'}$ is uniformly distributed).

Now we prepare an input for algorithm $\mathcal{A}$:

- choose x'_i, x''_i at random for $i = 0, 1, 2$,
- set $\underline{PK}_{\text{dom}} = A^{(x''_0 - x'_0)^{-1}}$, $PK_{\text{dom}} = C^{(x''_1 - x'_1)^{-1}}$, $\overline{PK}_{\text{dom}} = E^{(x''_2 - x'_2)^{-1}}$
- set $I_0 = B \cdot \underline{PK}_{\text{dom}}^{x'_0}$, $I_1 = D \cdot PK_{\text{dom}}^{x'_1}$, $I_2 = F \cdot \overline{PK}_{\text{dom}}^{x'_2}$,

Then for the tuple (A, B, C, D, E, F) the answer is positive, if and only if there is β, for which (5) holds. □

Remark 2. Theorem 2 can be extended to the version where the Issuer is given signatures of the user over messages of his choice, under the assumption of the Random Oracle Model.

Corollary 1. *If the Issuer can forge signatures of a user, given his pre-keys and the public keys after personalization, then he will be able forge the signatures of users for which he does not know the pre-keys.*

Proof. Any difference in forging capabilities would contradict Theorem 2. □

The above corollary says that creating the pre-keys does not endanger the user. Note that this holds despite that the user cannot personalize his keys to get an arbitrary valid triple of keys satisfying (2). Note that the situation is less obvious than in case of the scheme from [12], where any pair of keys could be derived during the personalization process.

6.3 Unlinkability

As pointed out in [6, 10] unlinkability proof from [1] is faulty, so at the moment we are missing even the proof for the standard [7]. The situation in our case is even more complex due to more relationships between the secret key components.

We express unlinkability property in the same way as in [9]: given the pseudonyms and signatures of a user in all domains except for a domain with parameters $\underline{PK}_{\text{dom}}$, PK_{dom}, $\overline{PK}_{\text{dom}}$, we wish to determine whether given pseudonyms I_0, I_1, I_2 in this domain correspond to the key (x_0, x_1, x_2):

$$I_0 = \underline{PK}_{\text{dom}}^{x_0}, \quad I_1 = PK_{\text{dom}}^{x_1}, \quad I_2 = \overline{PK}_{\text{dom}}^{x_2}$$

or have been created as

$$I_0 = \underline{PK}_{\text{dom}}^{x'_0}, \quad I_1 = PK_{\text{dom}}^{x'_1}, \quad I_2 = \overline{PK}_{\text{dom}}^{x'_2}$$

where (x'_0, x'_1, x'_2) have been chosen at random from the set of valid keys. "Unlinkability" means that it is infeasible to distinguish between these two cases given that the challenger chooses the option to present with probability $\frac{1}{2}$ and that the discrete logarithms of $\underline{PK}_{\text{dom}}$, PK_{dom}, $\overline{PK}_{\text{dom}}$ are not known. However, following the construction we know that two discrete logarithms are equal: $PK_{\text{dom}} = g^d$, $\overline{PK}_{\text{dom}} = \Delta^d$ for some (unknown) d.

Our strategy is to show that the case of unlinkability for the scheme [1] (called below the *basic scheme*) would be solved if unlinkability would be violated for our scheme. For this purpose we build a reduction where a case from the basic scheme is converted to a case for an adversary breaking unlinkability of our scheme. So we deal with a case when a user holds private keys x_0 and x_1. The corresponding pseudonyms are generated with these keys.

For our scheme we need three keys as well as new system keys SK_L and δ, γ. First we take ω at random and set:

$$PK_L = PK_M^{\omega}, \quad \text{that is,} \quad SK_L = SK_M \cdot \omega$$

The key δ, γ are chosen at random.

The next step is to convert the x_0 and x_1 from the basic scheme to keys x'_0, x'_1, x'_2. We put $x'_0 = x_0$, x'_1 is chosen at random, while x'_2 is chosen so that

$$x_1 = x'_1 + x'_2 \cdot \omega \ .$$

Of course, we cannot compute these keys explicitly. However note that

$$SK_{ICC} = x_0 + SK_M \cdot x_1 = x'_0 + SK_M \cdot (x'_1 + x'_2 \cdot \omega) = x'_0 + SK_M \cdot x'_1 + SK_L \cdot x'_2$$

and so the new keys fulfill the property required by our scheme.

Having the new keys one has to show how to create pseudonyms in our new case when having the pseudonyms in the corresponding domains of the basic scheme. First, as $\underline{PK}_{\text{dom}}$ is a result of a pseudorandom process, we can program the oracle to get $\underline{PK}_{\text{dom}} = (PK_{\text{dom}})^r$, where r is chosen independently at random for each domain. Then the pseudonym corresponding to $\underline{PK}_{\text{dom}}$ equals $I'_0 = (I_0)^r$. In the next step we set $PK'_{\text{dom}} = PK_{\text{dom}}$. Then the new pseudonym I'_1 is simply $(PK_{\text{dom}})^{x'_1}$ (recall that we know x'_1). The third domain parameter is $\overline{PK}'_{\text{dom}} = (PK'_{\text{dom}})^{\delta}$. So we have to derive the pseudonym $I'_2 = (\overline{PK}'_{\text{dom}})^{x'_2}$ – while of course we do not know x'_2. Note that

$$(x_1 - x'_1)/\omega = x'_2 \ .$$

Therefore it suffices to compute

$$(PK_{\text{dom}})^{\delta})^{(x_1 - x'_1)/\omega}, \quad \text{or equivalently} \quad ((PK_{\text{dom}})^{x_1})/(PK_{\text{dom}})^{x'_1}))^{\delta/\omega} \ .$$

As $(PK_{\text{dom}})^{x_1}$ is known (as the pseudonym in the basic scheme), we can compute the right-hand side expression.

The next problem is to convert the signatures created in the basic scheme into signatures created in our scheme. Let (c, s_0, s_1) be such a signature. Then in particular $s_1 = k_1 - c \cdot x_1$. As we shall program the hash oracle, let us note that finally we shall get the a signature of the form (c, s_0, s'_1, s'_2). First we choose k'_1 at random and set $s'_1 = k'_1 - c \cdot x'_1$. For the component s'_2 note that

$$s_1 = k_1 - c \cdot (x'_1 + x'_2 \cdot \omega)$$

As $s'_1 = k'_1 - c \cdot x'_1$, we can rewrite the last equation:

$$s_1 - s'_1 = k_1 - k'_1 + c \cdot x'_1 - c \cdot (x'_1 + x'_2 \cdot \omega) = (k_1 - k'_1) + c \cdot x'_2 \cdot \omega$$

So finally

$$(s_1 - s'_1)/\omega = (k_1 - k'_1)/\omega + c \cdot x'_2$$

and we set $s'_2 = (s_1 - s'_1)/\omega$. For the hash oracle we set

$$c = \mathsf{Hash}(Q', I'_0, A'_0, I'_1, A'_1, PK_{\text{dom}}, I'_2, A'_2, m)$$

where

$$\begin{aligned} Q' &= (PK_{ICC})^c \cdot g^{s_0} \cdot (PK_M)^{s'_1} \cdot (PK_L)^{s'_2} \\ A'_0 &= A_0 \\ A'_1 &= (PK_{\text{dom}})^{k'_1} \\ A'_2 &= (\overline{PK}'_{\text{dom}})^{s'_2} \cdot (I'_2)^c \end{aligned}$$

(Note that we could completely replace the computation of A'_0, A'_1, A'_2 by the one used by verification. This is possible as we program the hash oracle).

It is easy to see that the transformation of pseudonyms results in pseudonyms corresponding to x'_0, x'_1, x'_2 if in the basic scheme we had the pseudonyms corresponding to x_0, x_1. If this is not the case, and, say, the pseudonyms correspond to some different y_0, y_1. However, in this case $y_0 \neq x_0$, as all valid keys have to satisfy the same linear equation. Consequently, $I'_0 = I_0^r = g^{y_0 r} \neq g^{x_0 r}$. One can see also that I'_2 would also be different, then derived from x_1.

Remark 3. The same technique as above can be used to reduce the question of unlinkability for the scheme [1] to an artificial scheme, where a user holds only one secret key and the signature scheme reduces to Schnorr signatures where the base element is PK_{dom}. (Of course, such a scheme is useless as it is impossible to show seclusiveness property.) On the other hand, for the artificial scheme unlinkability property can be reduced to the DDH problem, as long as we can program the hash oracle. Consequently we may claim that the proposed schemes, including the standard [7], indeed satisfies unlinkability property.

6.4 Unforgeability

The argument for unforgeability of the scheme follows the same steps as for [1] and is based on the Forking Lemma.

7 Seclusiveness and revocation with certificates-of-health

From the practical point of view, the main problem related to domain specific signatures might be checking the status of the signature keys. Indeed, if the signature scheme has to be used to any serious application, then we have to take into account that the signing devices are sometime lost or stolen, the keys might be compromised, etc. This fundamental problem has not been addressed so far in the literature. In particular, the schemes based on pairing friendly groups solve the seclusiveness problem but provide no answer how to revoke the signing keys.

For traditional signature schemes the problem might be solved by PKI mechanisms. Each user gets a certificate for their public key issued by a Certification Authority. Moreover, there are either blacklists or whitelists that are kept up to date regarding revocation cases. For domain specific pseudonyms and signatures this approach is doomed to fail completely. First, while the number of potential domains is potentially unlimited, the number of the domains where a user will eventually be active is small. As it is not

known in which domains a user might be active, the white or blacklist should be updated for each single domain when a status change occurs. The fundamental problem is that even if we learn that a pseudonym I in domain D has to be included in the white- or blacklist, we cannot say which pseudonym in a domain D', for $D' \neq D$, should be revoked as well. So, alone computing all pseudonyms corresponding to a compromised key is a practical challenge and must require additional secret keys, presumably held by the Issuer. So potentially for each change of the status of a key the Issuer would be forced to create as many updates to the lists as the number of existing domains. This is hard to imagine to work in practice, especially in case of trans-border services.

A domain signature scheme by Słowik and Wszoła enable to issue a single certificate per user (see [3]). Such a certificate can be randomized and recomputed by the user to create a domain specific certificate authenticating the pseudonym of the user in a given domain. The derived certificates are unlinkable between themselves and in particular do not link to the "base" certificate. This solutions looks very attractive, however leaves revocation problem unsolved.

In this situation we believe that using *certificates-of health* might be the best solution. This can be supported by the following arguments:

- in most cases the signatures and pseudonyms will be used online, we may assume that at least the user will periodically go online and at his moment be able to fetch new certificates,
- in some application cases the risk is relatively low and we do not need to check the status of the signing key, so the number of certificates is not as high as the number of domains, where the user is active.

However, perhaps the most important argument is that (as a side effect) the certificates-of-health solve also the seclusiveness problem: no fake user can get a certificate-of-health.

Below we present a draft of the proposed solution. The construction is quite generic and we talk about a single private key x instead of two components x_1, x_2 that are used in case of Pseudonymous Signature. In the later case we can fetch and use a certificate for one component of the key or extend the procedures in a straightforward way.

FETCHING A CERTIFICATE

Assume that Alice is a system participant and holds a device A implementing her a private key x. Assume that the Issuer holds the public key g^x of Alice. The following steps are executed to get a certificate-of-health:

1. A chooses k at random and computes $h = g^{x \cdot k}$,
2. a secure connection between A and the Issuer is established, A authenticates itself against the Issuer with the key x, for instance by executing a chip authentication protocol based on Pseudonymous Signature and where g is the domain public key,
3. the Issuer checks the status of the public key g^x and rejects the request, if it is on a black list,
4. A requests a blind signature over h and the current time t,
5. th Issuer checks that A knows the discrete logarithm of h; for instance, the Issuer decides at random whether to proceed with signing blindly h, otherwise the Issuer

asks A to open the commitment for h and disclose k such that $h = (g^x)^k$, after the test the procedure is restarted with a new h,
6. A recovers a signature s of h, t created blindly by the Issuer, (h, t, s) is the certificate-of health.

In case of any irregularity during opening the commitment for h the device A is permanently blacklisted as a malicious one and consequently excluded from the system.

AUTHENTICATING A PSEUDONYM
Assume that Alice aims to confirm validity of her pseudonym $I = D^x$ for a domain with domain public key D. The following steps are executed:

1. A sends to the domain the following data:
the certificate (h, t, s), the pseudonym I, and the element $I' = I^k$,
2. A proves that it knows $\log_D I$, $\log_I I'$, and that $\log_g h = \log_D I'$.
3. The domain accepts pseudonym I if the proof is valid and the time t is fresh enough to be accepted according to the security policy of the domain.

If the second step succeeds, then one can conclude that A knows $\log_g h$. On the other hand, the certificate-of-health has been given to a device that has presented a valid ID and has known $\log_g h$, where the logarithm has been generated at random. So one can assume that that the device presenting the certificate and the pseudonym I is the same person that has obtained the certificate-of health

For the proof of knowledge of discrete logarithms any standard protocol can be applied.

References

1. Bender, J., Dagdelen, Ö., Fischlin, M., Kügler, D.: Domain-specific pseudonymous signatures for the German identity card. In Gollmann, D., Freiling, F.C., eds.: Information Security, ISC 2012,. Volume 7483 of LNCS., Springer (2012) 104–119
2. Bender, J., Dagdelen, Ö., Fischlin, M., Kügler, D.: Domain-specific pseudonymous signatures for the German identity card. IACR Cryptology ePrint Archive **2012** (2012) 558
3. Błaśkiewicz, P., Hanzlik, L., Kluczniak, K., Krzywiecki, Ł., Kutyłowski, M., Słowik, M., Wszoła, M.: Pseudonymous signature schemes. In Li, K.C., Chen, X., Susilo, W., eds.: Advances in Cyber Security: Principles, Techniques, and Applications. Springer Nature, Singapore (2018)
4. Bringer, J., Chabanne, H., Lescuyer, R., Patey, A.: Efficient and strongly secure dynamic domain-specific pseudonymous signatures for ID documents. IACR Cryptology ePrint Archive **2014** (2014) 67
5. Bringer, J., Chabanne, H., Lescuyer, R., Patey, A.: Efficient and strongly secure dynamic domain-specific pseudonymous signatures for ID documents. In Christin, N., Safavi-Naini, R., eds.: Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers. Volume 8437 of Lecture Notes in Computer Science., Springer (2014) 255–272
6. Bringer, J., Chabanne, H., Lescuyer, R., Patey, A.: Efficient and strongly secure dynamic domain-specific pseudonymous signatures for ID documents. In Christin, N., Safavi-Naini, R., eds.: Financial Cryptography and Data Security. Volume 8437 of LNCS. Springer Berlin Heidelberg (2014) 255–272

7. BSI: Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token 2.20. Technical Guideline TR-03110-2 (2015)
8. Hanzlik, L., Kluczniak, K., Kutyłowski, M., Dolev, S.: Local self-organization with strong privacy protection. In: Trustcom/BigDataSE/ISPA, 2016 IEEE, IEEE (2016) 775–782
9. Kluczniak, K.: Anonymous authentication using electronic identity documents. Institute of Computer Science, Polish Academy of Sciences, PhD Thesis (2015)
10. Kluczniak, K.: Domain-specific pseudonymous signatures revisited. IACR Cryptology ePrint Archive **2016** (2016) 70
11. Kluczniak, K., Hanzlik, L., Kutylowski, M.: A formal concept of domain pseudonymous signatures. In Bao, F., Chen, L., Deng, R.H., Wang, G., eds.: Information Security Practice and Experience - 12th International Conference, ISPEC 2016, Zhangjiajie, China, November 16-18, 2016, Proceedings. Volume 10060 of Lecture Notes in Computer Science. (2016) 238–254
12. Kutyłowski, M., Hanzlik, L., Kluczniak, K.: Pseudonymous signature on eIDAS token - implementation based privacy threats. In Liu, J.K., Steinfeld, R., eds.: Information Security and Privacy - 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part II. Volume 9723 of Lecture Notes in Computer Science., Springer (2016) 467–477
13. The European Parliament and the Council of the European Union: Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/ec (General Data Protection Regulation). Official Journal of the European Union **119**(1) (2016)