Multi-device Anonymous Authentication

Kamil Kluczniak[1(✉)], Jianfeng Wang[2], Xiaofeng Chen[2], and Mirosław Kutyłowski[1]

[1] Department of Computer Science, Wrocław University of Science and Technology, Wrocław, Poland
{kamil.kluczniak,miroslaw.kutylowski}@pwr.edu.pl
[2] State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an, China
wjf01@163.com, xfchen@xidian.edu.cn

Abstract. Recently, a few pragmatic and privacy protecting systems for authentication in multiple systems have been designed. The most prominent examples are Restricted Identification and Pseudonymous Signature schemes designed by the German Federal Office for Information Security for German personal identity cards. The main properties are that a user can authenticate himself with a single private key (stored on a smartcard), but nevertheless the user's IDs in different systems are unlinkable.

We develop a solution which enables a user to achieve the above mentioned goals while using more than one personal device, each holding a single secret key, but different for each device – as for security reasons no secret key is allowed to leave a secure device. Our solution is privacy preserving: it will remain hidden for the service system which device is used. Nevertheless, if a device gets stolen, lost or compromised, the user can revoke it (leaving his other devices intact).

In particular, in this way we create a strong authentication framework for cloud users, where the cloud does not learn indirectly personal data. In the standard solutions there is no way to avoid leaking information that, for instance, the user is in his office and authenticates via his desktop computer.

Our solution is based on a novel cryptographic primitive, called Pseudonymous Public Key Group Signature.

Keywords: Signature schemes · Privacy · Pseudonyms · Group signature

1 Introduction

So far most authentication systems for web services or cloud servers where designed having in mind a single user or a group of users and a single service

This research was supported by National Research Center grant PRELUDIUM 8 number 02NP/0016/15 (decision number 2014/15/N/ST6/04655) and Polish-Chinese cooperation venture of Xidian University and Wrocław University of Science and Technology on Secure Data Outsourcing in Cloud Computing.

© Springer International Publishing AG 2016
J. Chen et al. (Eds.): NSS 2016, LNCS 9955, pp. 21–36, 2016.
DOI: 10.1007/978-3-319-46298-1_2

provider. Today such systems become increasingly popular and the number of systems used per user is rapidly growing. If authentication is taken seriously (not based just on a login and a password), then for each service we get an independent authentication environment that requires generating and distribution of the secret keys for the users. Such a framework has serious disadvantages: the necessity of managing secret/public keys among certain parties, constant updates of user secret keys and maintaining large and costly PKI infrastructures.

In this paper, we develop a framework which aims to provide a cryptographically sound authentication scheme to a dynamically growing set of services, which preserves privacy for groups of users and does not require expensive, time and resource consuming infrastructures as well as key management procedures.

Application Scenario. In order to be more specific, we consider an application scenario of Multiple Mobile Devices and Authentication for Web Services, called below *domains*: We assume that:

- a user registers to a given domain only once,
- the user may register himself in many different domains, but he should use the same device or set of devices for interaction with these domains,
- a given user is in possession of a few devices that may be used interchangeably (mobiles devices, desktop computers, etc.),
- the user should not be bothered to register these devices in each single domain in order to use them, ...
- ... but must be able to revoke each of the devices in a case of theft, key leakage, etc.

For usability reasons, we assume that a user registers once in a domain by providing his public key for this domain. Moreover, no party except for the user and the service domain should be involved. (We do not consider how the user is initially authenticated – he may appear in person, authenticate himself via a payment, authenticate himself with a personal identity card or by other means.)

After registration, without any updates or interaction with any party, the participant should be able to delegate the right to run authentication protocol on behalf of the user and sign digitally challenges in order to authenticate the user.

Privacy and Unlinkability Issues. One of the major threats in a multi-system environment is that the authentication means from one domain can be misused for getting unlawful access into user's accounts in another domain. For password based systems this is a severe threat as the users tend to use the same password in multiple places. Many recent examples are known where compromise of one system resulted in compromising users' accounts in another systems.

Apart from unlawful access, it might be necessary to protect the information that a given physical person is a user in a domain. Therefore after the phase of registration the user's identity should be anonymized. Moreover, the pseudonyms

in different domains should be unlinkable, even when the data from authentication sessions are at hand. In this case a potential data leakage is not threatening the principles of personal data protection.

Group Signatures. Group signatures as defined in [1] or [2] are signature schemes in which a *group manager* admits the users to the group. Each of the group members may sign data anonymously on behalf of the group. Only an entity called an *opener* may *"open"* a signature and derive the signer's real identity. Informally, a group signature scheme has to fulfil the following properties:

anonymity: it is infeasible to establish the signer of a message. To be more specific, it is infeasible to link the signature to a single user, i.e. having two signatures one cannot even say whether they originate from one signer or from two different signers.

unframeability: it is infeasible, even for a coalition of malicious group members, to forge a signature which would open to the identity of a group member not belonging to the coalition.

traceability: it is infeasible to produce a signature which would open to an identity not added to the group by the group manager.

Group signatures is a well studied cryptographic primitive. There are many variants of them, with security proofs based either on the random oracle model (e.g. [3]), or on the standard model (e.g. [4]). Many variants of group signatures have been developed, like Verifier Local Group Signatures [5], Traceable Signatures [6], Hierarchical [7], Attribute [8] and Identity Based Group Signatures [9].

Ad Hoc Solution Based on Group Signatures. At a first look, group signature schemes address our practical problem pretty well. The user plays the role of the group manager for group signatures, while his devices play the role of group members (admitted by the manager). Note that this constructions gives some functionalities for free:

- the user can delegate his rights to authenticate on behalf of him to any number of his devices – indeed, the number of group members is typically unlimited,
- the devices are indistinguishable from the point of view of the verifier – this is the basic feature of group signatures,
- in case of a misbehavior, the user may open a signature and find which device has created it.

Unfortunately, there are also some drawbacks that have to be addressed. The main problem is that we have to create separate and unlinkable authentication means for different domains. Creating a new independent group for each domain separately would solve this problem, however this would require installing separate keys for each domain on each single device. For practical reasons this is not really acceptable.

Unfortunately, existing group signature schemes have been designed having in mind single groups or a hierarchy of groups with central authorities. In particular, existing schemes assume that a group of such a hierarchy is identified by a public key determined by the scheme setup. This makes such schemes unsuitable for our application. Our aim is therefore to design a group signature scheme in which group public keys may be derived spontaneously from a domain specific bit string (e.g. www.some-service.com), a secret key of the group manager, and with no involvement of PKI infrastructures and/or trusted authorities.

Moreover, group public keys or, as we will call it, *domain pseudonyms* must be unlinkable, what means that having two or more domain pseudonyms from distinct domains it is infeasible to tell whether the pseudonyms correspond to a group manager.

Such an anonymity notion is known from Domain Pseudonymous Signature schemes (see e.g. [10]), (see e.g. Direct Anonymous Attestation [11]) and Anonymous Credential Systems (see e.g. [12]). What is important, creating new public keys by a group manager does not require from group members to update their secret keys or any other information and they might automatically sign data corresponding to the new public key.

Contribution and Paper Overview. Our main technical contribution is a new concept of group signatures, where group public keys are domain pseudonyms which may be derived spontaneously. The particular setting is tailored for the above mention application of delegating authentication chores to multiple devices of a user.

In Sect. 3 We give a formal definition for our new primitive. This is followed in Sect. 4 by a relatively efficient construction based on pairings. We give also some intuition about its security properties and formulate corresponding theorems. The proofs of these theorems are based on the random oracle model assumption, which is dictated mainly by efficiency and practical needs of the construction. In Sect. 4.3 we provide some additional remarks and we show how to apply our scheme to solve our practical problem.

2 Preliminaries

Bilinear Groups. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of a prime order p, generated by $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$. In our scheme we make use of bilinear maps $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, which are:

- *bilinear*: for $a, b \in \mathbb{Z}_p$, we have $e(g_1^a, g_2^b) = e(g_1, g_2)^{a \cdot b}$,
- *non-degenerate*: the element $e(g_1, g_2) \in \mathbb{G}_T$ is a generator of $\mathbb{G}_T$.

Additionally, we require that e and all group operations are efficiently computable.

Throughout the paper we will use Type-3 pairing according to the classification from [13]. We call a pairing of Type-3, if $\mathbb{G}_1 \neq \mathbb{G}_2$ and no efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ is known.

Security Assumptions.

Definition 1 (Discrete Logarithm Problem (DLP)). *Let $\mathbb{G}$ be a cyclic group of prime order p with a generator $g \in \mathbb{G}$. An algorithm A has advantage ϵ in solving the DLP if*

$$\Pr[\mathsf{A}(g, g^{\alpha}) \rightarrow \alpha] \geq \epsilon,$$

where the probability is taken over the random choice of the generator $g \in \mathbb{G}$, the random choice of $\alpha \in \mathbb{Z}_p$, and the random bits of A.

We say that the (t, ϵ)-DL assumption holds in $\mathbb{G}$ if no time t algorithm has advantage ϵ in solving DLP in $\mathbb{G}$.

Definition 2 (Decisional Diffie-Hellman Problem (DDH)). *Let $\mathbb{G}$ be a cyclic group of order p with a generator $g \in \mathbb{G}$. An algorithm A has advantage ϵ in solving the DDH problem if*

$$|\Pr[\mathsf{A}(g^{\alpha}, g^{\beta}, g^{\alpha \cdot \beta}) \rightarrow 1] - \Pr[\mathsf{A}(g^{\alpha}, g^{\beta}, g^{\gamma}) \rightarrow 1]| \geq \epsilon,$$

where the probability is taken over the random choice of $g \in \mathbb{G}$, the random choice of $(\alpha, \beta, \gamma) \in \mathbb{Z}_p^3$, and the random bits of A.

We say that the (t, ϵ)-DDH assumption holds in $\mathbb{G}$, if no time t algorithm has advantage at least ϵ in solving the DDH problem in $\mathbb{G}$.

Definition 3 (Symmetric eXternal Diffie-Hellman assumption (SXDH)). *Let $\mathbb{G}_1$, $\mathbb{G}_2$ be cyclic groups of a prime order and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear map. The SXDH assumption says that the DDH assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.*

Definition 4 (Bilinear Decisional Diffie-Hellman Assumption). *Let $\mathbb{G}$ be a cyclic group of a prime order and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map. An algorithm A as advantage ϵ in solving the BDDH problem if*

$$|\Pr[\mathsf{A}(g^{\alpha}, g^{\beta}, g^{\gamma}, e(g, g)^{\alpha \cdot \beta \cdot \gamma}) \rightarrow 1] - \Pr[\mathsf{A}(g^{\alpha}, g^{\beta}, g^{\gamma}, e(g, g)^{\delta}) \rightarrow 1]| \geq \epsilon,$$

where the probability is taken over the random choice of $g \in \mathbb{G}$, the random choice of $(\alpha, \beta, \gamma, \delta) \in \mathbb{Z}_p^3$, and the random bits of A.

We say that the (t, ϵ)-BDDH assumption holds in $\mathbb{G}$, if no time t algorithm has advantage at least ϵ in solving the BDDH problem in $\mathbb{G}$.

Definition 5 (Collusion attack algorithm with q traitors (q-CAA)). *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be groups of a prime order p and generated by $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear map which maps into a target group $\mathbb{G}_T$.*

An algorithm A has advantage ϵ in solving the q-CAA problem, if

$$\Pr\left[\begin{array}{c} \mathsf{A}(g_1, g_1^z, (m_1, g_1^{\frac{1}{z+m_1}}), \ldots, (m_q, g_1^{\frac{1}{z+m_q}}), \\ g_2, g_2^z) \rightarrow (m, g_1^{\frac{1}{z+m}}) \wedge m \notin \{m_1, \ldots, m_q\} \end{array}\right] \geq \epsilon,$$

where the probability is taken over the random choice of $(g_1, g_2) \in \mathbb{G}_1 \times \mathbb{G}_2$, the random choice of $z \in \mathbb{Z}_p$, the random choice of $(m_1, \ldots, m_q) \in \mathbb{Z}_p^q$, and the random bits of A.

We say that (q, t, ϵ)-CAA assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$, if no time t algorithm has advantage at least ϵ in solving the q-CAA problem in $(\mathbb{G}_1, \mathbb{G}_2)$.

3 Formal Model of Pseudonymous Public Key Group Signature

A Pseudonymous Public Key Group Signature scheme consists of the following procedures:

$\mathsf{Setup}(1^\lambda)$: On input a security parameter λ, it outputs global parameters $param$.
$\mathsf{CreateUser}(param)$: On input the global parameters $param$, it creates and outputs the user's master secret key mSK.
$\mathsf{ComputePseudonym}(param, mSK, \mathtt{dom})$: On input the global parameters $param$, the master secret key mSK and a domain name $\mathtt{dom}$, it returns a pseudonym nym within domain $\mathtt{dom}$ for the user holding mSK.
$\mathsf{AddDevice}(param, mSK, i)$: On input the global parameters $param$, the master secret key mSK and a device identifier i, this procedure returns a device secret key uSK_i.
$\mathsf{CreateRevocationToken}(param, mSK, \mathtt{dom}, i, j)$: On input the global parameters $param$, user index i and his secret key mSK, the domain name $\mathtt{dom}$ and a device identifier j, this procedure computes and outputs a device revocation token $uRT_{i,j,\mathtt{dom}}$ within the domain $\mathtt{dom}$.
$\mathsf{Sign}(param, uSK, \mathtt{dom}, m)$: On input the global parameters $param$, a device secret key uSK, a domain name $\mathtt{dom}$ and a message m, it returns a signature σ on the message m. (Note that we do not require that the pseudonym nym is used.)
$\mathsf{Verify}(param, nym, \mathtt{dom}, \sigma, m, uRT)$: On input the global parameters $param$, a pseudonym nym with regards to a domain name $\mathtt{dom}$, a signature σ on a message m, and a revocation token uRT, this algorithm returns 1 (accept), or 0 (reject).

Below we discuss the required properties of Pseudonymous Public Key Group Signature.

Correctness. A Pseudonymous Public Key Group Signature is correct, if for every $\lambda \in \mathbb{N}$, $param \leftarrow \mathsf{Setup}(1^\lambda)$, domain name $\mathtt{dom} \in \{0,1\}^*$, and message $m \in \{0,1\}^*$, if

$$
\begin{aligned}
&mSK_i \leftarrow \mathsf{CreateUser}(param)\\
&uSK_{i,j} \leftarrow \mathsf{AddDevice}(param, mSK_i, j)\\
&nym \leftarrow \mathsf{ComputePseudonym}(param, mSK_i, \mathtt{dom})\\
&uRT_{i,j,\mathtt{dom}^*} \leftarrow \mathsf{CreateRevocationToken}(param, mSK_i, \mathtt{dom}^*, j)\\
&\sigma \leftarrow \mathsf{Sign}(param, uSK_{i,j}, \mathtt{dom}, m)
\end{aligned}
$$

then

$$
\begin{aligned}
&\mathsf{Verify}(param, nym, \mathtt{dom}, \sigma, m, R) = 1 \quad \text{for} \quad R \neq uRT_{i,j,\mathtt{dom}^*}\\
&\mathsf{Verify}(param, nym, \mathtt{dom}, \sigma, m, uRT_{i,j,\mathtt{dom}^*}) = 0.
\end{aligned}
$$

In order to define the remaining properties we use the following notation: $\mathcal{U}_{SET}$ stands for the list of users and their secret keys, $\mathcal{D}_{SET}$ contains triples

(i, j, uSK), where i denotes a user index, j is a device index and uSK is its secret key, $\mathcal{CD}$ is a list pointing to corrupted devices and $\mathcal{S}$ is a list of signature query records. Then we define the following oracles used by the adversary during the security games:

$\mathcal{O}_{\mathsf{CreateUser}}$: On input i, if there exists an entry $(i, .)$ in $\mathcal{U}_{SET}$, the oracle aborts. Otherwise the oracle runs $mSK_i \leftarrow \mathsf{CreateUser}(param)$ and adds the pair (i, mSK_i) to $\mathcal{U}_{SET}$.

$\mathcal{O}_{\mathsf{GetNym}}$: On input $\mathtt{dom}$ and i, the oracle finds the secret key mSK_i in $\mathcal{U}_{SET}$ corresponding to i. If no such entry exists, then the oracle aborts. Otherwise the oracle computes $nym_{i,\mathtt{dom}} \leftarrow \mathsf{ComputePseudonym}(param,\ mSK_i,\ \mathtt{dom})$ and returns $nym_{i,\mathtt{dom}}$.

$\mathcal{O}_{\mathsf{AddDevice}}$: On input a user index i and a device identifier j, the oracle finds an entry $(i, mSK_i) \in \mathcal{U}_{SET}$ and checks that $(i, j, \cdot) \notin \mathcal{D}_{SET}$. If $(i, j, \cdot) \notin \mathcal{D}_{SET}$, then the oracle aborts. Then $uSK_{i,j} \leftarrow \mathsf{AddDevice}(param, mSK_i, j)$ and the oracle adds the tuple $(i, j, uSK_{i,j})$ to $\mathcal{D}_{SET}$.

$\mathcal{O}_{\mathsf{AddCorruptedDevice}}$: On input a user identifier i and a device identifier j, the oracle finds $(i, mSK_i) \in \mathcal{U}_{SET}$ and checks that $(i, j, \cdot) \notin \mathcal{D}_{SET}$ (if this is not the case, then the oracle aborts). Otherwise the oracle runs $uSK_{i,j} \leftarrow \mathsf{AddDevice}(param,\ mSK,\ j)$, adds the tuple $(i, j, uSK_{i,j})$ to $\mathcal{D}_{SET}$ and $\mathcal{CD}$, and outputs $uSK_{i,j}$.

$\mathcal{O}_{\mathsf{GetRT}}$: On input a user identifier i and his master key mSK_i, a device identifier j and a domain name $\mathtt{dom}$, the oracle checks that $(i, j, \cdot) \in \mathcal{D}_{SET}$, (if this is not the case, then the oracle aborts). Then the oracle computes $uRT_{i,j,\mathtt{dom}} \leftarrow \mathsf{CreateRevocationToken}(param,\ mSK_i,\ \mathtt{dom},\ j)$ and returns $uRT_{i,j,\mathtt{dom}}$.

$\mathcal{O}_{\mathsf{Sign}}$: On input a user identifier i, a device identifier j, a domain name $\mathtt{dom}$ and a message m, the oracle finds the corresponding secret key $uSK_{i,j}$ in $\mathcal{D}_{SET}$, (if such an entry does not exist, then the oracle aborts). Otherwise, the oracle runs $\sigma \leftarrow \mathsf{Sign}(param, uSK_{i,j}, \mathtt{dom}, m)$, adds $(\sigma, m, \mathtt{dom}, j, i)$ to $\mathcal{S}$ and returns σ.

$\mathcal{O}_{\mathsf{CorruptDevice}}$: On input a user identifier i and a device identifier j, the oracle finds the secret key $uSK_{i,j}$ in $\mathcal{D}_{SET}$ corresponding to i and j. (If such an entry does not exist, then the oracle aborts.) Then the oracle returns $uSK_{i,j}$ and adds (i, j) to $\mathcal{CD}$.

Unforgeability. This property says that no coalition of malicious devices of a user can forge a signature on behalf of a device not belonging to the coalition. We define the unforgeability property by the following experiment:

Experiment $\mathsf{UNF}_{\mathsf{A}}^{S}(\lambda)$**:**

- $(param) \leftarrow \mathsf{Setup}(1^{\lambda})$.
- $\mathcal{O} \leftarrow \{\mathcal{O}_{\mathsf{CreateUser}}, \mathcal{O}_{\mathsf{GetNym}}, \mathcal{O}_{\mathsf{AddDevice}}, \mathcal{O}_{\mathsf{GetRT}}, \mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{CorruptUser}}\}$.
- $(\sigma^*, m^*, \mathtt{dom}^*, nym^*) \leftarrow \mathsf{A}^{\mathcal{O}}(param)$.

- If
  - $\mathsf{Verify}(param, nym^*, \mathtt{dom}^*, \sigma^*, m^*, \perp) = 1$ and
  - There exists $(i, mSK_i) \in \mathcal{U}_{SET}$, $(i, j, \cdot) \in \mathcal{D}_{SET}$ such that
    $nym^* = \mathsf{ComputePseudonym}(param, mSK_i, \mathtt{dom}^*)$,
    $uRT_{i,j,\mathtt{dom}^*} \leftarrow \mathsf{CreateRevocationToken}(param, mSK_i, \mathtt{dom}^*, j)$
    $\mathsf{Verify}(param, nym^*, \mathtt{dom}^*, \sigma^*, m^*, uRT_{i,j,\mathtt{dom}^*}) = 0$
    $(i, j) \notin \mathcal{CD}$ and $(\sigma^*, m^*, \mathtt{dom}^*, j, i) \notin \mathcal{S}$,

  then the challenger returns 1.
- Otherwise the challenger returns 0.

Definition 6. *A Pseudonymous Public Key Group Signature S is (t, ϵ)-unforgeable if $\Pr[UNF^S_{\mathsf{A}}(\lambda) = 1] \leq \epsilon$ for any adversary A running in time t.*

Seclusiveness. Seclusiveness means that it is infeasible to produce a signature on behalf of the user and that does not correspond to any device of the user. In other words, it is infeasible to create a signature that corresponds to none of the revocation tokens. Seclusiveness is formally defined by the following experiment.

Experiment $\mathsf{SEC}^S_{\mathsf{A}}(\lambda)$:

- $(param) \leftarrow \mathsf{Setup}(1^\lambda)$.
- $\mathcal{O} \leftarrow \{\mathcal{O}_{\mathsf{CreateUser}}, \mathcal{O}_{\mathsf{GetNym}}, \mathcal{O}_{\mathsf{AddCorruptedDevice}}, \mathcal{O}_{\mathsf{GetRT}}\}$.
- $(\sigma^*, m^*, \mathtt{dom}^*, nym^*) \leftarrow \mathsf{A}^{\mathcal{O}}(param)$.
- If
  - $\mathsf{Verify}(param, nym^*, \mathtt{dom}^*, \sigma^*, m^*, \perp) = 1$ and
  - there exists $(i, mSK_i) \in \mathcal{U}_{SET}$ such that
    $nym^* = \mathsf{ComputePseudonym}(param, mSK_i, \mathtt{dom}^*)$
    and for all j such that $(i, j, \cdot) \in \mathcal{D}_{SET}$:
    $uRT_{i,j,\mathtt{dom}^*} \leftarrow \mathsf{CreateRevocationToken}(param, mSK_i, \mathtt{dom}^*, j)$
    $\mathsf{Verify}(param, nym^*, \mathtt{dom}^*, \sigma^*, m^*, uRT_{i,j,\mathtt{dom}^*}) = 1$

  the challenger returns 1.
- Otherwise the challenger returns 0.

Definition 7. *We say that a Pseudonymous Public Key Group Signature S is (t, ϵ)-seclusive, if $\Pr[\mathsf{SEC}^S_{\mathsf{A}}(\lambda) = 1] \leq \epsilon$ for any adversary A running in time t.*

Anonymity. We require that it is infeasible to correlate two signatures of the same device (unless its revocation token is used). For the anonymity experiment we define an additional oracle:

$\mathcal{O}_{\mathsf{Challenge}}$: This oracle takes as input a bit b, a user index i^*, a domain name $\mathtt{dom}^*$, two device indexes j_0^*, j_1^* and a message m^*. If

– $(i^*, \cdot) \notin \mathcal{U}_{SET}$ or $j_0^* = j_1^*$, or

– $(i^*, j_0^*, \cdot) \notin \mathcal{D}_{SET}$ or $(i^*, j_1^*, \cdot) \notin \mathcal{D}_{SET}$, or
– $(i^*, j_0^*) \in \mathcal{CD}$ or $(i^*, j_1^*) \in \mathcal{CD}$, or
– the $\mathcal{O}_{\mathsf{GetRT}}$ oracle was called on input $(i^*, j_0^*, \mathtt{dom}^*)$ or $(i^*, j_1^*, \mathtt{dom}^*)$,

then the oracle returns $\perp$ and aborts. Otherwise, the oracle computes $\sigma \leftarrow \mathsf{Sign}(param, uSK_{i^*,j_b^*}, \mathtt{dom}^*, m^*)$ and returns σ.

After calling the $\mathcal{O}_{\mathsf{Challenge}}$ oracle, the adversary cannot call the $\mathcal{O}_{\mathsf{GetRT}}$ on input $(i^*, j_0^*, \mathtt{dom}^*)$ or $(i^*, j_1^*, \mathtt{dom}^*)$, and the $\mathcal{O}_{\mathsf{CorruptUser}}$ on input (i^*, j_0^*) or (i^*, j_1^*).

Experiment $\mathsf{Anon}_{\mathcal{A}}^{S}$**:**

- $(param) \leftarrow \mathsf{Setup}(1^\lambda)$.
- choose $b \in \{0,1\}$ at random,
- $\mathcal{O} \leftarrow \{\mathcal{O}_{\mathsf{CreateUser}}, \mathcal{O}_{\mathsf{GetNym}}, \mathcal{O}_{\mathsf{AddDevice}}, \mathcal{O}_{\mathsf{GetRT}}, \mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{CorruptUser}}, \mathcal{O}_{\mathsf{Challenge}}(b, \cdot, \cdot, \cdot, \cdot)\}$.
- $\hat{b} \leftarrow \mathsf{A}^{\mathcal{O}}(param)$.
- If $\hat{b} = b$, then output 1, otherwise output 0.

Definition 8. *A Pseudonymous Public Key Group Signature S is (t, ϵ)-anonymous if $|\Pr[\mathsf{Anon}_{\mathsf{A}}^{S}(\lambda) = 1] - \frac{1}{2}| \leq \epsilon$ for any adversary A running in time t.*

Domain Unlinkability. Informally, domain unlinkability means that it is infeasible to correlate two domain pseudonyms with a single user. We will give a simulation based definition for the domain unlinkability property.

First we need to define the following data structures: $\mathcal{D}$ denotes a set of domain names, $\mathcal{U}_{SET}^{I}$ is the set of user indexes, $\mathcal{K}$ denotes an associative map which maps a pair $(\mathtt{dom}, i) \in \{0,1\}^* \times \mathbb{N}$ into a master secret key from the secret key space $\mathcal{USK}$. Then we define an associative map $\mathcal{UK}$ which maps a tuple $(\mathtt{dom}, i, j) \in \{0,1\}^* \times \mathbb{N}^2$ into a device secret key.

Then we define the following oracles which implement the ideal functionality, where the keys of the user for different domains are independent (note that for Pseudonymous Public Key Group Signature they are the same):

$\mathcal{O}_{\mathsf{CreateUser}}^{Ideal}$: The query requests to create a secret key for the i-th user. If $i \notin \{1, \ldots, n\}$ or $i \in \mathcal{U}_{SET}^{I}$, then the oracle aborts. Otherwise, the oracle adds i to $\mathcal{U}_{SET}^{I}$ and for each $\mathtt{dom} \in \mathcal{D}$, the oracle chooses a secret key $mSK_{i,\mathtt{dom}}$ at random from $\mathcal{USK}$ and sets $\mathcal{K}[(i, \mathtt{dom})] \leftarrow mSK_{i,\mathtt{dom}}$.

$\mathcal{O}_{\mathsf{AddDevice}}^{Ideal}$: The query requests to create the j-th device for user i. For each $\mathtt{dom} \in \mathcal{D}$ the oracle obtains $mSK_{i,\mathtt{dom}} \leftarrow \mathcal{K}[(i, \mathtt{dom})]$ and runs $uSK_{\mathtt{dom},i,j} \leftarrow \mathsf{AddDevice}(param, mSK_{\mathtt{dom},i}, j)$, and sets $\mathcal{UK}[(\mathtt{dom}, i, j)] \leftarrow uSK_{\mathtt{dom},i,j}$.

$\mathcal{O}_{\mathsf{GetNym}}^{Ideal}$: The query requests the pseudonym of the i-th user with regards to a domain name $\mathtt{dom}$. If $i \notin \mathcal{U}_{SET}^{I}$, then the oracle aborts. If $\mathcal{K}[(i, \mathtt{dom})]$ is

undefined, then the oracle chooses a secret key $mSK_{i,\mathtt{dom}} \in \mathcal{USK}$ at random and sets $\mathcal{K}[(i, \mathtt{dom})] \leftarrow mSK_{i,\mathtt{dom}}$. Then the oracle runs $nym_{i,\mathtt{dom}} \leftarrow \mathsf{ComputePseudonym}(params, mSK_{i,\mathtt{dom}}, \mathtt{dom})$ and outputs $nym_{i,\mathtt{dom}}$.

$\mathcal{O}^{Ideal}_{\mathsf{GetRT}}$: The query requests a revocation token for the j-th device of user i with regards to a domain name $\mathtt{dom}$. If $i \notin \mathcal{U}^I_{SET}$, then the oracle aborts. If $\mathcal{UK}[(\mathtt{dom}, i, j)]$ is undefined, then the oracle runs the procedure $uSK_{\mathtt{dom},i,j} \leftarrow \mathsf{AddDevice}(param, mSK_{\mathtt{dom},i}, j)$, and sets $\mathcal{UK}[(\mathtt{dom}, i, j)] \leftarrow uSK_{\mathtt{dom},i,j}$. Then the oracle runs $uRT_{i,j,\mathtt{dom}} \leftarrow \mathsf{CreateRevocationToken}(param, mSK_{\mathtt{dom},i}, \mathtt{dom}, j)$ and outputs $uRT_{i,j,\mathtt{dom}}$.

$\mathcal{O}^{Ideal}_{\mathsf{Sign}}$: The query requests to sign a message m by the j-th device of user i with regards to a domain name $\mathtt{dom}$. If $i \notin \mathcal{U}^I_{SET}$, then the oracle aborts and returns $\perp$. If $\mathcal{UK}[(\mathtt{dom}, i, j)]$ is undefined, then the oracle runs $uSK_{\mathtt{dom},i,j} \leftarrow \mathsf{AddDevice}(param, mSK_{\mathtt{dom},i}, j)$, and sets $\mathcal{UK}[(\mathtt{dom}, i, j)] \leftarrow uSK_{\mathtt{dom},i,j}$. Finally, the oracle runs $\sigma \leftarrow \mathsf{Sign}(param, uSK_{\mathtt{dom},i,j}, \mathtt{dom}, m)$ and returns σ.

Definition 9. *We say that a Pseudonymous Public Key Group Signature S is (t, ϵ)-domain unlinkable if for any adversary A running in time t we have*

$$|\Pr[(param) \leftarrow \mathsf{Setup}(1^\lambda); A^{\mathcal{O}_{Real}}(param)] - \Pr[(param) \leftarrow \mathsf{Setup}(1^\lambda); A^{\mathcal{O}_{Ideal}}(param)]| \leq \epsilon,$$

where $\mathcal{O}_{Real} = \{\mathcal{O}_{\mathsf{CreateUser}}, \mathcal{O}_{\mathsf{AddDevice}}, \mathcal{O}_{\mathsf{GetNym}}, \mathcal{O}_{\mathsf{GetRT}}, \mathcal{O}_{\mathsf{Sign}}\}$ *and* $\mathcal{O}_{Ideal} = \{\mathcal{O}^{Ideal}_{\mathsf{CreateUser}}, \mathcal{O}^{Ideal}_{\mathsf{AddDevice}}, \mathcal{O}^{Ideal}_{\mathsf{GetNym}}, \mathcal{O}^{Ideal}_{\mathsf{GetRT}}, \mathcal{O}^{Ideal}_{\mathsf{Sign}}\}$.

4 Efficient Construction

4.1 Scheme Specification

In this section we describe our implementation of a Pseudonymous Public Key Group Signature.

The idea behind the construction is as follows. First a user chooses a secret key for the Boneh-Boyen signature scheme [14], i.e. $z \in \mathbb{Z}_p$ chosen at random. This key is then used to compute "pseudonymized" public keys as $nym \leftarrow \mathsf{H}_0(\mathtt{dom})^z$, where H_0 is a hash function and $\mathtt{dom}$ is a domain name. The same key is then used to issue Boneh-Boyen signatures $A_j \leftarrow g_1^{1/(z+u_j)}$ on a secret key $u_j \in \mathbb{Z}_p$ of his device j. Note that according to our security definition from Sect. 3, the user generates all secret keys for his devices and we do not define a $\mathsf{Join/Issue}$ procedure to ensure exculpability[1]. We intentionally defined our group signature scheme in this way due to specific use case.

Now, a device j holding a "certified" secret key (u_j, A_j), computes a signature of knowledge which is based on a Σ-protocol and turned into a signature scheme using the Fiat-Shamir paradigm. Informally, the signature carries a proof

[1] The exculpability property is known from dynamic group signatures [2] and assures that even the group manager cannot forge signatures on behalf of a user.

that the signer knows a secret key with a certificate which verifies correctly with a "pseudonymized" public key nym. The tricky part of our construction is that the signer does not know the "pseudonymized" public key to which his certificate verifies. The only information which allows to sign with regards to a pseudonymized public key is the basis of the public key, i.e. $\hat{g_2} \leftarrow \mathsf{H}_0(\mathtt{dom})$.

Bellow, we describe our scheme more formally.

$\mathsf{Setup}(1^\lambda)$:

1. Choose groups $\mathbb{G}_1$, $\mathbb{G}_2$ of a prime order p, a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, and choose a generator $g_1 \xleftarrow{\mathcal{R}} \mathbb{G}_1$ at random.
2. Define a hash function H_0 which maps into $\mathbb{G}_2$ and a hash function H which maps into $\mathbb{Z}_p$.
3. Output the global parameters $param = (p, \mathbb{G}_1, \mathbb{G}_2, e, g_1, \mathsf{H}_0, \mathsf{H})$.

$\mathsf{CreateUser}(param)$:

1. Choose $z \in \mathbb{Z}_p$ at random and output $mSK \leftarrow z$.

$\mathsf{ComputePseudonym}(param, mSK, \mathtt{dom})$:

1. Compute $\hat{g_2} \leftarrow \mathsf{H}_0(\mathtt{dom})$ and output $nym \leftarrow \hat{g_2}^z$.

$\mathsf{AddDevice}(param, mSK, i)$:

1. Choose $u_i \in \mathbb{Z}_p$ at random[2].
2. Compute $A_i = g_1^{1/(u_i+z)}$, return $uSK[i] \leftarrow (A_i, u_i)$ and store u_i for future use.

$\mathsf{CreateRevocationToken}(param, mSK, \mathtt{dom}, i)$:

1. Retrieve the user secret key u_i, compute $\hat{g_2} \leftarrow \mathsf{H}_0(\mathtt{dom})$ and return $uRT \leftarrow \hat{g_2}^{u_i}$.

$\mathsf{Sign}(param, uSK, \mathtt{dom}, m)$:

1. Compute $\hat{g_2} \leftarrow \mathsf{H}_0(\mathtt{dom})$.
2. Choose $(r_1, r_2) \in \mathbb{Z}_p^2$ at random and compute $R_1 \leftarrow A_i^{r_1}$, $R_2 \leftarrow g_1^{r_2}$ and $R_3 \leftarrow e(R_2, \hat{g_2})^{u_i}$.
3. Compute the following signature of knowledge:

$$S \leftarrow SoK\{(\alpha, \beta, \gamma) : R_1 = g_1^{\beta/(z+\alpha)} \wedge R_2 = g_1^{\gamma} \wedge R_3 = e(g_1, \hat{g_2})^{\alpha \cdot \gamma}\}(m)$$

   (a) Choose $t_1, t_2, t_3 \in \mathbb{Z}_p$ at random and compute

$$T_1 \leftarrow e(A_i, \hat{g_2})^{-t_1 \cdot r_1} \cdot e(g_1, \hat{g_2})^{t_2}\ ,\ T_2 \leftarrow g_1^{t_3} \qquad \textit{and} \qquad T_3 \leftarrow e(R_2, \hat{g_2})^{t_1}.$$

   (b) Compute the challenge $c = \mathsf{H}(param,\ m,\ \mathtt{dom},\ T_1,\ T_2,\ T_3)$.
   (c) Compute $s_1 \leftarrow t_1 + c \cdot u_i$, $s_2 \leftarrow t_2 + c \cdot r_1$ and $s_3 \leftarrow t_3 + c \cdot r_2$.
   (d) Set $S = (c, s_1, s_2, s_3)$
4. Output the signature $\sigma = (S, R_1, R_2, R_3)$.

$\mathsf{Verify}(param, nym, \mathtt{dom}, \sigma, m, uRT)$:

1. Compute $\hat{g_2} \leftarrow \mathsf{H}_0(\mathtt{dom})$.
2. Parse the signature as $\sigma = (S, R_1, R_2, R_3)$, where $S = (c, s_1, s_2, s_3)$.

[2] This value may be derived in a deterministic way, e.g. $u_i \leftarrow \mathsf{H}(z, i)$.

3. Restore the values

$$\tilde{T_1} = e(R_1, nym)^{-c} \cdot e(R_1, \hat{g_2})^{-s_1} \cdot e(g_1, \hat{g_2})^{s_2}$$
$$\tilde{T_2} = g_1^{s_3} \cdot R_2^{-c}$$
$$\tilde{T_3} = e(R_2, \hat{g_2})^{s_1} \cdot R_3^{-c}$$

4. If $c \neq \mathsf{H}(param, m, \mathsf{dom}, \tilde{T_1}, \tilde{T_2}, \tilde{T_3})$, then return 0 (reject).
5. If $e(R_2, uRT) = R_3$, then return 0 (reject).
6. Return 1 (accept).

Theorem 1. *Pseudonymous Public Key Group Signature is correct.*

Proof. The proof is simply due the inspection of the following equations:

$$\tilde{T_1} = e(R_1, nym)^{-c} \cdot e(R_1, \hat{g_2})^{-s_1} \cdot e(g_1, \hat{g_2})^{s_2} =$$
$$\left(e(R_1, \hat{g_2})^{-t_1} \cdot e(g_1, \hat{g_2})^{t_2}\right) \cdot e(R_1, nym)^{-c} \cdot e(R_1, \hat{g_2}^{-c \cdot u_i}) \cdot e(g_1^{c \cdot r_1}, \hat{g_2}) =$$
$$T_1 \cdot e(A_i^{r_1}, \hat{g_2}^{-c \cdot z} \cdot \hat{g_2}^{-c \cdot u_i}) \cdot e(g_1^{c \cdot r_1}, \hat{g_2}) = T_1 \cdot e(g_1, \hat{g_2})^{-r_1 \cdot c} \cdot e(g_1, \hat{g_2})^{r_1 \cdot c} = T_1$$
$$\tilde{T_2} = g_1^{s_3} \cdot R_2^{-c} = g_1^{t_3} \cdot g_1^{c \cdot r_2} \cdot g_1^{-c \cdot r_2} = T_2$$
$$\tilde{T_3} = e(R_2, \hat{g_2})^{s_1} \cdot R_3^{-c} = e(R_1, \hat{g_2})^{t_1} \cdot e(g_1, \hat{g_2})^{r_2 \cdot c \cdot u_i} \cdot e(g_1, \hat{g_2})^{-r_2 \cdot c \cdot u_i} = T_3$$

For the revocation procedure, let $uRT \leftarrow \hat{g_2}^{u_i}$ be a revocation token. Then we have $e(R_2, uRT) = e(g_1^{r_2}, \hat{g_2}^{u_i}) = R_3$.

4.2 Security Analysis

Due to space limitation we give only an intuition behind the security proofs. A detailed formal analysis is postponed to the full version of the paper.

Zero-Knowledge and Witness Extraction. Our construction is based on a known technique of using a Σ-protocol converted into a signature scheme via the Fiat-Shamir heuristic. For such a construction we may show that, in the random oracle model, there is a witness extractor (so the protocol is a proof of knowledge) and a simulator (so the protocol is zero-knowledge). Using the witness extractor, from a forged signature for a pseudonym *nym* within domain $\mathtt{dom}$, we may extract values $\tilde{u}$, $\tilde{r_1}$, $\tilde{r_2}$ and $\tilde{A}$, such that $g_1^{\tilde{r_2}} = R_2$, $e(R_2, \mathsf{H}_0(\mathtt{dom}))^{\tilde{u}} = R_3$ and $e(g_1, \mathsf{H}_0(\mathtt{dom})) = e(\tilde{A}, \mathsf{H}_0(\mathtt{dom})^{\tilde{u}} \cdot nym)$. Using the simulator we may generate a correct signature having only g_1, $\hat{g_2}$, *nym* and a revocation token $\mathsf{H}_0(\mathtt{dom})^{\tilde{u}}$.

Lemma 1. *The protocol has an extractor.*

Proof. Suppose we can rewind the Prover to the moment when he is given the challenge c. The Prover will send R_1, R_2, R_3, T_1, T_2 and T_3 and respond with the challenge c and s_1, s_2 and s_3. Then we rewind to the step when the Prover obtains c and send a different challenge $c' \neq c$. The Prover will answer with s_1',

s_2' and s_3' satisfying the verification equations. Let $\Delta s_i = (s_i - s_i')$ for $i = 1, 2, 3$ and $\Delta c = (c - c')$. From the equality

$$T_1 = e(R_1, nym)^{-c} \cdot e(R_1, \hat{g_2})^{-s_1} \cdot e(g_1, \hat{g_2})^{s_2} = e(R_1, nym)^{-c'} \cdot e(R_1, \hat{g_2})^{-s_1'} \cdot e(g_1, \hat{g_2})^{s_2'}$$

we have that

$$\begin{aligned} e(R_1, \hat{g_2})^{-\Delta s_1} \cdot e(g_1, \hat{g_2})^{\Delta s_2} &= e(R_1, nym)^{\Delta c} \\ e(R_1, \hat{g_2})^{-\Delta s_1/\Delta c} \cdot e(g_1, \hat{g_2})^{\Delta s_2/\Delta c} &= e(R_1, nym) \\ e(g_1, \hat{g_2})^{\Delta s_2/\Delta c} &= e(R, \hat{g_2}^{\Delta s_1/\Delta c} \cdot nym) \\ e(g_1, \hat{g_2}) &= e(R_1^{(\Delta s_2/\Delta c)^{-1}}, \hat{g_2}^{\Delta s_1/\Delta c} \cdot nym) \end{aligned}$$

Then from $T_2 = g_1^{s_3} \cdot R_2^{-c} = g_1^{s_3'} \cdot R_2^{-c'}$ we have

$$\begin{aligned} g_1^{\Delta s_3} &= R_2^{\Delta c} \\ g_1^{\Delta s_3/\Delta c} &= R_2 \end{aligned}$$

So, we may compute $\tilde{u} = \Delta s_1/\Delta c$, and $\tilde{r_1} = \Delta s_2/\Delta c$, $\tilde{r_2} = \Delta s_3/\Delta c$ and $\tilde{A} = R^{\tilde{r_1}^{-1}}$ such that $g_1^{\tilde{r_2}} = R_2$ and $e(g_1, \hat{g_2}) = e(\tilde{A}, \hat{g_2}^{\tilde{u}} \cdot nym)$.

Finally, from $T_3 = e(R_2, \hat{g_2})^{s_1} \cdot R_3^{-c} = e(R_2, \hat{g_2})^{s_1'} \cdot R_3^{-c'}$ we have

$$\begin{aligned} e(R_2, \hat{g_2})^{\Delta s_1} &= e(R_3, \hat{g_2})^{\Delta c} \\ e(g_1, \hat{g_2})^{\tilde{r_2} \cdot \tilde{u}} &= R_3. \end{aligned}$$

Lemma 2. *The protocol is Zero-Knowledge.*

Proof. Given the common input ($\mathbb{G}_1$, $\mathbb{G}_2$, e, g_1, $\hat{g_2}$, nym), where $\hat{g_2} = \mathsf{H}_0(\mathtt{dom})$ and $nym = \hat{g_2}^z$ for some $z \in \mathbb{Z}_p$, the simulator works as follows. Choose $R_3 \overset{\mathcal{R}}{\leftarrow} \mathbb{G}_1$ and $R_2 \overset{\mathcal{R}}{\leftarrow} \mathbb{G}_1$. Note that now the value of A_i is fixed by the choice of R_3 and R_2. In order to highlight this we may denote $R_2 = g_1^{r_2}$ and $R_3 = e(R_2, \hat{g_2})^{u_i}$ for some $u_i \in \mathbb{Z}_p$, hence we have $A = g_1^{1/(u_i+z)}$. Choose $R_1 \overset{\mathcal{R}}{\leftarrow} \mathbb{G}_1$ at random. See that $R_1 = A^{r_1/(u+z)}$ for some r_1, thus the values R_1, R_2, R_3 are distributed as in a real protocol. Now, the simulator chooses $(c, s_1, s_2, s_3) \overset{\mathcal{R}}{\leftarrow} \mathbb{Z}_p^4$ and computes

$$\begin{aligned} T_1 &\leftarrow e(R_1, nym)^{-c} \cdot e(R_1, \hat{g_2})^{-s_1} \cdot e(g_1, \hat{g_2})^{s_2}, \\ T_2 &\leftarrow g_1^{s_3} \cdot R_2^{-c} \quad \text{and} \quad T_3 \leftarrow e(R_2, \hat{g_2})^{s_1} \cdot R_3^{-c}. \end{aligned}$$

Obviously, T_1, T_2 and T_3 along with the values c, s_1, s_2, s_3 satisfy the verification equations. Moreover, R_1, R_2, R_3, c, s_1, s_2, s_3 are uniformly distributed as in the real executions, so the simulation is perfect.

Theorem 2. *If DLP is (ϵ', t')-hard in $\mathbb{G}_2$, then the Pseudonymous Public Key Group Signature is (ϵ, t)-unforgeable, where $\epsilon \approx q_U \cdot n \cdot \sqrt{q_{\mathsf{H}}(\epsilon' + 1/p)}$ and $t \approx t'$, and n, q_U and q_{H} are the upper bounds on the number of invocations of, respectively, $\mathcal{O}_{\mathsf{CreateUser}}$, $\mathcal{O}_{\mathsf{AddDevice}}$ and hash queries.*

The unforgeability property relies on the DLP problem. Here we put a DL problem instance $\Lambda \in \mathbb{G}_2$ into the revocation tokens of a chosen device. We may program the random oracle to output $g_2^{r_{\mathtt{dom}}} \leftarrow \mathsf{H}_0(\mathtt{dom})$, and then we may compute the revocation tokens as $uRT \leftarrow \Lambda^{r_{\mathtt{dom}}}$. If the adversary successfully forges a signature for that device, we use the extractor and extract the discrete logarithm $\alpha = \log_{g_2}(\Lambda)$.

Theorem 3. *If q-CAA is (ϵ', t')-hard in $\mathbb{G}_1$, then the Pseudonymous Public Key Group Signature is (ϵ, t)-seclusive, where $\epsilon \approx n \cdot \sqrt{q_{\mathsf{H}}(\epsilon' + 1/p)}$, $t \approx t'$, and n, q and q_{H} are the upper bounds on the number of, respectively, $\mathcal{O}_{\mathsf{CreateUser}}$, $\mathcal{O}_{\mathsf{AddDevice}}$ and hash queries.*

Seclusiveness follows from the fact that device secret keys are CAA instances, i.e. they consist of pairs $(u, g_1^{1/(u+z)}) \in \mathbb{Z}_p \times \mathbb{G}_1$. If an adversary would forge a signature, then from the extractor we may obtain a pair $(\tilde{u}, \tilde{A})$. If the forged signature cannot be revoked, then from the revocation equation $e(R_2, \hat{g_2}^{u_i}) \neq R_3$ follows that $\tilde{u} \neq u_i$ for each device secret key u_i issued by the user holding z. Thus $(\tilde{u}, \tilde{A})$, is the solution to the CAA problem instance.

Theorem 4. *If BDDH is (ϵ', t')-hard in $\mathbb{G}_1, \mathbb{G}_2$, then the Pseudonymous Public Key Group Signature is (ϵ, t)-anonymous, where $\epsilon \approx \frac{\epsilon'^2}{n \cdot q_{\mathsf{H}} \cdot q_U^2}$, $t \approx t'$, and n, q_U and q_{H} are upper bounds on the number of, respectively, $\mathcal{O}_{\mathsf{CreateUser}}$, $\mathcal{O}_{\mathsf{AddDevice}}$ and hash queries.*

In order to proof anonymity, we describe a sequence of games. We will start with a game where the challenge signature is returned by the device j_0^* (bit $b = 0$). Then in each game we change the protocol execution so that the adversary has only a negligible chance of noticing these changes. Finally, we will end up in a game where the challenge signature is computed for user j_1^* (bit $b = 1$).

The strategy of changing the protocol is as follows. First we need to simulate the signatures for all devices. Then, for the j_0^*-th device, under the BDDH assumption, we choose these values independently at random. Next, instead of choosing R_1, R_2, R_3 independently we compute these values as for device j_1^*. Below, we shed some light on the step of changing the values of R_1, R_2, R_3 into random values.

Let (g_2^a, g_2^b, g_1^c) be a BDDH problem instance and let $\mathtt{dom}^*$ denote the domain from the challenge oracle. In all domains $\mathtt{dom} \neq \mathtt{dom}^*$ we choose $r_{\mathtt{dom}}$ at random, program the hash oracle to output $g_2^{r_{\mathtt{dom}}} \leftarrow \mathsf{H}_0(\mathtt{dom})$ and we compute $uRT \leftarrow (g_2^a)^{r_{\mathtt{dom}}}$ and $R_3 \leftarrow e(g_1^{r_{\mathtt{dom}}}, g_2^a)^{r_2}$ for device j_0^*. In domain $\mathtt{dom}^*$, we program the hash oracle to return $g_2^b \leftarrow \mathsf{H}_0(\mathtt{dom}^*)$. Then, we choose $r_2 \xleftarrow{\mathcal{R}} \mathbb{Z}_p$ at random, compute $R_2 \leftarrow g_1^{c \cdot r_2}$ and $R_3 \leftarrow e(g_1, g_2)^{abc \cdot r_2}$(the current game) or R_3 is chosen at random (the next game). Note that if an adversary would distinguish whether $R_3 = e(g_1, g_2)^{abc \cdot r_2}$ or R_3 is random, then it would break the BDDH assumption.

Theorem 5. *If SXDH is (ϵ', t')-hard in $\mathbb{G}_1$, then the Pseudonymous Public Key Group Signature is (ϵ, t)-domain unlinkable, where $\epsilon \approx \epsilon' \cdot q_{\mathsf{H}}(q_U + n)$, $t \approx t'$, and n, q_U and q_{H} are the upper bounds on the number of, respectively, $\mathcal{O}_{\mathsf{CreateUser}}$, $\mathcal{O}_{\mathsf{AddDevice}}$ and hash queries.*

Domain unlinkability follows from the fact that we may simulate the signatures for each device and that in each domain we have a distinct base $\hat{g_2} \leftarrow \mathsf{H}_0(\mathtt{dom})$. Note that the device revocation tokens are computed as $uRT_{i,j,\mathtt{dom}} = \hat{g_2}^{u_j}$ for a device secret key $u_j \in \mathbb{Z}_p$. In a given domain we may choose $uRT_{i,j,\mathtt{dom}}$ at random. It is easy to see that if an adversary would recognize this change, he would serve as a distinguisher for the SXDH problem. In the proof, we need to choose revocation tokens of all devices in all domains at random. Finally, we may use the same reasoning to choose pseudonyms $nym = \mathsf{H}_0(\mathtt{dom})$ at random in each domain, finally ending up in an ideal system as defined in Sect. 3.

4.3 Additional Procedures and Scheme Variants

Here we describe briefly some additional procedures and variations of our scheme, which may be useful for certain practical situations.

First, note that the signing device needs only to know his private key consisting of an SDH pair $(u, g_1^{1/(z+u)})$ and nothing else, in order to create a signature. In particular, the signing device does not need to know the public key, aka the pseudonym, with which the signature will later be verified. Moreover, it seems that the signing device alone is not even able to compute the pseudonym by itself. However, in some cases it may be desirable that the signing device can compute a pseudonym, what in our case may be $nym' = e(Z, \hat{g_2})$, assuming the user also issues the value $Z = g_1^z$. Such nym' may serve as a temporal pseudonym, until the owner of the device confirms this pseudonym by proving his knowledge of the secret key $z \in \mathbb{Z}_p$.

Proving the knowledge of the master key may be required as a part of user registration. This may be simply done by designing a Σ-protocol [15] which will prove the knowledge of $\log_{g_1}(nym)$. Such standard protocol may be transformed into a zero-knowledge proof of knowledge protocol or into its non-interactive version in the random oracle model.

5 Conclusions

Beyond the concrete application case of delegating the rights by a user to multiple own devices, we have introduced a novel notion for group signature schemes. It expands the functionality of group signatures by adding the feature that group public keys may be pseudonyms derived ad hoc.

We have introduced a security framework for our scheme supporting strong privacy protection on one hand, and revocation capabilities on the other hand.

Finally, we have designed a scheme based on bilinear groups which implements such a system. Even if it uses bilinear groups and pairings, it is relatively simple and implementable on relatively weak devices. Note that the user's root of trust may be a relatively weak device, since no procedure executed by it requires computation of pairings. They are needed for signature creation (this can be done by smart phones) and verification (on strong servers).

References

1. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003). doi:10.1007/3-540-39200-9_38
2. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: the case of dynamic groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005)
3. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
4. Boyen, X., Waters, B.: Full-domain subgroup hiding and constant-size group signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007)
5. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, pp. 168–177. ACM, New York (2004)
6. Kiayias, A., Tsiounis, Y., Yung, M.: Traceable signatures. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 571–589. Springer, Heidelberg (2004)
7. Trolin, M., Wikström, D.: Hierarchical group signatures. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 446–458. Springer, Heidelberg (2005)
8. Ali, S.T., Amberker, B.B.: Dynamic attribute based group signature with attribute anonymity and tracing in the standard model. In: Gierlichs, B., Guilley, S., Mukhopadhyay, D. (eds.) SPACE 2013. LNCS, vol. 8204, pp. 147–171. Springer, Heidelberg (2013)
9. Han, S., Wang, J., Liu, W.: An efficient identity-based group signature scheme over elliptic curves. In: Freire, M.M., Chemouil, P., Lorenz, P., Gravey, A. (eds.) ECUMN 2004. LNCS, vol. 3262, pp. 417–429. Springer, Heidelberg (2004)
10. 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.) FC 2014. LNCS, vol. 8437, pp. 252–269. Springer, Heidelberg (2014)
11. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, pp. 132–145. ACM, New York (2004)
12. Camenisch, J., Mödersheim, S., Sommer, D.: A formal model of identity mixer. In: Kowalewski, S., Roveri, M. (eds.) FMICS 2010. LNCS, vol. 6371, pp. 198–214. Springer, Heidelberg (2010)
13. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Appl. Math. **156**(16), 3113–3121 (2008)
14. Boneh, D., Boyen, X.: Short signatures without random oracles and the sdh assumption in bilinear groups. J. Cryptol. **21**(2), 149–177 (2008)
15. Damgård, I.: On Σ-protocols. Lecture notes for CPT, v. 2

http://www.springer.com/978-3-319-46297-4

Network and System Security
10th International Conference, NSS 2016, Taipei, Taiwan, September 28-30, 2016, Proceedings
Chen, J.; Piuri, V.; Su, C.; Yung, M. (Eds.)
2016, XIV, 540 p. 128 illus., Softcover
ISBN: 978-3-319-46297-4