Ad-Hoc-Domain Signatures for Personal eID Documents

Kamil Kluczniak, Lucjan Hanzlik, Mirosław Kutyłowski

Wrocław University of Science and Technology, Poland

ArcticCrypt 2016, Longyearbyen, Svalbard
Overview

1. Domain Pseudonymous Signatures
2. Ad Hoc Domain Signatures - Formal Models
3. Scheme
4. Open Problems
Motivation

- **eIDAS - EU REGULATION No 910/2014**
  identification, authentication and other trust services in the European market

- **growing scope of usage of electronic documents**
  reliable authentication of documents badly needed. Electronic signatures one of a few reliable choices.

- **“Privacy by Design” paradigm**
  a technical system must be designed in a way that protects privacy
  privacy protection is a fundamental security condition
Pseudonym:
A unique ID in each service that does not reveal the real identity

preventing Sybil attacks: appearing under different IDs in the same service.
Domain Signatures:

1. one user - just one private key for all domains
Domain Pseudonyms Signatures Concept

Domain Signatures:

1. one user - just one private key for all domains
2. domain pseudonym acts as a public key
Domain Pseudonyms Signatures Concept

Domain Signatures:

1. one user - just one private key for all domains
2. domain pseudonym acts as a public key
3. verification related to the domain pseudonym
Domain Pseudonyms Signatures Concept

Domain Signatures:

1. One user - just one private key for all domains
2. Domain pseudonym acts as a public key
3. Verification related to the domain pseudonym
4. Verification must not reveal the real identity
Domains and Requirements

Domain/Sector

Service area where the user must appear under the same (pseudonymous) identity.

like a user account
Domains and Requirements

**Domain/Sector**

Service area where the user must appear under the same (pseudonymous) identity.

like a user account

**Unlinkability**

The pseudonyms in different sectors must be unlinkable.
Domains and Requirements

**Domain/Sector**

Service area where the user must appear under the same (pseudonymous) identity.

like a user account

**Unlinkability**

The pseudonyms in different sectors must be unlinkable.

**Seclusiveness**

Only the Issuer may create/admit new users.

like for issuing personal ID cards
Revocation

The Issuer can revoke a user within a domain. Like for stolen personal ID cards.
Requirements

Revocation
The Issuer can revoke a user within a domain.
like for stolen personal ID cards

Pseudonym Uniqueness - Resistance to Sybil attacks
A user may have just one pseudonym per domain.
previous work was focused on this, but surprisingly a formal requirement was missing
Comparison to Direct Anonymous Attestation

Domain Signatures

Direct Anonymous Attestation
## Comparison to Direct Anonymous Attestation

<table>
<thead>
<tr>
<th></th>
<th><strong>Ad Hoc DS:</strong></th>
<th><strong>DAA:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Environment:</strong></td>
<td>Smart Cards</td>
<td>Host with TPM</td>
</tr>
<tr>
<td><strong>Privacy issues:</strong></td>
<td>a reader is a privacy threat</td>
<td>host is <strong>NOT</strong> a privacy threat</td>
</tr>
<tr>
<td><strong>Revocation method:</strong></td>
<td>blacklist a pseudonym</td>
<td>publish the secret key</td>
</tr>
<tr>
<td><strong>Updating the state of a device:</strong></td>
<td>Impossible</td>
<td>Possible</td>
</tr>
</tbody>
</table>
## Comparison to Direct Anonymous Attestation

<table>
<thead>
<tr>
<th></th>
<th>Ad Hoc DS:</th>
<th>DAA:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment:</td>
<td>Smart Cards</td>
<td>Host with TPM</td>
</tr>
<tr>
<td>Privacy issues:</td>
<td>a reader is a privacy threat</td>
<td>host is NOT a privacy threat</td>
</tr>
<tr>
<td>Revocation method:</td>
<td>blacklist a pseudonym</td>
<td>publish the secret key</td>
</tr>
<tr>
<td>Updating the state of a device:</td>
<td>Impossible</td>
<td>Possible</td>
</tr>
</tbody>
</table>

- differences mainly implied by the execution environment
## Comparison to Direct Anonymous Attestation

<table>
<thead>
<tr>
<th></th>
<th>Ad Hoc DS:</th>
<th>DAA:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment:</td>
<td>Smart Cards</td>
<td>Host with TPM</td>
</tr>
<tr>
<td>Privacy issues:</td>
<td>a reader is a privacy threat</td>
<td>host is NOT a privacy threat</td>
</tr>
<tr>
<td>Revocation method:</td>
<td>blacklist a pseudonym</td>
<td>publish the secret key</td>
</tr>
<tr>
<td>Updating the state of a device:</td>
<td>Impossible</td>
<td>Possible</td>
</tr>
</tbody>
</table>

- Differences mainly implied by the execution environment
- In contrast to Domain Signatures, DAA does not have a revocation method without publishing the secret key
Domain Signatures - Formal Definition

Procedures

**Setup:** \( \text{Setup}(1^k) \rightarrow (gPK, iSK) \)
Domain Signatures - Formal Definition

Procedures

**Setup**: \( \text{Setup}(1^k) \rightarrow (gPK, iSK) \)

**Join/Issue**: \( (uSK[i]) \leftarrow \text{Join}(gPK, i) \leftrightarrow \text{Issue}(gPK, iSK, uRT) \rightarrow (uRT[i]) \)
Domain Signatures - Formal Definition

Procedures

Setup: $\text{Setup}(1^k) \rightarrow (gPK, iSK)$

Join/Issue: $(uSK[i]) \leftarrow \text{Join}(gPK, i) \leftrightarrow \text{Issue}(gPK, iSK, uRT) \rightarrow (uRT[i])$

Generate Pseudonym: $\text{NymGen}(gPK, \text{dom}, uSK[i]) \rightarrow nym$
## Domain Signatures - Formal Definition

### Procedures

**Setup:**  \( \text{Setup}(1^k) \rightarrow (gPK, iSK) \)

**Join/Issue:**  \( (uSK[i]) \leftarrow \text{Join}(gPK, i) \leftrightarrow \text{Issue}(gPK, iSK, uRT) \rightarrow (uRT[i]) \)

**Generate Pseudonym:**  \( \text{NymGen}(gPK, \text{dom}, uSK[i]) \rightarrow nym \)

**Generate Domain Revocation Token:**  \( \text{DomainRevocationTokenGen}(gPK, \text{dom}, uRT[i]) \rightarrow dRT[i] \)
Domain Signatures - Formal Definition

Procedures

**Setup:** \( \text{Setup}(1^k) \rightarrow (gPK, iSK) \)

**Join/Issue:** \((uSK[i]) \leftarrow \text{Join}(gPK, i) \leftrightarrow \text{Issue}(gPK, iSK, uRT) \rightarrow (uRT[i]) \)

**Generate Pseudonym:** \( \text{NymGen}(gPK, \text{dom}, uSK[i]) \rightarrow \text{nym} \)

**Generate Domain Revocation Token:**
\( \text{DomainRevocationTokenGen}(gPK, \text{dom}, uRT[i]) \rightarrow dRT[i] \)

**Revocation Check:** \( \text{RevocationCheck}(dPK, \text{dom}, \text{nym}, dRT[i]) \rightarrow \{0, 1\} \)
Domain Signatures - Formal Definition

Procedures

Setup: \(\text{Setup}(1^k) \rightarrow (gPK, iSK)\)

Join/Issue: \((uSK[i]) \leftarrow \text{Join}(gPK, i) \leftrightarrow \text{Issue}(gPK, iSK, uRT) \rightarrow (uRT[i])\)

Generate Pseudonym: \(\text{NymGen}(gPK, \text{dom}, uSK[i]) \rightarrow nym\)

Generate Domain Revocation Token:
\(\text{DomainRevocationTokenGen}(gPK, \text{dom}, uRT[i]) \rightarrow dRT[i]\)

Revocation Check: \(\text{RevocationCheck}(dPK, \text{dom}, nym, dRT[i]) \rightarrow \{0, 1\}\)

Sign: \(\text{Sign}(gPK, \text{dom}, uSK[i], m) \rightarrow \sigma\)
Domain Signatures - Formal Definition

Procedures

Setup: \(\text{Setup}(1^k) \rightarrow (gPK, iSK)\)

Join/Issue: \((uSK[i]) \leftarrow \text{Join}(gPK, i) \leftrightarrow \text{Issue}(gPK, iSK, uRT) \rightarrow (uRT[i])\)

Generate Pseudonym: \(\text{NymGen}(gPK, \text{dom}, uSK[i]) \rightarrow nym\)

Generate Domain Revocation Token:
\(\text{DomainRevocationTokenGen}(gPK, \text{dom}, uRT[i]) \rightarrow dRT[i]\)

Revocation Check: \(\text{RevocationCheck}(dPK, \text{dom}, nym, dRT[i]) \rightarrow \{0, 1\}\)

Sign: \(\text{Sign}(gPK, \text{dom}, uSK[i], m) \rightarrow \sigma\)

Verify: \(\text{Verify}(gPK, \text{dom}, nym, m, \sigma) \rightarrow \{0, 1\}\):
1. The adversary obtains Issuer’s secret key
Unforgeability

1. The adversary obtains Issuer’s secret key

2. The adversary may:
   - add new honest users – as the Issuer,
   - ask for pseudonyms, signatures and user secret keys.
Unforgeability

1. The adversary obtains Issuer’s secret key

2. The adversary may:
   - add new honest users – as the Issuer,
   - ask for pseudonyms, signatures and user secret keys.

3. The adversary returns a pseudonym \( nym \), a domain \( dom \) and a signature \( \sigma \) on message \( m \), and wins if:
Unforgeability

1. The adversary obtains Issuer’s secret key

2. The adversary may:
   - add new honest users – as the Issuer,
   - ask for pseudonyms, signatures and user secret keys.

3. The adversary returns a pseudonym \( nym \), a domain \( dom \) and a signature \( \sigma \) on message \( m \), and wins if:
   - The signature \( \sigma \) verifies correctly with respect to \( nym \) and \( dom \).
Unforgeability

1. The adversary obtains Issuer’s secret key

2. The adversary may:
   - add new honest users – as the Issuer,
   - ask for pseudonyms, signatures and user secret keys.

3. The adversary returns a pseudonym \( nym \), a domain \( dom \) and a signature \( \sigma \) on message \( m \), and wins if:
   - The signature \( \sigma \) verifies correctly with respect to \( nym \) and \( dom \)
   - The revocation token of some user \( i \) revokes \( nym \).
Unforgeability

1. The adversary obtains Issuer’s secret key.
2. The adversary may:
   - add new honest users – as the Issuer,
   - ask for pseudonyms, signatures and user secret keys.
3. The adversary returns a pseudonym $nym$, a domain $dom$ and a signature $\sigma$ on message $m$, and wins if:
   - The signature $\sigma$ verifies correctly with respect to $nym$ and $dom$.
   - The revocation token of some user $i$ revokes $nym$.
   - The adversary has not asked for the secret key of this user.
The adversary creates all users by interacting with the Issuer.
(all users are under control of the adversary)
Seclusiveness

1. The adversary creates all users by interacting with the Issuer.
(adjacent users are under control of the adversary)

2. The adversary returns a pseudonym $nym$, a domain $dom$ and a signature $\sigma$ on a message $m$. 
Seclusiveness

1. The adversary creates all users by interacting with the Issuer.
   (all users are under control of the adversary)

2. The adversary returns a pseudonym $nym$, a domain $dom$ and a signature $\sigma$ on a message $m$.

3. The adversary, and wins if:
   - The signature $\sigma$ verifies correctly with respect to $nym$ and $dom$. 
Seclusiveness

1. The adversary creates all users by interacting with the Issuer.
   (all users are under control of the adversary)

2. The adversary returns a pseudonym \textit{nym}, a domain \textit{dom} and a signature \(\sigma\) on a message \(m\).

3. The adversary , and wins if:
   - The signature \(\sigma\) verifies correctly with respect to \textit{nym} and \textit{dom}.
   - No revocation token created by the Issuer revokes \textit{nym}.
1. The adversary obtains the Issuer’s secret key.
The adversary obtains the Issuer’s secret key.

His goal is to return a revocation token $uRT$, a domain $dom$, and tuples $(m_0, nym_0, \sigma_0)$ and $(m_1, nym_1, \sigma_1)$. 
1. The adversary obtains the Issuer’s secret key.

2. His goal is to return a revocation token $uRT$, a domain $dom$, and tuples $(m_0, nym_0, \sigma_0)$ and $(m_1, nym_1, \sigma_1)$.

3. The adversary wins if

   - signatures $\sigma_0, \sigma_1$ verify correctly with respect to $(m_0, nym_0)$ and $(m_1, nym_1)$, respectively,
Ad-Hoc-Domain Signatures

---

Klucznik, Hanzlik, Kutyłowski

---

Domain Signatures

Models

Scheme

Problems

---

Pseudonym Uniqueness

1. The adversary obtains the Issuer’s secret key.

2. His goal is to return a revocation token $uRT$, a domain $dom$, and tuples $(m_0, nym_0, \sigma_0)$ and $(m_1, nym_1, \sigma_1)$.

3. The adversary wins if
   - signatures $\sigma_0$, $\sigma_1$ verify correctly with respect to $(m_0, nym_0)$ and $(m_1, nym_1)$, respectively,
   - $uRT$ revokes both $nym_0$ and $nym_1$. 
Note that in each experiment, **the challenger identifies the signer** (or may identify that no such signer exist).
Note that in each experiment, the challenger identifies the signer (or may identify that no such signer exist).

In Direct Anonymous Attestation the challenger cannot identify the signer...
Note - Identification of a User

- Note that in each experiment, the challenger identifies the signer (or may identify that no such signer exist).
- In Direct Anonymous Attestation the challenger cannot identify the signer...
- In DAA challenger does not even know, whether the adversary broke unforgeability or seclusiveness.
Note that in each experiment, the challenger identifies the signer (or may identify that no such signer exist).

In Direct Anonymous Attestation the challenger cannot identify the signer...

In DAA challenger does not even know, whether the adversary broke unforgeability or seclusiveness.

In the security proofs for DAA, establishing the origin of the signature is done by an artificial procedure (e.g. knowledge extractor in ROM).
We may assign an index to every user in the system.
- We may assign an index to every user in the system.
- The adversary may ask for,
  - pseudonyms signatures and private keys of the \( i \)th user,
Unlinkability - Game Based Definition

- We may assign an index to every user in the system.
- The adversary may ask for,
  - pseudonyms signatures and private keys of the $i$th user,

If the adversary gives as input user indexes, he knows exactly which pseudonyms belong to which users.
We may assign an index to every user in the system. The adversary may ask for, pseudonyms signatures and private keys of the $i$th user,

If the adversary gives as input user indexes, he knows exactly which pseudonyms belong to which users.

Example

- Pseudonym of the $i$-th user in domain $dom_1 \rightarrow nym_1$
- Pseudonym of the $i$-th user in domain $dom_2 \rightarrow nym_2$
Game based definitions

- Bender, Dagdelen, Fischlin, Kügler: ISC 2012 [BDFK12]
  - a mistake, every adversary can win the game.
Game based definitions

- Bender, Dagdelen, Fischlin, Kügler: ISC 2012 [BDFK12]
  - a mistake, every adversary can win the game.

- Bringer, Chabanne, Lescuyer, Patey: Financial Cryptography 2014 [BCLP14]
  - attempt to cover the problem with “uncertainty sets”
  - obscure and hard to understand
  - restricts the adversary to some narrow strategies and does not cover some real world cases
Unlinkability - Previous work

Game based definitions

- Bender, Dagdelen, Fischlin, Kügler: ISC 2012 [BDFK12]
  - a mistake, every adversary can win the game.

- Bringer, Chabanne, Lescuyer, Patey: Financial Cryptography 2014 [BCLP14]
  - attempt to cover the problem with “uncertainty sets”
  - obscure and hard to understand
  - restricts the adversary to some narrow strategies and does not cover some real world cases

- Brickell, Chen, Li: International Journal of Information Security [BCL09]
  - considers just two users in one domain.
Change of concept for Defining Unlinkability

Ideal World

- sk_{1,1} → nym_1
- sk_{1,2} → nym_2
- sk_{2,1} → nym_1
- sk_{2,2} → nym_2

Domain 1

Domain 2
Change of concept for Defining Unlinkability

**Ideal World**

- $sk_{1,1} ightarrow nym_1$
- $sk_{1,2} ightarrow nym_2$
- $sk_{2,1} ightarrow nym_1$
- $sk_{2,2} ightarrow nym_2$

**Real World**

- $sk_1 ightarrow nym_1$
- $sk_1 ightarrow nym_2$
- $sk_2 ightarrow nym_1$
- $sk_2 ightarrow nym_2$
Defining unlinkability
long story of problems with a formal treatment

Two approaches

- Game Based definitions - huge problems for pseudonym unlinkability
- Simulation based approaches - static corruptions only
Defining unlinkability
long story of problems with a formal treatment

Two approaches

- Game Based definitions - huge problems for pseudonym unlinkability
- Simulation based approaches - static corruptions only

New approaches

- this work - game based definitions, except for anonymity which is simulation based:
  how much new knowledge for the adversary is brought by the particular crypto algorithm instead of independent keys for each domain
Defining unlinkability
long story of problems with a formal treatment

Two approaches

- Game Based definitions - huge problems for pseudonym unlinkability
- Simulation based approaches - static corruptions only

New approaches

- this work - game based definitions, except for anonymity which is simulation based:
  how much new knowledge for the adversary is brought by the particular crypto algorithm instead of independent keys for each domain
Designs related to Pseudonymous Signature

1 prototype of PS: [BDFK12] Bender, Dagdelen, Fischlin, Kügler: ISC 2012

- No seclusiveness. If the adversary gets two secret key, then he might compute the Issuer’s secret key.
Designs related to Pseudonymous Signature

1. **Prototype of PS**: [BDFK12] Bender, Dagdelen, Fischlin, Kügler: ISC 2012
   - No seclusiveness. If the adversary gets two secret keys, then he might compute the Issuer’s secret key.

2. **A solution from pairings but no group key problem**: [BCLP14] Bringer, Chabanne, Lescuyer, Patey: Financial Cryptography 2014
   - Minor problems (proofs do not work).
   - Pairing delegation procedure leaks partially the user’s secret key.
Designs related to Pseudonymous Signature

1. **prototype of PS:** [BDFK12] Bender, Dagdelen, Fischlin, Kügler: ISC 2012
   - No seclusiveness. If the adversary gets two secret key, then he might compute the Issuer’s secret key

2. **a solution from pairings but no group key problem:** [BCLP14] Bringer, Chabanne, Lescuyer, Patey: Financial Cryptography 2014
   - Minor problems (proofs do not work).
   - Pairing delegation procedure leaks partially the user’s secret key.

3. **solution from pairings, model issues fixed:** this work
Solution Overview

- Boneh-Boyen like signature based on user’s secret key:
  
  \[(u, x, A = (g \cdot h^x)^{1/(z+u)})\]
Ad hoc Domain Signatures

Solution Overview

- Boneh-Boyen like signature based on user’s secret key:
  \[ (u, x, A = (g \cdot h^x)^{1/(z+u)}) \]
- deriving a pseudonym of a user in a domain
  \[ nym = \text{Hash}(\text{domain-name})^u \cdot g^x \]
Solution Overview

- Boneh-Boyen like signature based on user’s secret key:
  \[
  (u, x, A = (g \cdot h^x)^{1/(z+u)})
  \]

- deriving a pseudonym of a user in a domain
  \[
  nym = \text{Hash} (\text{domain-name})^u \cdot g^x
  \]

- Signing via a Sigma Protocol and Fiat-Shamir transformation:
  \[
  ZKPoK \{ (\alpha, \beta, \gamma) : \\
  nym = H(\text{domain-name})^\alpha \cdot g^\beta \land \gamma^{z+\alpha} \cdot h^{-\beta} = g_1 \}
  \]
Efficiency comparison

### Signature Size

<table>
<thead>
<tr>
<th>Scheme</th>
<th>$G_1$</th>
<th>$G_2$</th>
<th>$G_T$</th>
<th>$Z_q$</th>
<th>Bit Size$^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our scheme</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>1792</td>
</tr>
<tr>
<td>[BDFK12]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>768</td>
</tr>
<tr>
<td>[BCLP14]</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>1792</td>
</tr>
</tbody>
</table>

### Signature Creation

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Multiplications</th>
<th>Exponentiations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Scheme</td>
<td>$3 \cdot G_1 + 2 \cdot G_T$</td>
<td>$6 \cdot G_1 + 3 \cdot G_T$</td>
</tr>
<tr>
<td>[BDFK12]</td>
<td>$1 \cdot G_1$</td>
<td>$3 \cdot G_1$</td>
</tr>
<tr>
<td>[BCLP14]</td>
<td>$4 \cdot G_1 + 2 \cdot G_T$</td>
<td>$6 \cdot G_1 + 3 \cdot G_T$</td>
</tr>
</tbody>
</table>

### Signature Verification

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Multiplications</th>
<th>Exponentiations</th>
<th>Inv.</th>
<th>Pairing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Scheme</td>
<td>$4 \cdot G_1 + 1 \cdot G_2 + 2 \cdot G_T$</td>
<td>$6 \cdot G_1 + 2 \cdot G_2 + 2 \cdot G_T$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>[BDFK12]</td>
<td>$1 \cdot G_1$</td>
<td>$3 \cdot G_1$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[BCLP14]</td>
<td>$4 \cdot G_1 + 2 \cdot G_T$</td>
<td>$6 \cdot G_1 + 3 \cdot G_T$</td>
<td>1</td>
<td>$G_T$</td>
</tr>
</tbody>
</table>

$^1$Counted according to RFC3766 for 256-bit representation $Z_p$, $G_1$ and 512-bit $G_2$. (3707-bit RSA modulus)
The current state-of-the-art: we may:
- request a signer to update his state (download new credentials/certificates), or
- use blacklists like in VRL Group Signatures.
Open Problems - Revocation

- The current state-of-the-art: we may:
  - request a signer to update his state (download new credentials/certificates), or
  - use blacklists like in VRL Group Signatures.

- If there are blacklists, then a the party which creates blacklists (issuer) may trace users.
The current state-of-the-art: we may:
- request a signer to update his state (download new credentials/certificates), or
- use blacklists like in VRL Group Signatures.

If there are blacklists, then a the party which creates blacklists (issuer) may trace users.

For Ad Hoc Domain Signatures: we may not be aware about every domain used, thus it is hard to blacklist.
We gave a new and presumably correct definition for Ad Hoc Domain Signatures. At least some issues from previous works are solved.
Conclusions

- We gave a new and presumably correct definition for Ad Hoc Domain Signatures. At least some issues from previous works are solved.
- It may prove useful for giving a sound definition for Direct Anonymous Attestation.
Conclusions

- We gave a new and presumably correct definition for Ad Hoc Domain Signatures. At least some issues from previous works are solved.
- It may prove useful for giving a sound definition for Direct Anonymous Attestation.
- We designed an “efficient” (?) scheme from Bilinear Maps.
Conclusions

- We gave a new and presumably correct definition for Ad Hoc Domain Signatures. At least some issues from previous works are solved.
- It may prove useful for giving a sound definition for Direct Anonymous Attestation.
- We designed an “efficient” (?) scheme from Bilinear Maps.
- Revocation may still be a problem.
Thank You