Anonymous Credentials on a Standard Java Card

Thomas Gross
Patrik Bichsel, Jan Camenisch, Victor Shoup
IBM’s BlueZ Group for Strong Authentication
Overview

- Introduction
- Camenisch-Lysyanskaya Signatures
- Problem Statement
- Key Ideas
- Results
Example: Age Proof with Strong Privacy

Identity Mixer Certificate
Address
DoB = 1980/12/01
Nr = 123456...

Citizen

Proof: “I’ve an EID card AND I’m older than 18.”

Authorities

offline

Service
Policy:
Have an EID card AND Be older than 18.
Java Card* Limitations

- 8-bit CPU (3.57 MHz)
- Limited access to public key-CP (only standard RSA, DSA)
- Limited RAM (2K)

*: JCOP 41/v2.2
Basis: Camenisch-Lysyanskaya Signatures

[Camenisch & Lysyanskaya ’01]

Public key of signer: RSA modulus $n$ and $a_i, b, d \in QR_n$.

Secret key: factors of $n$.

Signature of $L$ attributes $m_1, \ldots, m_L \in \{0,1\}^\ell : (c,e,s)$

For random prime $e > 2^\ell$ and integer $s \approx n$, compute $c$ such that

$$d = a_1^{m_1} \cdots a_L^{m_L} b^s c^e \mod n$$

Theorem: Signature scheme is secure against adaptively chosen message attacks under SRSA assumption.

[SRSA: Barić & Pfitzmann '97 and Fujisaki & Okamoto '97]
Basis: Camenisch-Lysyanskaya Signatures  

[Camensich & Lysyanskaya ’01]

Public key of signer: RSA modulus \(n\) and \(a, b, d \in QR_n\),

Secret key: factors of \(n\)

Signature of \(L\) attributes \(m_1, \ldots, m_L \in \{0,1\}^\ell : (c,e,s)\)

For random prime \(e > 2^\ell\) and integer \(s \approx n\), compute \(c\) such that

\[
d = a_1^{m_1} \cdot \ldots \cdot a_L^{m_L} \cdot b^s \cdot c^e \mod n
\]

**Theorem:** Signature scheme is secure against adaptively chosen message attacks under SRSA assumption.

[SRSA: Barić & Pfitzmann '97 and Fujisaki & Okamoto '97]
Signature of \( L \) attributes \( m_1, \ldots, m_L \in \{0,1\}^\ell : (c,e,s) \)

For random prime \( e > 2^\ell \) and integer \( s \approx n \), compute \( c \) such that

\[
d = a_1^{m_1} \cdot \ldots \cdot a_L^{m_L} \cdot b^s \cdot c^e \mod n
\]

Abstractly requires computation of:

\[
A_1^{x_1} \cdot \ldots \cdot A_i^{x_i} \cdot \ldots \cdot A_L^{x_L} \mod n
\]

where \( x_i \) correspond to attributes in the certificates and potentially \( |x_i| > |n| \)
Problem Statement

Run anonymous credential system autonomously and securely on a standard off-the-shelf Java Card.

Autonomy
All data on card
Malicious terminal

Security
CL-Signatures
Realistic keys

Efficiency
Proof in seconds

[Independent result: Sterckx, Gierlichs, Preneel, Verbauwhede '09]
[Balasch '02, Bichsel '07, Danes '07]
Java Card Structure

Source: Prof. Wolfgang Reif – chip cards
Java Card Structure

modExp() \rightarrow RSAEnc() ⊆ in EEPROM

modExp() \rightarrow adapt key in RAM 😊
RSAEnc()

IDMX Applet

Transient RSA

Basic Ops

RSA Enc() interface

Card Manager

Java Card

Card-Specific Operating System

8-bit CPU

3DES CP

Public Key CP

Source: Prof. Wolfgang Reif – chip cards
(Ab-)Using Standard RSA Interface

- Recall RSA Encryption: \( m^e \mod n \) (Limited size of \( e \))

- ModExp() with Big Exponents ➔ Split exponents:

\[
A_1^{x_1} A_2^{x_2} = A_1^{x_{11} + x_{12}2^k} A_2^{x_{21} + x_{22}2^k} \mod n
\]

\[
= A_1^{x_{11}} (A_1^{2k})^{x_{12}} A_2^{x_{21}} (A_2^{2k})^{x_{22}} \mod n
\]

\[
= A_1^{x_{11}} A'_1^{x_{12}} A_2^{x_{21}} A'_2^{x_{22}} \mod n
\]

- ModMultiply(): RSA interface can only do exponentiation ➔ Reduce multiply to modExp() by binomial formula:

\[
A \times B = ((A+B)^2 - A^2 - B^2)/2 \mod n
\]
Execution Times Full Proof (Including Communication)

<table>
<thead>
<tr>
<th>Modulus</th>
<th>1280 bit</th>
<th>1536 bit</th>
<th>1984 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precomputation</td>
<td>5203 ms</td>
<td>7828 ms</td>
<td>13250 ms</td>
</tr>
<tr>
<td>Compute A’</td>
<td>2125 ms</td>
<td>2906 ms</td>
<td>5000 ms</td>
</tr>
<tr>
<td>Compute T1</td>
<td>3078 ms</td>
<td>4922 ms</td>
<td>8250 ms</td>
</tr>
<tr>
<td>Policy-dependent</td>
<td>2234 ms</td>
<td>2625 ms</td>
<td>3298 ms</td>
</tr>
<tr>
<td>Compute 1 Response</td>
<td>562 ms</td>
<td>656 ms</td>
<td>828 ms</td>
</tr>
<tr>
<td>Total</td>
<td>7437 ms</td>
<td>10453 ms</td>
<td>16548 ms</td>
</tr>
</tbody>
</table>
Results

- **Anonymous credential system on standard Java Card**
  - JCOP 41/v2.2
  - Future: Java Card 3.0 standard

- **Attributes**: Focus on proof of possession
  - rely on hardware tamper resistance for statement, and
  - detect / revoke broken cards.

- **Autonomous**: secure in face of untrusted terminal

- **Efficient**: 10 sec (at 1536 bits)
  - 7.5 sec pre-computation / 2.5 sec on-line
I’m happy to answer questions…

Identity Mixer Community Site:

idemix.wordpress.com

- See what’s going on…
- Look at the spec…
- Download the library…
Detailed Performance Analysis: Modulus 1536 bit
Amortized Estimates over 1000 Ops, Upper Bound on Parameter Length, Percent Rounded Down

<table>
<thead>
<tr>
<th>Function</th>
<th>Time</th>
<th>Ops</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplication</td>
<td>4’653 ms</td>
<td>9 Ops</td>
<td>39 %</td>
</tr>
<tr>
<td>Addition</td>
<td>2988 ms</td>
<td>36 Ops</td>
<td>25 %</td>
</tr>
<tr>
<td>ModSquare</td>
<td>243 ms</td>
<td>27 Ops</td>
<td>2 %</td>
</tr>
<tr>
<td>ModExp</td>
<td>4’308 ms</td>
<td>10 Ops</td>
<td>36 %</td>
</tr>
<tr>
<td>SRNG</td>
<td>1’088 ms</td>
<td>16 Ops</td>
<td>9 %</td>
</tr>
<tr>
<td>TRNG</td>
<td>815 ms</td>
<td>1 Op</td>
<td>6 %</td>
</tr>
<tr>
<td>Addition</td>
<td>581 ms</td>
<td>7 Ops</td>
<td>4 %</td>
</tr>
<tr>
<td>Digest</td>
<td>220 ms</td>
<td>10 Ops</td>
<td>1 %</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>11’665 ms</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Recall: The Strong RSA Assumption

Flexible RSA Problem: *Given RSA modulus* \( n \) *and* \( z \in \text{QR}_n \) *find integers* \( e \) *and* \( u \) *such that*

\[
u^e = z \mod n
\]

(Recall: \( \text{QR}_n = \{x: \text{exist } y \text{ s.t. } y^2 = x \mod n\} \))

- Introduced by Barić & Pfitzmann '97 and Fujisaki & Okamoto '97
- Hard in generic algorithm model \[\text{[Damgård & Koprowski '01]}\]
Signature Scheme based on the SRSA I

[Anonymous Credentials on a Standard Java Card]

Public key of signer: RSA modulus \( n \) and \( a_i, b, d \in QR_n \)

Secret key: factors of \( n \)

To sign \( k \) messages \( m_1, \ldots, m_k \in \{0,1\}^\ell \):

- choose random prime \( e > 2^\ell \) and integer \( s \approx n \)

- compute \( c \) such that

\[
d = a_1^{m_1} \cdots a_k^{m_k} b^s c^e \mod n
\]

- signature is \((c,e,s)\)
Signature Scheme based on the SRSA II

A signature \((c, e, s)\) on messages \(m_1, \ldots, m_k\) is valid iff:

- \(m_1, \ldots, m_k \in \{0, 1\}^\ell\):
- \(e > 2^\ell\)
- \(d = a_1^{m_1} \cdots a_k^{m_k} b^s c^e \mod n\)

Theorem: Signature scheme is secure against adaptively chosen message attacks under SRSA assumption.
Proof of Knowledge of a Signature

Observe:

Let \( c' = c \cdot b^{s'} \mod n \) with random \( s' \)

then \( d = c'^e \cdot a_1^{m_1} \cdot \cdots \cdot a_k^{m_k} \cdot b^{s*} \mod n \), with \( s^* = s - es' \)

i.e., \( (c', e, s^*) \) is a also a valid signature!

Therefore, to prove knowledge of signature on some \( m \)

1. provide \( c' \)
2. \( PK\{(e, m_1, \ldots, m_k, s) : \quad d := c'^e \cdot a_1^{m_1} \cdot \cdots \cdot a_k^{m_k} \cdot b^s \)

\[ \land \quad m_i \in \{0,1\}^\ell \quad \land \quad e \in 2^{\ell+1} \pm \{0,1\}^\ell \]
Proof of Knowledge of a Signature

Using second Commitment

assume second group $n, a, b, n$

$2^{nd}$ commitment $C = a_1^{sk} b^{s^*}$

To prove knowledge of signature on some $m$
provide $c'$

$PK\{ (e, m_1, \ldots, m_k, s, s^* ) :$

$C = a_1^{m_1} b^{s^*} \land d := c'^e a_1^{m_1} \cdot \ldots \cdot a_k^{m_k} b^s \}$