Anonymous Credentials on Smart Cards

Pim Vullers MSc
p.vullers@cs.ru.nl

Institute for Computing and Information Sciences – Digital Security
Radboud University Nijmegen

21st June 2011
Background

- Master Pure maths, mathematical education
  University of Budapest, Hungary
- Master Mathematics for Industry, ICT, Cryptography and Coding Theory
  Eindhoven University of Technology, The Netherlands
- Business Experience: Online communication

Research (TNO ICT, Information Security group)

- Project: Mobiman
- Identity management for mobile devices
Gergely Alpár

Background

- Bachelor (Technical) Computer Science  
  Eindhoven University of Technology, The Netherlands
- Master Information Security Technology  
  Kerckhoffs Institute (Eindhoven University of Technology,  
  University of Twente and Radboud University Nijmegen)

Research (Radboud University Nijmegen, Digital Security group)

- Project: Applet-based e-Ticketing
- Privacy-preserving protocols
- Smart card implementations
• L. Batina, J.H. Hoepman, B. Jacobs, W. Mostowski and P. Vullers: Developing efficient blinded attribute certificates on smart cards via pairings CARDIS 2010


• P. Vullers: Privacy in Advanced Smart Card Applications - a Challenging Task e-Smart 2010

• W. Mostowski and P. Vullers: Efficient U-Prove Implementation for Anonymous Credentials on Smart Cards SecureComm 2011 (submitted)
Outline

Introduction

Self-blindable Certificates on Java Card

U-Prove on MULTOS

Idemix on MULTOS?
Self-blindable certificates (Verheul, Radboud University)

- Central point: Attribute certificate
  - Single attribute
  - Issuer’s signature
  - Prover’s public key
- Issuance
  - Issuer learns the public key
  - Strongly identifying
- Attribute proving
  - Fresh blinding of certificate and public key for each session
  - Untraceable

Performance
Keep smart card implementation in mind while designing.
Related Work on Smart Cards

Related work

- Bichsel et al. (IBM Research, 2009), ± 7.5 sec
  Camenisch & Lysyanskaya anonymous credential system

- Tews & Jacobs (RU Nijmegen, 2009), ± 5 sec
  Selective disclosure of Brands (U-Prove)

- Sterckx et al. (KU Leuven, 2009), ± 3 sec
  Direct anonymous attestation

Our results

- Batina et al. (RU Nijmegen, 2010), ± 1.5 sec
  Self-blindable certificates of Verheul

- Current (optimised) running time: ± 0.6 sec
Elliptic Curve Cryptography

- Point multiplication: $k \cdot P = Q$
  - repeated addition
- Easy computation:
  - double $2P' = R'$ and
  - add $P + Q = R$
- Hard problem: EC discrete log
  - given $P$ and $Q$, determine $k$

- Point multiplication is a one way function which can be used to build public key cryptosystems
- The public key is $Q$ and the private key is $k$ for a fixed point $P$
- Allows for key agreement (Diffie-Hellman), signatures (DSA), encryption (ElGamal), and more . . .
• A bilinear pairing is a map $e : G_1 \times G_2 \to G_T$ which is bilinear, that is, linear in both components:

$$e(P + P', Q) = e(P, Q) \cdot e(P', Q)$$

and

$$e(P, Q + Q') = e(P, Q) \cdot e(P, Q')$$

• As a result, $e(n \cdot P, m \cdot Q) = e(P, Q)^{nm}$

• For practical purposes, $e$ has to be computable in an efficient manner
Self-blindable Signatures

**Pairing-based signatures**

- Signature $S = s \cdot P$ over a point $P$ is multiplication by a private key $s$
- Check $e(P, s \cdot Q) \overset{?}{=} e(S, Q)$ to verify a signature $S$ over $P$ using the public key $s \cdot Q$

**Self-blinding**

- Card generates random blinding factor $b$
- Forms new pair
  
  $P_b = b \cdot P$
  
  $S_b = b \cdot S = b \cdot s \cdot P = s \cdot b \cdot P = s \cdot P_b$
- Check $e(P_b, s \cdot Q) \overset{?}{=} e(S_b, Q)$ to verify the blinded $P$ and $S$
Introduction
Self-blindable Certificates on Java Card
U-Prove on MULTOS
Idemix on MULTOS?

Sketch of the Protocol

Keys $K_{private}$ and $K_{public}$, attribute $A$, certificate $C$

Card

Show attribute

Blind ($\#$) $K_{public}$, $K_{private}$ and $C$

$K_{public}$, $\not{\#}$, $A$

Terminal

Verify $\not{\#}$ using $K_{certificate}$ and $K_{public}$

Generate a random number $N$

Sign $N$

Sign ($\ast$) $N$ using $K_{private}$

$\not{\ast}N$

Verify $\not{\ast}N$ using $K_{public}$
System Setup

- Public fixed point $Q$ and a finite set of attributes
- A secret key $s_a$ and public key $Q_a = s_a \cdot Q$ for each attribute $a$
- The associated pairs $(a, Q_a)$ are publicly known, and stored in all terminals together with the fixed point $Q$

- Card $c$ generates a key pair $k_c, P_c = k_c \cdot P$
- Private key $k_c$ is assumed to be stored in a protected manner
- Card $c$ receives an attribute together with a certificate $C_a = s_a \cdot P_c$ linking its public key $P_c$ to the attribute $a$
Protocol for Attribute-proving

\[ k_c, \quad P_c = k_c \cdot P, \quad a, \quad C_a = s_a \cdot P_c \]

Card

\[ P, \quad Q, \quad (a, Q_a) \]

Terminal

Random nonce \( n \)

\[ N = n \cdot P \]

Fresh blinding \( b \)

\[ b \cdot k_c \cdot N, \quad b \cdot P_c, \quad b \cdot C_a, \quad a \]

Verify \( n \cdot (b \cdot P_c) = b \cdot k_c \cdot N \)

Verify \( e(b \cdot P_c, Q_a) = e(b \cdot C_a, Q) \)
Java Card Applet

The platform

- Java Card: Java language with specialised (limited) API
- Support for ECC by the cryptographic coprocessor: primitives for EC Diffie-Hellman (DH), EC DSA and key generation

Implementation details

- *Abuse* EC key generation to generate blinding factor
- *Abuse* EC DH primitive for point multiplications
Terminal Application

Components

- Bouncy Castle Library with an extension for Pairings
- javax.smartcardio Smart Card IO Library

Implementation details

- Needs to cope with the shortcomings of the Java Card applet
- Point reconstruction: derive $y$ using $y^2 = x^3 + ax + b$
- Signature verification: just guess the sign
- Pairing verification: exploit bilinearity property:
  either $e(b \cdot P_c, Q_a) = e(b \cdot C_a, Q)$ or $e(b \cdot P_c, Q_a)e(b \cdot C_a, Q) = 1$
### Test Results

<table>
<thead>
<tr>
<th>key length (bits)</th>
<th>attribute &amp; signature (ms)</th>
<th>verification (ms)</th>
<th>protocol total (ms)</th>
<th>communication (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>192</td>
<td>787</td>
<td>116</td>
<td>904</td>
<td>155</td>
</tr>
<tr>
<td>160</td>
<td>645</td>
<td>102</td>
<td>747</td>
<td>135</td>
</tr>
<tr>
<td>128</td>
<td>535</td>
<td>82</td>
<td>617</td>
<td>115</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>key length (bits)</th>
<th>key generation (ms)</th>
<th>key agreement (ms)</th>
<th>processing overhead (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>192</td>
<td>379</td>
<td>98</td>
<td>114</td>
</tr>
<tr>
<td>160</td>
<td>307</td>
<td>78</td>
<td>104</td>
</tr>
<tr>
<td>128</td>
<td>242</td>
<td>62</td>
<td>107</td>
</tr>
</tbody>
</table>
Analysis

Achievements

- On-card time of below one second is possible
- Cryptographic coprocessor is used for all calculations
- Amount of communication is far less than RSA approaches: 155, 135 and 115 bytes for key lengths of 192, 160 and 128 bits

Issues

- Key generation on the card is time consuming
- The card only returns the x-coordinate of the blinded values point reconstruction (involving guessing) is required for EC DSA
- Not fast enough for actual use (in e-Ticketing)
- ECC support on the smart card is rather limited, sofar
Results

- Anonymous credentials on smart cards are becoming possible
- Our results are in line with previous work
- Major bottleneck is the limited access to the cryptographic coprocessor of the smart card

Challenges for future research

- The attribute may be used for tracing (privacy)
- This protocol proves only a single attribute (efficiency)
- Attributes do not have values (encoding)
- Revocation is not supported by the current protocol
Outline

Introduction

Self-blindable Certificates on Java Card

U-Prove on MULTOS

Idemix on MULTOS?
U-Prove (Brands, Microsoft)

- Main ingredient: U-Prove token
  - Multiple attributes
  - Token’s public key
  - Issuer’s signature
- Blind issuance
  - Issuer does not learn the public key, only the attribute values
  - Issuer unlinkability
- Selective disclosure
  - Prover can decide which (properties of) attributes to show
  - Data minimisation

Traceability

Public key and signature can be used as a pseudonym.
MULTOS Application

The platform

- MULTOS: Smart card platform which can be programmed in MEL (assembly), C or Java with a very low-level API
- Developer cards based on Infineon SLE66 hardware
  - Only 960 bytes of RAM
  - Modular exponentiations only up to 1024 bits (quite fast), up to 2048 bits (very slow) with exponent restrictions
  - SHA-1 (the only available hash) is slow

Implementation details

- Simple APDU API: one value a message
- Straightforward implementation of the protocols
Example of MULTOS C code

```c
void computeCommitmentA(void) {
    ModularExponentiation(QSIZE_BYTES, PSIZE_BYTES,
        w_i[0].number, p.number, h.number, t.number);
    for(int i = 0; i < MAX_ATTR; i++) {
        if(UD[i]) continue; // i is in D, not interested
        ModularExponentiation(QSIZE_BYTES, PSIZE_BYTES,
            w_i[i+1].number, p.number, g_i[i+1].number, temp_ram.vars.a.number);
        ModularMultiplication(PSIZE_BYTES,
            t.number, temp_ram.vars.a.number, p.number);
    }

    // t now contains $h^{w_0} \prod_{i \in U} g_i^{w_i} \mod p$
    int len = putNumberIntoArray(PSIZE_BYTES, t.number, temp_ram.array);

    // $a = H(t) \mod q$
    SHA1(len, a.number, temp_ram.array);
    ModularReduction(QSIZE_BYTES, QSIZE_BYTES, a.number, q.number);

    debugValue("a", a.number, QSIZE_BYTES);
}
```
U-Prove (Brands, Microsoft)

Results

• Previous Java Card implementation: Tews and Jacobs (2009)
  • 5 seconds (for 2 attributes), 8 seconds (for 4)
• Efficient MULTOS implementation: Mostowski and Vullers
  • 0.5 seconds (for 2), 0.8 seconds (for 5)
• Compatible with U-Prove SDK (only smart card limitations)
• It would be even faster on NXP SmartMX hardware

Issues

• The token serves as a pseudonym (multi-show linkability)
• Microsoft pursues a different smart card approach
• Advanced features (derived attributes) are costly
• Our MULTOS cards have little RAM and limited cryptography
Introduction

Self-blindable Certificates on Java Card

U-Prove on MULTOS

Idemix on MULTOS?
Idemix (Camenisch & Lysyanskaya, IBM Research Zürich)

Components

- Pseudonyms
- Camenisch-Lysyanskaya signatures
  - blind signature scheme
  - self-blindable signatures
- Zero-knowledge proofs

Features

- Both issuer and multi-show unlinkability
- Efficient attributes encoding

Complexity

The many zero-knowledge proofs make it hard to understand and lead to a high computational complexity.
Results

- Direct Anonymous Attestation
  - Commercial use of anonymous credentials
  - Anonymous authentication of a TPM
  - No attributes
- Java Card implementations (of DAA):
  - Bichsel et al. (2009): 7.5 seconds
  - Sterckx et al. (2009): 3 seconds

Issues

- Complexity (steep learning curve)
- Only smart card implementations for DAA
Our Plans with Idemix

Implementation on MULTOS

• Similar to U-Prove capabilities, that is:
  • issuing and selective disclosure
• No pseudonyms, five attributes
• Lengths based on modulus size of 1024 bits

...