Attributes on Smart Cards

Efficient Implementations of Self-blindable Credentials, U-Prove and Idemix

Pim Vullers  Wojciech Mostowski

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

ABC4Trust general meeting, Aarhus
7th November 2012
Motivation

Privacy issues

- Smart cards are “Big Brother’s little helper” (Stefan Brands)
- With OV-chipcard / Oyster / Charlie / . . . , you tell who you are when you get on a bus, metro, train, . . .
- Identity-based solutions violate their users’ privacy (and increase identity-fraud risk)

Attribute-based authorisation

- Only provide the information which the system needs
- Example for e-Ticketing (electronic train ticket):
  - card only says “I’m a first class year pass, valid in 2012”
- Example for e-Identity (electronic wietpas):
  - card only says “I’m a Dutch citizen, and my age is above 18”
Related Work on Smart Cards

- Danes (RU Groningen, 2007), ± 6 sec
  Idemix (estimated running time, trusted terminal)

- 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

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

- Bjones et al. (ABC4Trust, 2012), ± 0.4 sec
  ECC-based U-Prove (estimated running time)
Outline

Introduction

Self-blindable credentials

U-Prove

Idemix

Summary
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 generator 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
- Certificate verification: exploit bilinearity property:
  
  either $\text{either } e(b \cdot P_c, Q_a) = e(b \cdot C_a, Q)$ or $e(b \cdot P_c, Q_a)e(b \cdot C_a, Q) = 1$
## 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>
## 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 (strong) RSA or subgroup DL approaches:
  only 155, 135 and 115 bytes for key lengths of respectively 192, 160 and 128 bits, fitting in a single APDU

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
- Not fast enough for use in public transport sector
- ECC support on the smart card is rather limited, sofar
### Table: Comparison of Microsoft’s device-protected U-Prove token approach and our U-Prove token on a smart card approach.

<table>
<thead>
<tr>
<th></th>
<th>Microsoft’s approach</th>
<th>our approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>characteristics</td>
<td>add-on security measure</td>
<td>full protocol implementation</td>
</tr>
<tr>
<td>card stores</td>
<td>single device-protection attribute</td>
<td>all attributes, other token values</td>
</tr>
<tr>
<td>card computes</td>
<td>short zero-knowledge proof for the device-protection attribute</td>
<td>complete presentation proof</td>
</tr>
<tr>
<td>advantages</td>
<td>fast, lightweight, protect any number of dynamically issued tokens using pre-issued devices</td>
<td>independent use of the card, no need to trust the terminal</td>
</tr>
<tr>
<td>disadvantages</td>
<td>trusted terminal required</td>
<td>requires more card resources (?)</td>
</tr>
</tbody>
</table>
The MULTOS Smart Card Platform

Benefits:
- Low-level access to the cryptographic co-processor
- Proper cryptographic/mathematical API
- Very flexible memory management

Drawbacks
- Low-level language causes a steep learning curve.
  - Tool-support could help (auto-complete, API documentation)
- Limited amount of memory available
  - Reserved portions of RAM per application (only 960 bytes)
- Less wide-spread/well-known as Java Card
  - Hence less resources (examples etc.)
Design of the System

- CardService translates U-Prove commands and data types into APDU commands for smart card communication.
- MULTOS implementation, limited by smart card characteristics, is 90-95% compatible with U-Prove SDK.
Issuance Results

Figure: U-Prove token issuance times (●: computation, □: overhead).
Verification Results

Figure: Attribute verification times (gray: computation, white: overhead).
Analysis

- Efficient MULTOS implementation of the U-Prove technology
- Complete prover side of the protocols run on a smart card
- Anonymous credentials on smart cards are becoming possible
- Major improvement over Microsoft’s device-binding feature

Next steps:
- MULTOS implementation of IBM’s Identity Mixer
- Studying elliptic curve cryptography based variants
Outline

Introduction

Self-blindable credentials

U-Prove

Idemix

Summary
Design of the System

- CardService translates Idemix commands and data types into APDU commands for smart card communication.
- MULTOS implementation, limited by smart card characteristics, is 90-95% compatible with Idemix Library.

![Diagram showing the relationship between Idemix CardService, IBM's Idemix Library, and MULTOS card with Idemix application.](image-url)
### Issuance Results

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Computation</th>
<th>Overhead</th>
<th>Total running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 attributes</td>
<td>2455</td>
<td>546</td>
<td>3001</td>
</tr>
<tr>
<td>5 attributes</td>
<td>2522</td>
<td>737</td>
<td>3259</td>
</tr>
</tbody>
</table>

**Table:** Credential issuance times (in milliseconds).
Verification Results

(a) 2 stored attributes

(b) 5 stored attributes

Figure: Attribute verification times (■: computation, □: overhead).
Analysis

- *Efficient MULTOS implementation* of the Idemix technology
- *Multi-show unlinkability* of the credentials on the smart card

- Major improvement over IBM’s DAA implementation

Next steps:
- Further optimise current implementations
- Studying other technologies
- Practicing with other platforms

- Making anonymous credentials *usable*
Outline

Introduction

Self-blindable credentials

U-Prove

Idemix

Summary
Summary

Related work

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

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

Our results

- Hoepman et al. (RU Nijmegen, 2010), ± 0.6 sec
  Self-blindable certificates of Verheul (1 attribute)

- Mostowski and Vullers (RU, 2011), ± 0.5 - 0.9 sec
  U-Prove selective disclosure (2-5 attributes)

- Vullers and Alpár (RU, 2012), ± 1.5 - 2.0 sec
  Idemix selective disclosure (2-5 attributes)