Attribute-based Credentials on Smart Cards

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

Privacy & Identity Lab
Institute for Computing and Information Sciences – Digital Security
Radboud University Nijmegen

SaToSS Research Meeting
28th February 2012

Context

Smart cards are "Big Brother's little helper" (Stefan Brands)

e-Ticketing

- The OV-chipkaart stores your identity
- With an OV-chipkaart you tell . . .
  - when (date and time),
  - how (bus, train, metro, . . .), and
  - where (precisely at which stop)

  . . . you travel
- This data is stored for several years; Serious privacy concerns!

Detailed profiles

. . . can be composed by *both* legitimate *and* malicious parties.

Attribute-based Credentials

Possible solution

- Authorisation based on attributes rather than on identities
- Attribute-based credentials:
  card only says "I'm a first class year pass valid in 2011"
- *Subtle point*: an attribute may be non-identifying, but the digital signature might be used for tracing cards/individuals

Why attribute-based credentials?

- Identity-based solutions violate their users' privacy (and increase identity-fraud risk)
- Attribute-based credentials provide the same level of security
- Attributes provide all the system needs to know

Broader Context

e-Identity

- Electronic passport and identity cards
- Storing (sensitive) personal information: your identity
- Newest features:
  - e-Signature application
  - On-line authorisation
  - (Attributes)

Beyond the government...

Use of verified attributes by commercial parties can easily result in undesired traceability by both the government and third parties.

Privacy and Smart Cards

Protection against outsiders

- Random UID
- Reader authentication
- Secure messaging
- Problem: performance

Protection against insiders

- Harder problem
- Zero-knowledge proofs or blinding of identifiable information
- Practical implementations are rare
- Bad performance

Outline

Self-blindable certificates (Verheul, Radboud University)

- Main ingredient: 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.

Self-blindable certificates (Verheul, Radboud University)

Related work

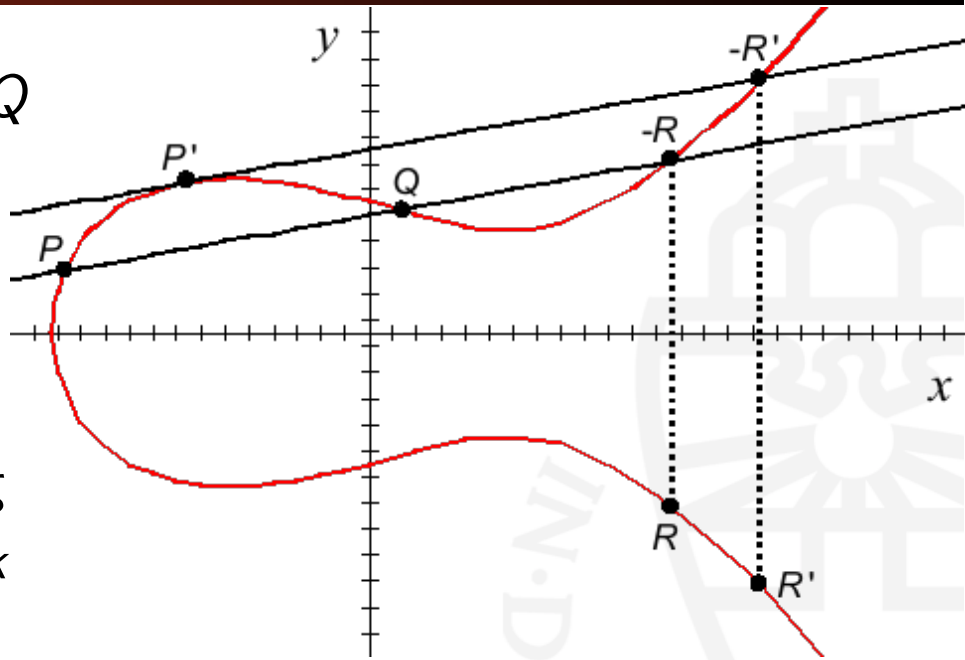
- Bichsel et al. (IBM Research, 2009), ± 7.5 sec
  Camenisch & Lysyanskaya anonymous credential system
- Tews & Jacobs (RU Nijmegen, 2009), ± 5 sec
  Selective disclosure of Brands (U-Prove)
- Sterckx et al. (KU Leuven, 2009), ± 3 sec
  Direct anonymous attestation

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 ...

Pairings

- A bilinear pairing is a map $e : G_1 \times G_2 \rightarrow G_T$ which is bilinear, that is, linear in both components:

$$e(P + P',\ Q) = e(P,\ Q) \cdot e(P',\ Q)$$
$$\text{and}$$
$$e(P,\ Q + Q') = e(P,\ Q) \cdot e(P,\ Q')$$

- As a result, $e(n \cdot P,\ m \cdot Q) = e(P,\ Q)^{n\,m}$
- For practical purposes, e has to be computable in an efficient manner

The decisional Diffie-Hellman problem

. . . is an example of a problem that is now easy to compute.

Self-blindable Signatures

Pairing-based signatures

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

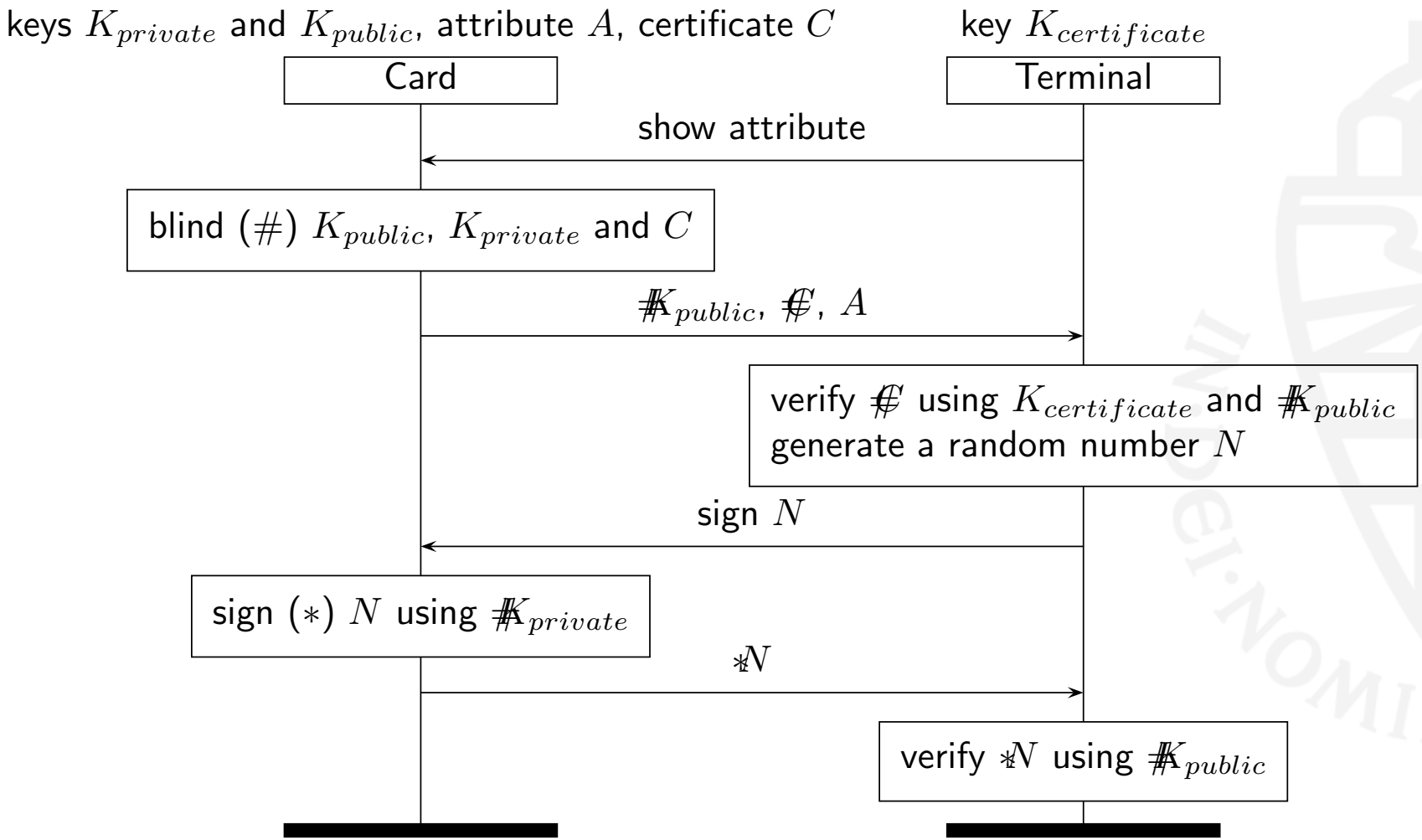
Self-blinding

- 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 $e(P_b, s \cdot Q) \stackrel{?}{=} e(S_b, Q)$ to verify the blinded P and S

Sketch of the Protocol

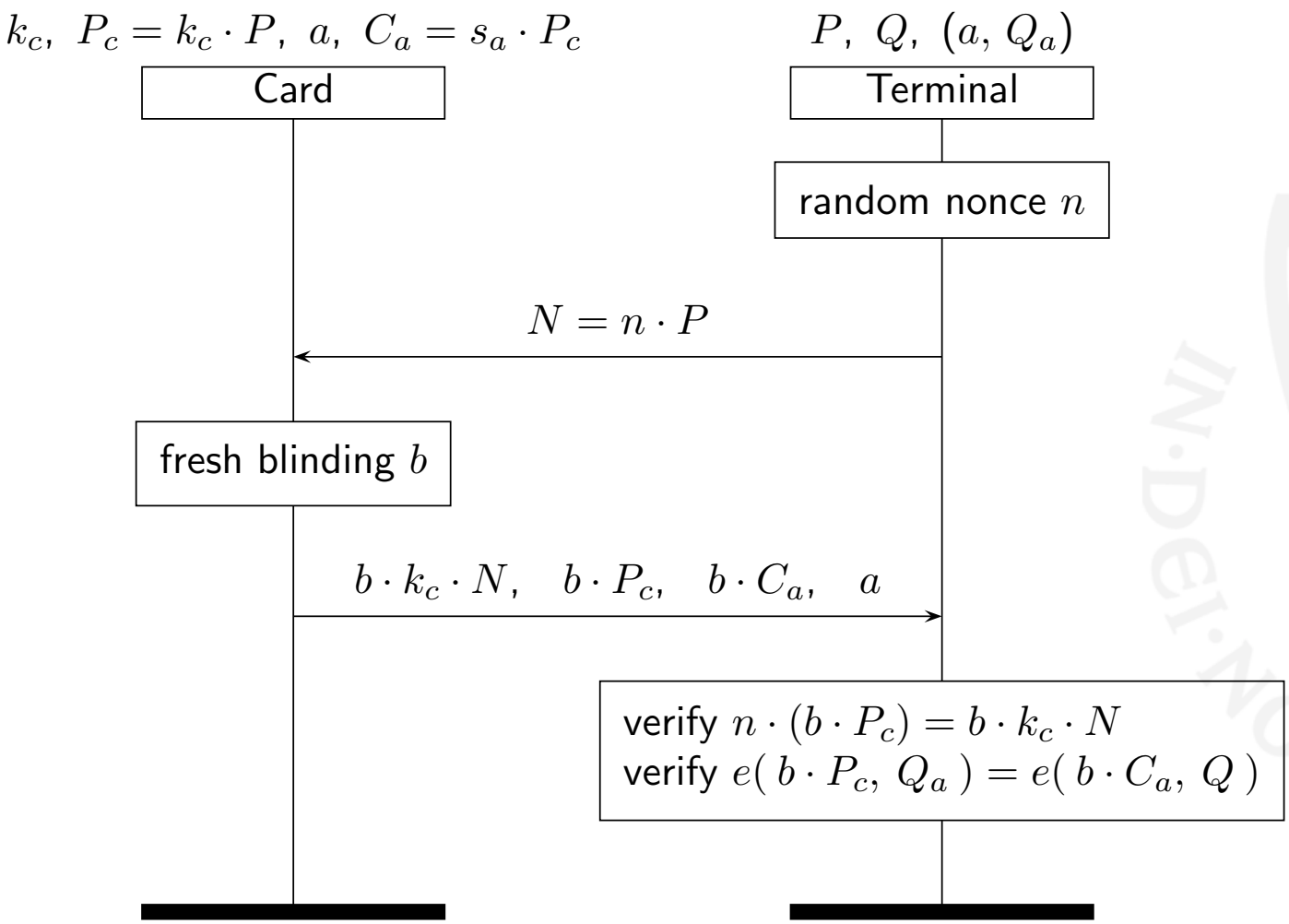


System Setup

- 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 know, 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



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

key length (bits)	attribute & signature (ms)	verification (ms)	protocol total (ms)	communication (bytes)
192	787	116	904	155
160	645	102	747	135
128	535	82	617	115

key length (bits)	key generation (ms)	key agreement (ms)	processing overhead (ms)
192	379	98	114
160	307	78	104
128	242	62	107

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
- Not fast enough for actual use (in e-Ticketing)
- ECC support on the smart card is rather limited, sofar

Self-blindable certificates (Verheul, Radboud University)

Issues

- This protocol proves only a single attribute (efficiency)
- Attributes do not have values
- Revocation is not supported by the current protocol
- Major bottleneck is the limited access to the cryptographic coprocessor of the Java Card smart card

Work in progress

- Support for revocation (two different approaches)
  - Using Boneh-Boyen signature scheme
  - Using Okamoto public key scheme
- Development of proper on-line demonstrator

Outline

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.

U-Prove (Brands, Microsoft)

Results

- Previous Java Card implementation: Tews and Jacobs (2009)
  - 5 seconds (for 2 attributes), 8 seconds (for 4)
- Efficient MULTOS impl.: Mostowski and Vullers (2011)
  - 0.5 seconds (for 2), 0.8 seconds (for 5)
- Compatible with U-Prove SDK (only smart card limitations)

U-Prove (Brands, Microsoft)

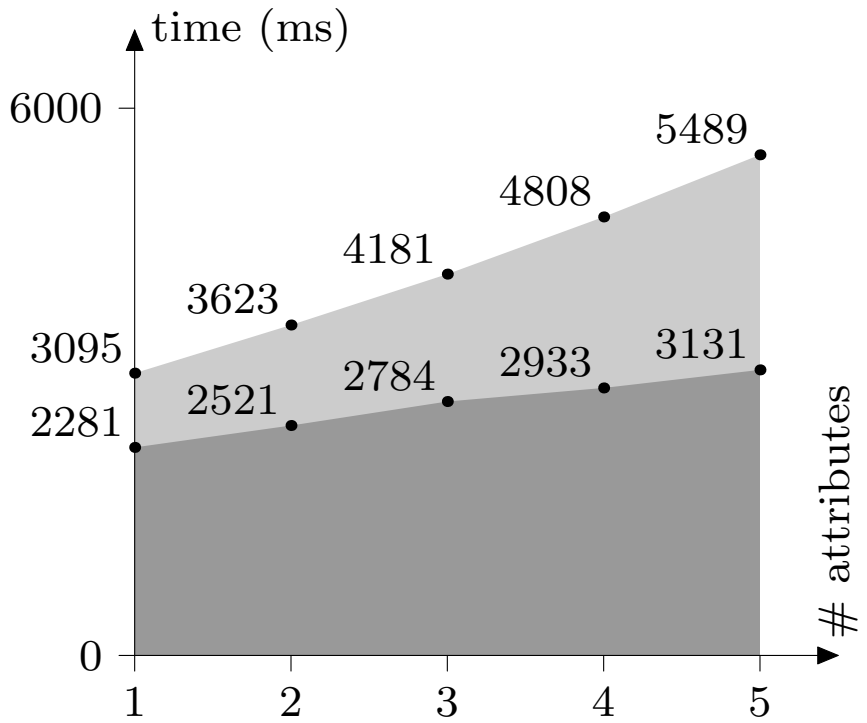
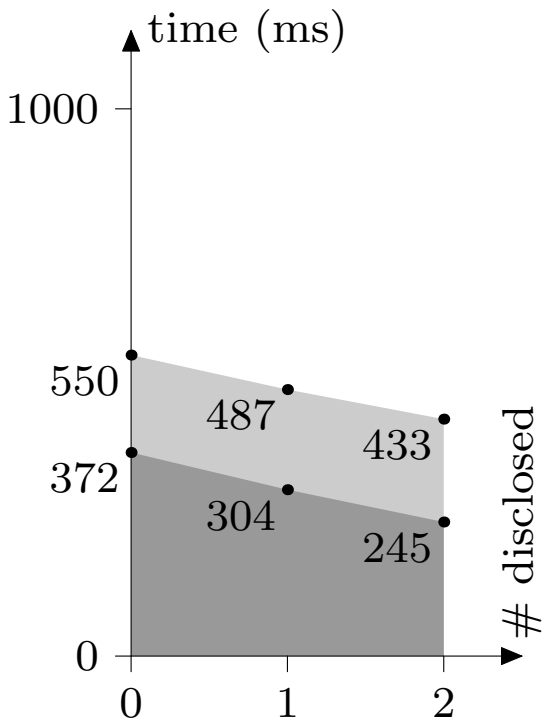
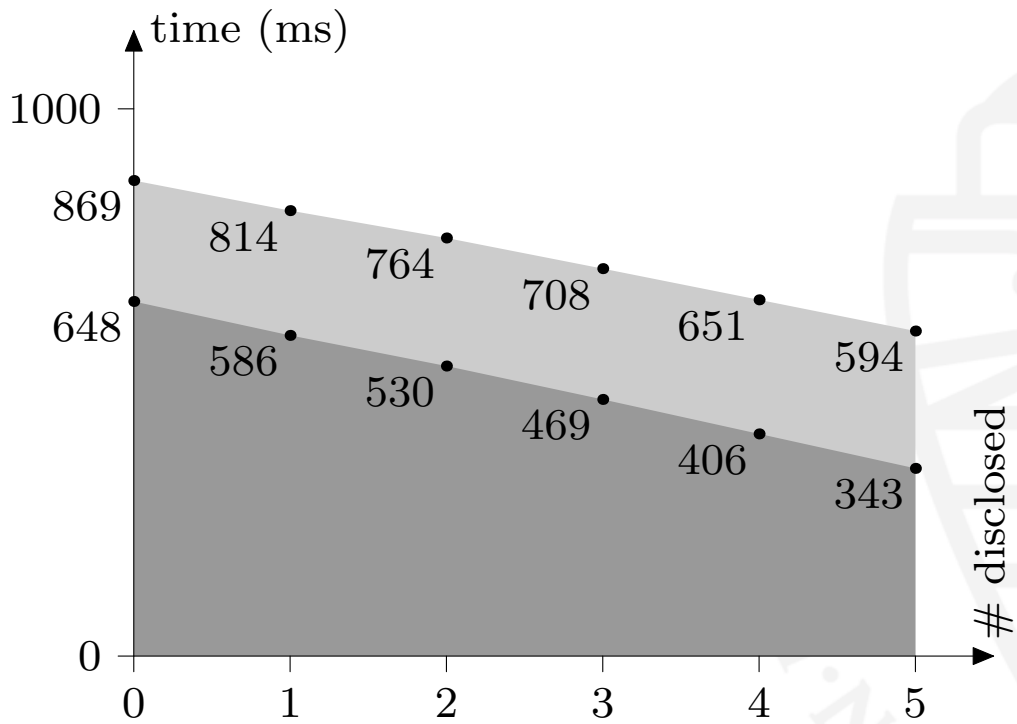


Figure: U-Prove token issuance times (: computation, : overhead).

U-Prove (Brands, Microsoft)



(a) 2 stored attributes



(b) 5 stored attributes

Figure: Attribute proving times (■: computation, ■: overhead).

U-Prove (Brands, Microsoft)

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

Work in progress

- Internship at Microsoft Research, eXtreme Computing Group
  - U-Prove on NFC-enabled devices
- ABC4Trust: Pilot with U-Prove on a smart card
  - Based on elliptic curve cryptography
  - Developed by CryptoExperts on a special card

Outline

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.

Idemix (Camenisch & Lysyanskaya, IBM Research Zürich)

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

Work in progress

- MULTOS smart card implementation
- Improvements of the specification
- Modifications of the Java library

Outline

Conclusion

- Anonymous credentials on smart cards are becoming possible
- Our results are in line with previous work
- Major bottleneck:
  - Java Card: limited access to the cryptographic coprocessor
  - MULTOS: little RAM and limited support for cryptography (no ECC or RSA larger than 1024 bits)

Challenges for future research

- Implementing Idemix on MULTOS
- Dealing with smart card platform shortcomings
- Adoption (ongoing project with Novay)