Privacy Enhancing Technologies
An Introduction

Jaap-Henk Hoepman
Digital Security (DS)
Radboud University Nijmegen, the Netherlands
@atoutw:// jhh@cs.ru.nl // www.cs.ru.nl/~jhh

Schedule

Agenda

Properties
- Anonymity etc
Strategies & patterns
PETs
- Blind signatures
- Group signatures
- Zero knowledge protocols
- Credentials/pseudonyms
- ... and some more...

Typical technical privacy properties

Anonymity (vs pseudonymity) / K-anonymity
- means that the subject is not identifiable within a set of subjects, the anonymity set

Unlinkability
- of two or more items of interest (IOIs, e.g., subjects, messages, actions, ...) means that the attacker cannot sufficiently distinguish whether these IOIs are related or not

Undetectability
- of an item of interest (IOI) means that the attacker cannot sufficiently distinguish whether it exists or not

Unobservability
- Equals Undetectability plus anonymity of subjects involved, even against other subjects

Software development cycle

Privacy enhancing technologies

Concept Development
Design
Implementation
Development
Levels of abstraction

- **Design strategy**
  - "A basic method to achieve a particular design goal" – that has certain properties that allow it to be distinguished from other basic design strategies

- **Design pattern**
  - "Commonly recurring structure to solve a general design problem within a particular context"

- **(Privacy enhancing) technology**
  - "A coherent set of ICT measures that protects privacy" – implemented using concrete technology

Data protection law

- **Core principles**
  - Data minimisation
  - Purpose limitation
  - Proportionality
  - Subsidiarity
  - Data subject rights: consent, (re)view
  - Adequate protection
  - (Provable) Compliance

6 privacy design strategies

- **Minimise**
  - The amount of PI should be minimal

- **Separate**
  - Process PI in a distributed fashion

- **Aggregate**
  - Process PI in the least possible detail

- **Hide**
  - PI should not be stored in plain view

- **Enforce**
  - A privacy policy should be in place and be enforced

- **Inform**
  - Subjects should be informed when PI is processed

- **Control**
  - Subjects should have control over when PI is processed

- **Demonstrate**
  - Compliance to policies and legal requirements must be demonstrated

What about design patterns?

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Patterns</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimise</td>
<td>Select before you collect, anonymisation, ...</td>
<td>green</td>
</tr>
<tr>
<td>Separate</td>
<td>Distribute, sector-specific pseudonyms</td>
<td>red</td>
</tr>
<tr>
<td>Aggregate</td>
<td>Data fuzzing, coarse-grained location</td>
<td>orange</td>
</tr>
<tr>
<td>Hide</td>
<td>Encryption, onion routing, ...</td>
<td>yellow</td>
</tr>
<tr>
<td>Enforce</td>
<td>Access control, privacy licenses</td>
<td>blue</td>
</tr>
<tr>
<td>Inform</td>
<td>PIP(1)</td>
<td>purple</td>
</tr>
<tr>
<td>Control</td>
<td>Informed consent(?)</td>
<td>green</td>
</tr>
<tr>
<td>Demonstrate</td>
<td>Privacy management system, logging</td>
<td>red</td>
</tr>
</tbody>
</table>

Basic techniques
Perfect forward security

- Time divided into epochs
- Each epoch users update their keys
  - Preferably without communicating with each other
- Suppose adversary compromises user at epoch \( j \)
  - Then he cannot recover the keys used at epoch \( i < j \), and hence not recover the messages exchanged in previous epochs

Example using symmetric keys

- \( k_i = H(k_{i-1}) \)
  - Where \( H \) is a Key Derivation Function (KDF)

Alternatively: session keys

- Established using Diffie-Hellman key exchange

Hiding metadata

- Mixnetworks
- Onion routing
- DC networks

Will be discussed in the Advanced Network Security course

(Partial) Homomorphic encryption

- A public key encryption protocol \( E_K : P \rightarrow C \)
  - From plaintext space \( P \), with group operation +
  - To ciphertext space \( C \), with group operation \( \times \)
- Such that
  - \( E(a+b) = E(a) \times E(b) \) and hence also \( c \times E(a) = E(ac) \)
- Example: Paillier (1999)

What can you do with homomorphic encryption

- Jointly compute the sum of private values, e.g. smart grid
- Electronic voting

Private information retrieval

- User wants to access record \( x_i \) from a remote database with \( n \) records, without revealing \( i \) to the database
- Obvious but bad solution
  - Send the whole database to the user; this costs \( n \) bits
- Q: can we do better?
  - A: yes, if we assume the database is split over \( k \) non-colluding servers
  - Also with 1 server, but then not information-theoretically secure

PIR (\( k=2 \))

- Two non-colluding servers with a copy of the database
- User generates a set \( J \) by randomly deciding for each \( j \) whether \( j \in J \) or not, with \( p = 0.5 \)
- User sends \( J \) to server 1 and \( J \cup i \) to server 2
  - I.e. sends two bit vectors of length \( n \)
- Each server computes \( r = \bigoplus_{j \in J} x_j \) for the set \( J \); it receives and returns \( r \) to the user
  - I.e. the number of bits in a single record
- The user receives \( r \) and computes \( x_i = r \oplus r' \)
- Bit complexity \( 2n \) plus the record size

Blind signatures & anonymous e-cash
Blind signatures: Digicash

Blind signature

Digicash offline coin

Coin deposit

Zero knowledge

The cave of Ali Baba

Zero-knowledge protocols
Proofs of knowledge

- **Completeness**
  - Verifier accepts the proof if the assertion is true
  - Assumption: the parties follow the protocol

- **Soundness**
  - If the fact is false, the verifier rejects the proof
  - Assumption: the parties follow the protocol

- **Zero knowledge**
  - No information about the prover's private input is revealed to the verifier
  - The verifier cannot convince a third party of the correctness of the assertion

Schnorr protocol

- **Cyclic group G with generator g**

Spot the Fed

Private handshaking

Private handshake
Requirements

Protocol

Attribute based credentials

Typical system

Typical system
How to implement a credential?

- Anonymous credentials: 3 techniques
  - Use a different one each time
    - UProve (Microsoft)
  - Prove knowledge of credential
    - Idemix (IBM)
  - Randomise credential each time
    - Self-blindable (RU)

Self blindable credentials

- Cryptographic implementation
  - Elliptic curve cryptography
    - Points \( P \) on a curve \( E \)
    - \( xP \): multiplying point \( P \) with scalar \( x \) in \( \mathbb{Z}_4 \)
    - CDH assumption: given \( xP \) and \( P \) it is hard to compute \( x \).
  - Pairing / bilinear map \( e \) :
    - \( e(P, Q) = e(P^q, Q^p) \)
    - DDH now is easy:
    - \( e(xP, yP) = e(P, zP) ? \)
  - Type 3: no computable homomorphism \( \phi : G_2 \rightarrow G_1 \)

- Public system parameters
  - \( P \in E(F_p) \), \( Q \in E(F_p^k) \), \( e \)

- Schnorr
  - Zero knowledge proof of DL of \( X = xP \)
    - Prover commits to \( R = rP \) for random \( r \in F_p \)
    - Verifier sends random challenge \( c \) \( < p \)
    - Prover sends \( s = r + xc \); Verifier checks \( R = sP - cX \)
  - Signature on message \( m \)
    - Signer commits to \( R = rP \) for random \( r \in F_p \)
    - Signer computes \( c = H(m) \) and \( s = r + xc \)
    - Signature \( (R, c, s) \)
    - To verify, take \( R \) and \( s \) to compute \( c \). Then check \( R = sP + cX \)

Keys and certificates

- Issuer
  - Private key \( a \)
  - Public key \( A = aQ \)
  - Signature \( C = ap \)

- Prover
  - Private key \( k \)
  - Public key \( K = kP \)
## Problems with basic protocol

- **Smart card must be highly secure**
  - Compromise of a single card (or private key $k_i$) allows one to create credentials for everyone
  
  $$C_j = k_j (C_i k_i)$$

- **No known way to revoke cards**
  - Published protocols are insecure because they make users traceable

## Questions

- twitter: @xotoxot
  - blog.xot.nl
  - jhh@cs.ru.nl
  - www.cs.ru.nl/~jhh