Privacy Enhancing Technologies
An Introduction

Jaap-Henk Hoepman

Digital Security (DS)
Radboud University Nijmegen, the Netherlands

@xotoxot // jhh@cs.ru.nl // www.cs.ru.nl/~jhh

Privacy Enhancing Technologies
An Introduction

Jaap-Henk Hoepman // Radboud University Nijmegen //

Schedule

<table>
<thead>
<tr>
<th>Date</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feb 25</td>
<td>Introduction to PETs and &quot;New frontiers&quot;</td>
</tr>
<tr>
<td>Feb 26</td>
<td>Lecture 1: Anonymous credentials</td>
</tr>
<tr>
<td>Feb 27</td>
<td>Lecture 2: Group signatures</td>
</tr>
<tr>
<td>Feb 28</td>
<td>Lecture 3: Zero knowledge protocols</td>
</tr>
<tr>
<td>Mar 07</td>
<td>Lecture 4: Credentials/pseudonyms</td>
</tr>
<tr>
<td>Mar 08</td>
<td>Lecture 5: Blind signatures</td>
</tr>
<tr>
<td>Mar 09</td>
<td>Lecture 6: Group signatures</td>
</tr>
<tr>
<td>Mar 10</td>
<td>Lecture 7: Zero knowledge protocols</td>
</tr>
<tr>
<td>Mar 11</td>
<td>Lecture 8: Credentials/pseudonyms</td>
</tr>
<tr>
<td>Mar 12</td>
<td>Lecture 9: Blind signatures</td>
</tr>
</tbody>
</table>

Agenda

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

16-02-17
Privacy from a technical perspective

Privacy goals
- Unlinkability
- Transparency
- Intervenability

As well as
- Confidentiality
- Integrity
- Availability

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

Privacy design strategies
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, review
  - Adequate protection
  - (Provable) Compliance
8 privacy design strategies

- **Minimise**
  - The amount of PII should be minimal
  - Process PII in the least possible detail
  - PII should not be stored in plain view

- **Separate**
  - Process PII in a distributed fashion
  - Process PII in a sector-specific pseudonym
  - Process PII in a sector-specific anonymisation

- **Aggregate**
  - Process PII in coarse-grained locations
  - Process PII in coarse-grained locations
  - Process PII in coarse-grained locations

- **Hide**
  - Encryption, onion routing, …

- **Enforce**
  - Access control, privacy licenses
  - P3P (XML)

- **Inform**
  - Informed consent (?)

- **Control**
  - Informed consent (?)

- **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 youcollect, anonymisation</td>
<td>Green</td>
</tr>
</tbody>
</table>
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 $i$
  - Then he cannot recover the keys used at past epoch $i < j$, and hence not recover the messages exchanged in previous epochs

Example using symmetric keys

- $k_{i,j} = \text{KDF}(k_i)$
  - Where $\text{KDF}$ is a key Derivation Function (KDF)

Alternatively: session keys

- Established using Diffie-Hellman key exchange

Future secrecy

- 'Self-healing' property
- Suppose adversary compromised user at some epoch (or recovered keys used in this epoch), but user recovers at epoch $j$
  - i.e. Adversary no longer controls user at epoch $i$
- Then adversary
  - cannot recover the keys used at future epoch $j > i$, and hence not recover the messages exchanged in future epochs

How to implement this: use OTR

- OTR advertises next key to use in a message, and sender will use this key as soon as recipient acknowledges this key
Hiding metadata

- Mixnetworks
- Onion routing
- DC networks

Will be discussed in the Advanced Network Security course

(Partial) Homomorphic encryption

- A public key encryption protocol \( E_k : P \rightarrow \mathbb{C} \)
  - From plaintext space \( P \) with group operation \( + \)
  - To ciphertext space \( \mathbb{C} \) with group operation \( \times \)
- Such that
  - \( E_k(a + b) = E_k(a) \times E_k(b) \) and hence also \( c \times E_k(a) = E_k(ca) \)
- 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 \( r_i \) 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 \( j \in E \) or not, with \( p = 0.5 \)
- User sends \( E \) to server 1 and \( J \cup E \) to server 2
  - I.e. sends two bit vectors of length \( n \)
- Each server computes \( r = \bigoplus_{j \in E} s_j \) for the set it receives and returns \( r \) to the user
  - I.e. the number of bits in a single record
- The user receives \( r, r' \) and computes \( r = r \oplus r' \)
- Bit complexity \( 2n \) plus the record size

Blind signatures & anonymous e-cash

Blind signatures: Digicash
Blind signature

\[ \begin{align*}
\text{Alice} & \quad \text{Bob}
\end{align*} \]

\[ r \in \mathbb{Z}_n^* \quad n \quad k \]

\[ n \rightarrow (n\cdot k)^r \rightarrow \text{sign with } k \]

\[ (n\cdot k)^r \rightarrow \text{divide by } r \]

\[ k \]

DigiCash offline coin

Coin deposit
Zero-knowledge protocols

Zero knowledge
- The cave of Ali Baba

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 $g$

Prover

\[ a \xrightarrow{(r, a^r)} \frac{a^r}{g^r} \xrightarrow{c} \frac{c}{g^r} \]

Verif\(\)

\[ 1 \xrightarrow{(r, a^r)} \frac{a^r}{g^r} \xrightarrow{c} \frac{c}{g^r} \]

- Completeness?
- Soundness?
- Zero knowledge?

Private handshaking
Spot the Fed

Private handshake

Requirements

security: Adversary A is unable to distinguish a protocol run involving nodes 1 and 2 from a protocol run involving nodes 1 and 3, even though messages exchanged between 1 and 2 are indistinguishable to the adversary.

forward secrecy: After the run, nodes 1 and 2 cannot retrieve another node's session key except with the adversary, even if messages exchanged between 1 and 2 are indistinguishable.
Protocol

Alice
- generate $a$
- group secret $g(a_1, ..., a_n)$ (randomly permuted)

Bob
- generate $a$
- group secret $g(a_1, ..., a_n)$ (randomly permuted)

1. Exchange
- pick random $x$
- send $x^a$ to $a$ (to Bob)
- send $y$ to $a$ (to Alice)

2. Involve $w$
- $k := 1 (n^2)$
- $k := k^a$ (Bob)

Group membership
- $g(a_1^w, a_2^w, ..., a_n^w)$
- involve same $w$

Invoke into $\pi_B^w$:
- $\pi_B = \pi_B^w | (g(a_1^w, a_2^w, ..., a_n^w))$
- $\pi_B = k^a | (k^a)^2 | (k^a)^3 | (< H_k)$

Attribute based credentials

IRMA
Credential Issuers

User

Relying Parties

Token Provider

Unforgeability

Non transferability

Unlinkability

Revocatability

New Authority

How to implement a credential?

Anonymous credentials: 3 techniques

- Use a different one each time
  - Eprove (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 $x \in \mathbb{Z}_p$
  - CDH assumption: given $xP$ and $P$, it is hard to compute $x$. (Note: division by $x$ is easy).

- Pairing / bilinear map $e : \hat{G}_1 \times \hat{G}_2 \rightarrow \hat{G}_T$
  - $e(P, yQ) = e(P, y)^{Qe(x^yP)}$
  - DDH is now easy: $e(P, yP) = e(P, y)^{P} = (0, a)$
  - Type 3: no computably homomorphism $\phi : \hat{G}_1 \rightarrow \hat{G}_2$

- Public system parameters
  - $P \in \hat{G}_1^*$, $Q \in \hat{G}_2^*$, $x$
Schnorr

- Zero knowledge proof of DL of \( x = \alpha^p \)
  - Prover commits to \( r = r' \) for random \( r \in \mathbb{F}_p \)
  - Verifier sends random challenge \( e < p \)
  - Prover sends \( s = r + xe \); Verifier checks \( x = \alpha^s - e^r \)

Signature on message \( m \)

- Signer commits to \( s = r + x_e \) for random \( r \in \mathbb{F}_p \)
- Signer computes \( c = h(m) \) and \( s = r + x_e \)
- Signature \( \sigma = (r, c, e) \)
- To verify, take \( r \) and \( m \) to compute \( c \). Then check \( x = \alpha^x - e^r \)

Keys and certificates

Issuer  Prover

<table>
<thead>
<tr>
<th>Private key</th>
<th>( x )</th>
<th>( s )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public key</td>
<td>( A = x^q )</td>
<td>( A = x^q )</td>
</tr>
<tr>
<td>Certificate</td>
<td>( c \in \mathbb{F}_q )</td>
<td>( c \in \mathbb{F}_q )</td>
</tr>
</tbody>
</table>

Basic protocol

\[
\begin{align*}
\text{Prover} & \quad \text{Verifier} \\
\text{blind} \quad \alpha & \in \mathbb{Z}_p \\
\text{send} \quad \alpha N, \alpha C & \quad \text{into} \quad \overline{K}, \overline{C} \\
\quad & \text{verify} \quad \epsilon(N, A) = \epsilon(C, Q) \\
& \text{nonce} \quad e \in \mathbb{Z}_p \\
& \quad \text{into} \quad N \\
& \text{send} \quad \overline{M} \\
& \quad \text{verify} \quad \overline{M} = eK
\end{align*}
\]
Basic protocol

\[
\text{Prover} \quad \epsilon(\alpha k, \alpha x) \quad \epsilon(x, \tilde{x})
\]
\[
\text{Verifier} \quad \epsilon(x, \tilde{x})
\]

1. Blind \( \alpha \in \mathbb{Z}_p \)
   - send \( \alpha K, \alpha C \) into \( \tilde{K}, \tilde{C} \)
   - verify \( \epsilon(K, A) = \epsilon(C, Q) \)

2. Send \( \alpha k, N \) into \( \tilde{M} \)
   - verify \( \tilde{M} = qK \)

Problems with basic protocol

- Smart card must be highly secure
  - Compromise of a single card (or private key \( k_0 \)) allows one to create credentials for everyone
  \[\gamma = \frac{k_0}{\mathbb{Z}_p}\]
- No known way to revoke cards
  - Published protocols are insecure because they make users traceable

Questions

?