Selective Disclosure for Identity Management
A critique of identity

- Identity as a proxy to check credentials
  - Username decides access in Access Control Matrix

- Sometimes this leaks too much information

- Real world examples
  - Tickets allow you to use cinema / train
  - Bars require customers to be older than 18
    - But do you want the barman to know your address?
The privacy-invasive way

- **Usual way:**
  - **Identity provider** certifies attributes of a **subject**.
  - **Relying Party** checks those attributes
  - Match credential with **live person** (biometric)

- **Examples:**
  - E-passport: signed attributes, with lightweight access control.
    - Attributes: nationality, names, number, pictures, ...
  - Identity Cards: signatures over attributes
    - Attributes: names, date of birth, picture, address, ...
Selective Disclosure Credentials

- The players:
  - Issuer (I) = Identity provider
  - Prover (P) = Subject
  - Verifier (V) = Relying party

- Properties:
  - The prover convinces the verifier that he holds a credential with attributes that satisfy some boolean formula:
    - Simple example “age=18 AND city=Cambridge”
  - Prover cannot lie
  - Verifier cannot infer anything else aside the formula
  - Anonymity maintained despite collusion of V & I
1. Issuing protocol:
   Prover gets a certified credential.

2. Showing Protocol:
   Prover makes assertions about some attributes

Issuer
Name=Peggy, age=25, address=Cambridge, Status=single

Prover Peggy

Passport Issuing Authority

Verifier Victor
(Bar staff Checking age)

Can't learn anything beyond age

age=25
Two flavours of credentials

- Single-show credential (Brands & Chaum)
  - Blind the issuing protocol
  - Show the credential in clear
  - Multiple shows are linkable – BAD

- Multi-show (Camenisch & Lysyanskaya)
  - Random oracle free signatures for issuing (CL)
  - Blinded showing
    - Prover shows that they know a signature over a particular ciphertext.
  - Cannot link multiple shows of the credential
  - More complex – BAD

We will Focus on these
Cryptographic preliminaries
- The discrete logarithm problem
- Schnorr’s Identification protocol
  - Unforgeability, simulator, Fiat-Shamir Heuristic
  - Generalization to representation

Showing protocol
- Linear relations of attributes
- AND-connective

Issuing protocol
- Unlikable issuing
- Efficient proof of a signature.

What is a Zero-Knowledge Proof?
Assume $p$ a large prime
- ($>1024$ bits—$2048$ bits)
- Detail: $p = qr+1$ where $q$ also large prime
- Denote the field of integers modulo $p$ as $\mathbb{Z}_p$

Example with $p=5$
- Addition works fine: $1+2 = 3$, $3+3 = 1$, ...
- Multiplication too: $2*2 = 4$, $2*3 = 1$, ...
- Exponentiation is as expected: $2^2 = 4$

Choose $g$ in the multiplicative group of $\mathbb{Z}_p$
- Such that $g$ is a generator
- Example: $g=2$
Exponentiation is computationally easy:
- Given $g$ and $x$, easy to compute $g^x$

But logarithm is computationally hard:
- Given $g$ and $g^x$, difficult to find $x = \log_g g^x$
- If $p$ is large it is practically impossible

Related DH problem
- Given $(g, g^x, g^y)$ difficult to find $g^{xy}$
- Stronger assumption than DL problem
More on $\mathbb{Z}_p$

- **Efficient to find inverses**
  - Given $c$ easy to calculate $g^{-c} \mod p$
    - $(p-1) - c \mod p-1$

- **Efficient to find roots**
  - Given $c$ easy to find $g^{1/c} \mod p$
    - $c (1/c) = 1 \mod (p-1)$
  - Note the case $N=pq$ (RSA security)

- No need to be scared of this field.
Exemplary of the zero-knowledge protocols credentials are based on.

Players
- Public – $g$ a generator of $\mathbb{Z}_p$
- Prover – knows $x$ (secret key)
- Verifier – knows $y = g^x$ (public key)

Aim: the prover convinces the verifier that she knows an $x$ such that $g^x = y$
- Zero-knowledge – verifier does not learn $x$!

Why identification?
- Given a certificate containing $y$
Schnorr’s protocol

Knows: x

Public: g, p

Knows: y = g^x

Random: w

\[
\begin{align*}
P \rightarrow V &: g^w = a \\
V \rightarrow P &: c \\
P \rightarrow V &: cx + w = r
\end{align*}
\]

(witness)  (challenge)  (response)

Check:
\[
g^r = y^c a \\
g^{cx + w} = (g^x)^c g^w
\]
Assume that Peggy (Prover) does not know $x$?

- If, for the same witness, Peggy forges two valid responses to two of Victor’s challenges
  \[
  r_1 = c_1 x + w \\
  r_2 = c_2 x + w
  \]

- Then Peggy must know $x$
  \[
  \text{2 equations, 2 unknowns (x,w) – can find } x
  \]
Zero-knowledge (intuition)

- The verifier learns nothing new about \( x \).
- How do we go about proving this?
  - Verifier can simulate protocol executions
    - On his own!
    - Without any help from Peggy (Prover)
  - This means that the transcript gives no information about \( x \)
- How does Victor simulate a transcript?
  - (Witness, challenge, response)
Simulator

Need to fake a transcript \((g^{w'}, c', r')\)

Simulator:

- Trick: do not follow the protocol order!
- First pick the challenge \(c'\)
- Then pick a random response \(r'\)
  - Then note that the response must satisfy:
    \[ g^{r'} = (g^x)^{c'} g^{w'} \rightarrow g^{w'} = g^{r'} / (g^x)^{c'} \]
- Solve for \(g^{w'}\)

Proof technique for ZK
- but also important in constructions (OR)
Schnorr’s protocol

- Requires interaction between Peggy and Victor
- Victor cannot transfer proof to convince Charlie
  - (In fact we saw he can completely fake a transcript)

Fiat-Shamir Heuristic

- $H[\cdot]$ is a cryptographic hash function
- Peggy sets $c = H[g^w]$
- Note that the simulator cannot work any more
  - $g^w$ has to be set first to derive $c$

Signature scheme

- Peggy sets $c = H[g^w, M]$
Generalise to DL represenations

- Traditional Schnorr
  - For fixed $g, p$ and public key $h = g^x$
  - Peggy proves she knows $x$ such that $h = g^x$

- General problem
  - Fix prime $p$, generators $g_1, \ldots, g_l$
  - Public key $h' = g_1^{x_1}g_2^{x_2} \ldots g_l^{x_l}$
  - Peggy proves she knows $x_1, \ldots, x_l$ such that $h' = g_1^{x_1}g_2^{x_2} \ldots g_l^{x_l}$
DL representation – protocol

Public: $g, p$

Knows: $x_1, ..., x_l$

Knows: $h = g_1^{x_1} g_2^{x_2} ... g_l^{x_l}$

Peggy
(Prover)

Victor
(Verifier)

I random: $w_i$

$r_i = cX_i + w_i$

P->V: $\prod_{0<i<l} g^{w_i} = a$ (witness)

V->P: $c$ (challenge)

P->V: $r_1, ..., r_l$ (response)

Check:

$\left(\prod_{0<i<l} g_i^{r_i}\right) = h^c a$

Let’s convince ourselves: $\left(\prod_{0<i<l} g_i^{r_i}\right) = \left(\prod_{0<i<l} g_i^{x_i}\right)^c \left(\prod_{0<i<l} g^{w_i}\right) = h^c a$
**DL representation vs. Schnorr**

**Public:** $g, p$

**Knows:** $x_1, \ldots, x_l$

**Knows:** $h = g_1^{x_1} g_2^{x_2} \ldots g_l^{x_l}$

**Peggy (Prover):**

- $r_i = cX_i + w_i$

**Victor (Verifier):**

- I random: $w_i$

**P -> V:** $\prod_{0<i<l} g^{w_i} = a$ (witness)

**V -> P:** $c$ (challenge)

**P -> V:** $r_1, \ldots, r_l$ (response)

**Check:**

$\left( \prod_{0<i<l} g_i^{r_i} \right) = h^c a$

Lets convince ourselves:

$\left( \prod_{0<i<l} g_i^{r_i} \right) = \left( \prod_{0<i<l} g_i^{x_i} \right)^c \left( \prod_{0<i<l} g^{w_i} \right) = h^c a$
Credentials – showing

- Relation to DL representation

- Credential representation:
  - Attributes $x_i$
  - Credential $h = g_1^{x_1} g_2^{x_2} \ldots g_l^{x_l}, \text{Sig}_{\text{Issuer}}(h)$

- Credential showing protocol
  - Peggy gives the credential to Victor $(h, \text{Sig}_{\text{Issuer}}(h))$
  - Discloses only some attributes
  - Peggy proves a statement on values $x_i$
    - $X_{\text{age}} = 28 \text{ AND } x_{\text{city}} = H[\text{Cambridge}]$
How?

- It always reduces to proving knowledge of a DL representation.
  - But which one?

- To simply disclose attributes
  - Cancel them out of the credential
  - For $X_{\text{age}} = 28$ AND $x_{\text{city}} = H[\text{Cambridge}]

- Proves she know the DL representation of

\[ h/(g_{\text{age}})^{X_{\text{age}}}(g_{\text{city}})^{X_{\text{city}}} = h' = \prod_{3<i<l} g^{x_i} \]

(Also do not forget to check the signature!)
Linear relations of attributes (1)

- Remember:
  - Attributes $x_i$, $i = 1, \ldots, 4$
  - Credential $h = g_1^{x_1} g_2^{x_2} g_3^{x_3} g_4^{x_4}$, $\text{Sig}_{\text{Issuer}}(h)$

- Example relation of attributes:
  - $(x_1 + 2x_2 - 10x_3 = 13)$ AND $(x_2 - 4x_3 = 5)$
  - Implies: $(x_1 = 2x_3 + 3)$ AND $(x_2 = 4x_3 + 5)$
  - Substitute into $h$
    - $h = g_1^{2x_3+3} g_2^{4x_3+5} g_3^{x_3} g_4^{x_4} = (g_1^3 g_2^5)(g_1^2 g_2^4 g_3)^{x_3} g_4^{x_4}$
    - Implies: $h / (g_1^3 g_2^5) = (g_1^2 g_2^4 g_3)^{x_3} g_4^{x_4}$
Example (continued)

- \((x_1 + 2x_2 - 10x_3 = 13)\) AND \((x_2 - 4x_3 = 5)\)
- Implies: \(h / (g_1^3g_2^5) = (g_1^2g_2^4g_3)^{x_3} g_4^{x_4}\)

How do we prove that in ZK?

- DL representation proof!
  - \(h' = h / (g_1^3g_2^5)\)
  - \(g_1' = g_1^2g_2^4g_3\) \(g_2' = g_4\)
  - Prove that you know \(x_3\) and \(x_4\) such that \(h' = (g_1')^{x_3} (g_2')^{x_4}\)
Peggy (Prover) Knows: $x_1, x_2, x_3, x_4$

Victor (Verifier) Knows: $h = g_1^{x_1} g_2^{x_2} g_3^{x_3} g_4^{x_4}$

Public: $g, p$

random: $w_1, w_2$

$P \rightarrow V$: $g_1^{w_1} g_2^{w_2} = a'$ (witness)

$V \rightarrow P$: $c$ (challenge)

$P \rightarrow V$: $r_1, r_2$ (response)

Check:

$\left( g_1' \right)^{r_1} \left( g_2' \right)^{r_2} = (h')^c a$
Check \((g_1')^{r_1} (g_2')^{r_2} = (h')^c a\)

- **Reminder**
  - \(h = g_1^{x_1} g_2^{x_2} g_3^{x_3} g_4^{x_4}\)
  - \(h' = h / (g_1^{3} g_2^{5})\)
  - \(g_1' = g_1^2 g_2^4 g_3\)
  - \(g_2' = g_4\)
  - \(a = g_1'^{w_1} g_2'^{w_2}\)
  - \(r_1 = c x_3 + w_1\)
  - \(r_2 = c x_4 + w_1\)

- **Check:**
  - \((g_1')^{r_1} (g_2')^{r_2} = (h')^c a \Rightarrow (g_1')^{(c x_3 + w_1)} (g_2')^{(c x_4 + w_1)} = (h / (g_1^{3} g_2^{5}))^{c} g_1'^{w_1} g_2'^{w_2} \Rightarrow (g_1^{2x_3 + 3} g_2^{4x_3 + 5} g_3^{x_3} g_4^{x_4}) = h\)
- Showing any relation implies knowing all attributes.
- Can make non-interactive (message m)
  - \( c = H[h, m, a'] \)
- Other proofs:
  - (OR) connector (*simple concept*)
    - \( x_{\text{age}} = 18 \ AND \ x_{\text{city}} = H[\text{Cambridge}] \) OR \( x_{\text{age}} = 15 \)
  - (NOT) connector
  - Inequality \( x_{\text{age}} > 18 \)
Summary of key concepts (1)

- **Standard tools**
  - Schnorr – ZK proof of knowledge of discrete log.
  - DL rep. – ZK proof of knowledge of representation.

- **Credential showing**
  - representation + certificate
  - ZK proof of linear relations on attributes (AND)
  - More reading: (OR), (NOT), Inequality
1. Issuing protocol:
Prover gets a certified credential.

2. Showing Protocol:
Prover makes assertions about some attributes

Credential
\[ h = g_1^{x_1} g_2^{x_2} \cdots g_l^{x_l} \]

\[ \text{Sig}_{\text{Issuer}}(h) \]
Issuing security

- **Issuing: What do we want?**
  - Peggy authenticates and provides a list of attributes.
  - Issue checks all and provides a signed credential.
    - In the form we discussed previously.

- **Peggy needs to do two things:**
  - Blind the credential.
    - Multiple times
  - Prove that she possesses a valid signature on it.
    - Without revealing the actual signature.

- **Solution:** the CL signature scheme.
CL Signature Scheme

- **Setup:**
  - Generate and RSA modulus $n = pq$
    (with $p = 2p' + 1$, $q = 2q' + 1$, $p, q, p', q'$ large primes)
  - Choose $g_1, \ldots, g_l, b, c$
    (all of which are quadratic residues)
  - Public key = $(n, g_1, \ldots, g_l, b, c)$
    Private Key = $p, q$

- **Signature:**
  - Attributes: $x_1, \ldots, x_l$
  - Pick a random prime $e$, and random $s$
  - $v = (c / ((g_1)^{x_1} \ldots (g_l)^{x_l} b^s)^{1/e}) \mod n$
  - Output signature $(e, s, v)$
    - Cannot forge because $(.)^{1/e}$ requires knowledge of $p, q$
How to verify a CL signature?

- Reminder
  - Public: $c$, $g_i$, $b$, $n$
  - $v = (c / ((g_1)^{x_1} \ldots (g_l)^{x_l} b^s)^{1/e}) \mod n$
  - Signature $(e, s, v)$

- Zero-knowledge DL Rep. Proof:
  - Get a random $r$
  - Define $v' = v b^r$
  - Reveal: $v'$
  - DL Rep. proof of: $c = (v')^e ((g_1)^{x_1} \ldots (g_l)^{x_l} b^{s-er})$
Does that work?

- \( c = (v')^e (g_1)^{x_1} \ldots (g_l)^{x_l} b^{s-er} \)
  - \( c = (v b^r)^e (g_1)^{x_1} \ldots (g_l)^{x_l} b^s b^{-er} \)
  - \( c = (v)^e (b^{re}) (g_1)^{x_1} \ldots (g_l)^{x_l} b^s b^{-er} \)

  - Remember: \( v = (c / ((g_1)^{x_1} \ldots (g_l)^{x_l} b^s)^{1/e} \)

- \( c = ((c / ((g_1)^{x_1} \ldots (g_l)^{x_l} b^s)^{1/e})^e (g_1)^{x_1} \ldots (g_l)^{x_l} b^s \)
  - \( c = (c / ((g_1)^{x_1} \ldots (g_l)^{x_l} b^s) ((g_1)^{x_1} \ldots (g_l)^{x_l} b^s \)
  - \( c = c \)
Unforgeability of signature

Based on Strong RSA assumption:

- Impossible to find a $v'$
- Without computing $(.)^{1/e}$
- Which is infeasible without $p$, $q$
- Prover does not know $p$, $q$ (only $n$)
Privacy

- Unlikability of signature and showing
  - Signature (e, s, v)
  - Showing (v’) + ZK proof
    - V and v’ are unlinkable
    - Proof does not learn s, e

- Result:
  - We can show the credential many times.
  - Each time is unlikable to the others.
  - One issue – many (unlinkable) uses.
Putting it all together:

- CL signature proof is already a DL proof:
  \[
  c = (v')^e (g_1)^{x_1} \cdots (g_l)^{x_l} b^{s-er}
  \]

- Integrate all previous tricks to reveal or show relations on attributes.

- E.g. show attributes \( x_1 \) and \( x_2 \):
  - Reveal \( x_1 \) and \( x_2 \)
  - Show \( c / (g_1)^{x_1}(g_2)^{x_2} = (v')^e (g_3)^{x_3} \cdots (g_l)^{x_l} b^{s-er} \)
Key concepts so far (2)

- Credential issuing
  - Authentication & Authorization
  - Signing (using CL)

- Showing Credential
  - Re-randomize and proof possession of signature
  - Integrate proof over attributes

- Further topics
  - Transferability of credential
  - Double spending
Key applications

- Attribute based access control
- Federated identity management
- Electronic cash
  - (double spending)
- Privacy friendly e-identity
  - Id-cards & e-passports
- Multi-show credentials!
References

Core:


More:

- Jan Camenisch and Anna Lysianskaya. *A signature scheme with efficient proofs*. (CL signatures)