Identity management

Bart, Bas, Maikel
What are we going to discuss

- Introduction to identity management
- Current IdM issues
- Technologies
  - OAuth
  - U-prove
  - ABC-credentials
Identity explained

- Entity, scope and characteristics
- Not the same as identifier
- Multiple identities within the same scope
Identity defined

The identity of an entity within a scope is the set of all characteristics that have been attributed to this entity within that scope. (Alpár, Hoepman, Siljee)
So what is identity management?

Identity Management (IdM) consists of the processes and all underlying technologies for the creation, management and usage of digital identities. (Alpár, Hoepman, Siljee)
Three parties

- User

- Relying Party (or Service Provider)

- Identity Provider
IdM Systems - Network based

1. request service
2. authenticate at IdP
3. authentication result
4. exchange additional info

network-based
IdM Systems - Claim Based

Diagram showing the interaction between IdP, U, and RP.

- 'cachable' steps
  1. request service
  3. authenticate
  4. send claims

- claim-based

Steps:
1. Request service
2. Send policy
3. Authenticate
4. Send claims
5. Supply claims
Federation and circle of trust
Privacy and IdM

- You control which information you disclose and to whom
- Reduces information overspill
- One way to manage all identities
- No need to enter the same data every time
IdM Problems?

Network and claim based problems?

Federation problems?

What do you think?
IdM Problems

- IdP is a single point of failure
- One authentication to rule them all
- Federation trust issues
- Linkability issue with IdP
- Identity is limited by the available characteristics
Double edged sword

- Over usage
- Harder data obfuscation
Schemes

- Scheme authority
- RP has to let the authority sign his policy (requested attributes)
- Authority creates trust by enforcing policy on IdP and RP
Example

- Education roaming
- Federated IdM
- Authentication and Authorisation
(Privacy Enhancing) Technologies

- OAuth
- U-prove
- and some other technologies...
Open Authorisation

- Network based IdM system
- Open standard for authorization, not authentication
- Used by Google, Twitter, Github, Yandex etc.
OAuth message flow

1) User requests access to service

2) Redirect to IdP

3) Return token

4) Send token to RP

5) Request scope info

User authenticates to IdP and determines scope
OAuth message flow

1. request service
2. authenticate at IdP
3. authentication result
4. exchange additional info

optional step

network-based
OAuth problems

- SPOF
- Phishing
- Confidentiality
- Traceability
- Scope (grouping)
OpenId

- Proving who you are instead of granting access to certain resources / attributes
- Layer on top of OAuth to improve security
OAuth and OpenId

- OAuth is meant for Authorisation
  - Requesting access to attributes of an identity
  - Attributes can be used as identifier for an identity

- OpenId is meant for Authentication
  - Requesting an identity
  - After authentication OAuth can be used for attributes
Zero knowledge example: The Cave

- Based on the idea of Quisquater
- Prover (User) & Verifier (RP)
- The Prover goes in and chooses a side, then the Verifier goes in and tells which side the prover must come out
Zero knowledge example: The Cave

- Repeat this for example 10 times
- Why could the verifier not just ask the prover to walk the full circle?
Zero knowledge example: The Cave

- Someone who looks at the transcript doesn’t know if the prover knows the secret
- The transcript is not trustworthy
U-prove

- U-prove is a privacy enhancing technology
- First integral description by Stefan Brands
- Currently owned by Microsoft
- Makes use of DL-problem and zero-knowledge proof
U-prove identity management

- U-prove is claim based
- User is in control
- Selective reveal of attributes chosen by the user
U-prove privacy challenges

- U-prove tries to solve some (privacy) problems with identity management (in distributed environments)
  - DDOS attack on IdP
  - IdP/RP tracking (timing attack between IdP/RP)
  - Leaking of user tokens
  - … + a few more

- Device protected tokens possible
The protocol

1. I want to have access
2. I want to see these attributes
3. User authenticates to IdP
4. IdP sends all ‘obfuscated’ attributes to user
5. User reveals the attributes he wants to.
User is checked with challenge - response

IdP

RP

User

- Attribute 1
- Attribute 2
- ...
- User PK

Signed by IdP
Problem: How can a RP check a user is not lying?

- RP cannot communicate with IdP
- Solution: signature of IdP is added
- The IdP sends all the user attributes in a token (signed) to the user
Problem: How can a RP check a user is not lying?

- User can choose not to send some attributes
- It needs to ‘delete’ the attributes from the IdP response (token)
- What is the problem here?
  - Message cannot be changed, it is signed by IdP
Possible fix: use hashes

- Hash all attributes
- RP cannot reveal the attributes
- User can choose to send his attributes plain after
- RP can check with hashes if plain attributes are correct
U-prove way

- U-prove uses DL-problem to hide attributes
- Instead of $H(a_1), H(a_2), H(a_3) \ldots$ it uses generators with attributes in the exponents
- It verifies the attributes with a zero-knowledge proof
U-prove math

- U-prove math is complicated, we will simplify a bit
- Full specification: “U-Prove Cryptographic Specification V1.1” (Panquin, Zaverucha)
- We will call the IdP Issuer, the User Prover and the RP Verifier
DL-problem

- Discrete logarithm problem is in the form: \( G^e = H \mod P \)
- Generator G, prime P and H are public
- e is a random number that is private
- Given H it is (extremely) hard to find exponent e
Create generators

- Issuer (IdP) choose prime \( P \) and subgroup \( Q \) of the form \( P = 2 \times Q + 1 \) where \( Q \) is also a prime
- \( G_q \) are generators from subgroup \( Q \). The property of the generators are:
  \[ G^{q-1} = G \]
  \[ G^q = 1 \]
Attribute encoding

- The Issuer (IdP) who generates the token hides the prover (User) attributes in the token.
- It uses a subgroup generator for every attribute for this.
- We calculate $g_1^{a_1} g_2^{a_2} g_3^{a_3} g_4^{a_4} \pmod{p}$ etc. for every attribute.
- $\alpha$ is the user private key, is added for extra security ($\beta$)

\[ \prod_{i \in A} g_i^{a_i} \cdot \beta \]
Protocol scheme

\[ H = \prod_{i \in A} g_{ai}^* \times \beta \]

\[ \beta = g^\alpha \]

{H}_{\text{Sign}}

Verification of attributes

Selective disclosure of attributes (+ challenge-response)
Verification of attributes

- $A =$ revealed attributes $B =$ non revealed attributes
- User sends $A$, $B$, $H$ (+ public info like generators) to Verifier
- Verifier checks $A \times B = H \mod P$
- $B$ is not safe! Why?
Schnorr protocol

- For ZkP of B the Schnorr protocol is used
- We will explain a simplified version of this protocol
Disclosed: $a_1 \ldots a_3$, $B = g_4^{a_4} g_5^{a_5} \ldots g_8^{a_8} g_9^\alpha$

Random values in exponents: $A = g_4^{w_4} g_5^{w_5} \ldots g_8^{w_8} g_9^\beta$

Challenge $C$

$R_4 = C*a_4 + w_4$ \ldots $R_9 = C*\alpha + \beta$

$g_4^{R_4} g_5^{R_5} \ldots g_8^{R_8} g_9^{R_9} \overset{?}{=} B^C \ast A$
Why does this work?

Let’s take a simple example:

\[ B = g_1^{a_1} \quad A = g_1^{w_1} \]

\[ R_1 = c * a_1 + w_1 \]

\[ g_1^{R_1} = g_1^{c*a_1+w_1} \quad ? \quad g_1^{a_1*c} * g_1^{w_1} = B^c * A \]
Some extra advantages

- What if the id provider is not available?
  - Tokens can last longer than one session
- RP cannot track activity
  - User key-pair is refreshed for every new token
- Tokens are useless for other users
  - You need to know the attributes and the private key
ABC’s

- U-prove alternative: Identity Mixer
- These technologies make use of “Attribute-based credentials”
- Some can be implemented on a smart-card (for example the IRMA-card)
Honourable mention

- Hub of all things
  - Not really IdM
  - Collect and control your own private data
  - Data in exchange for services/offers
  - Team of researchers worldwide working at it
Questions?