

ABC4Trust & PrimeLife Tutorial

Part IV: Use Cases and 2nd Demo

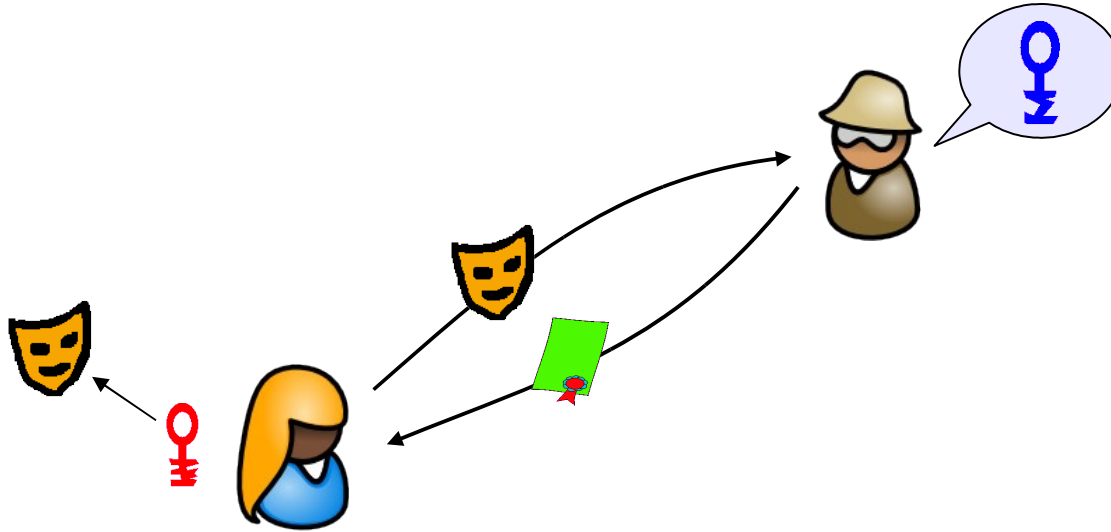




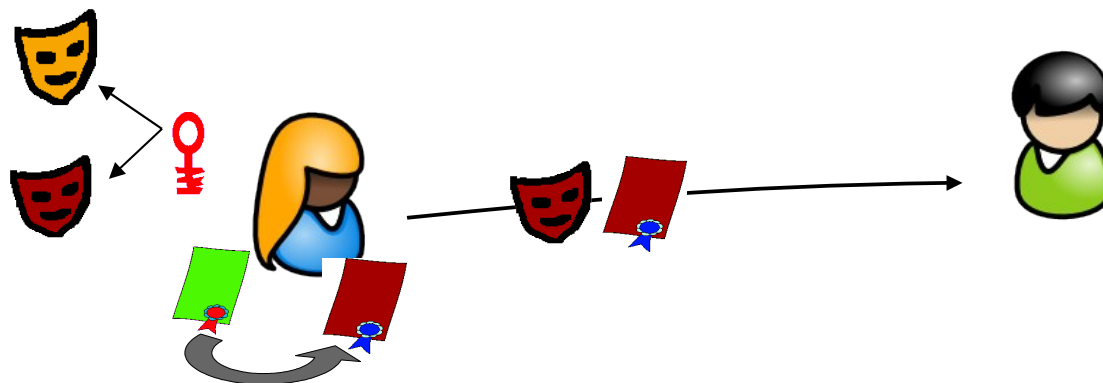
Use Case: Polling


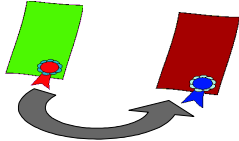
Scenario:

- Pollster(s) and a number of users
- Only registered user (e.g., students) can voice opinion
- User can voice opinion only once (subsequent attempts are dropped)
- Users want to be anonymous
- A user's opinion in different polls not linkable



- User generates pseudonym (Id for registration)
- User obtains credential on pseudonym stating that she is eligible for polls
- Credential can contain attributes about her



1. User generates domain pseudonym, domain = pollID 
2. User transforms credential 
3. Transformed credential with a subset of the attributes
 - User is anonymous and unlinkable
 - Multiple opinions are detected because uniqueness of domain pseudonym



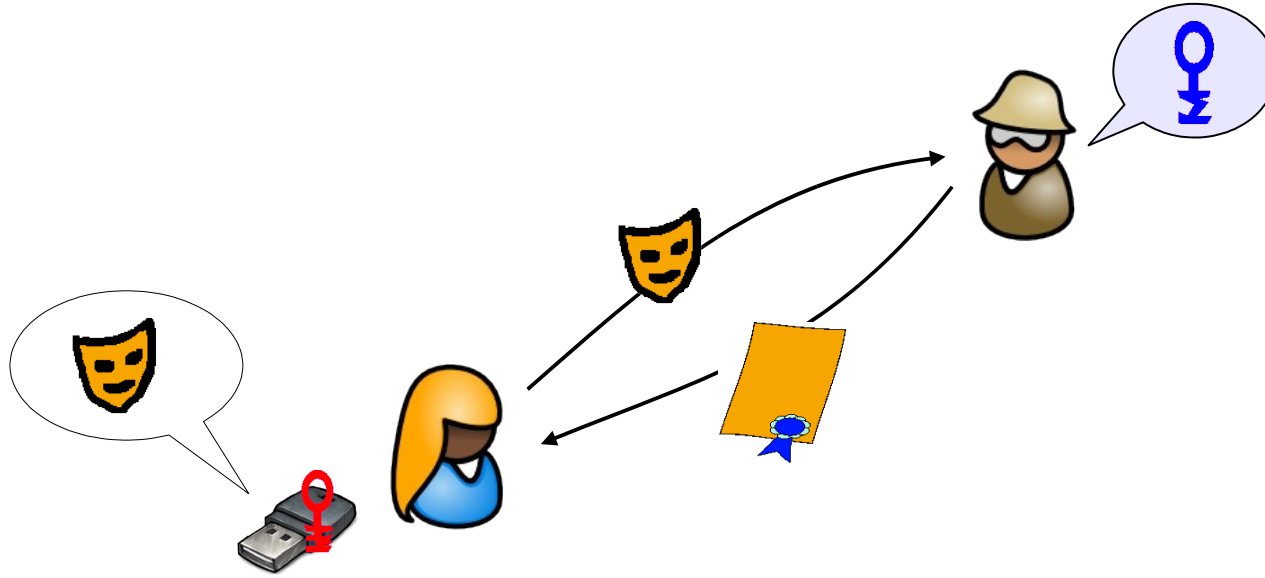
Use Case: Hotspot



Scenario:

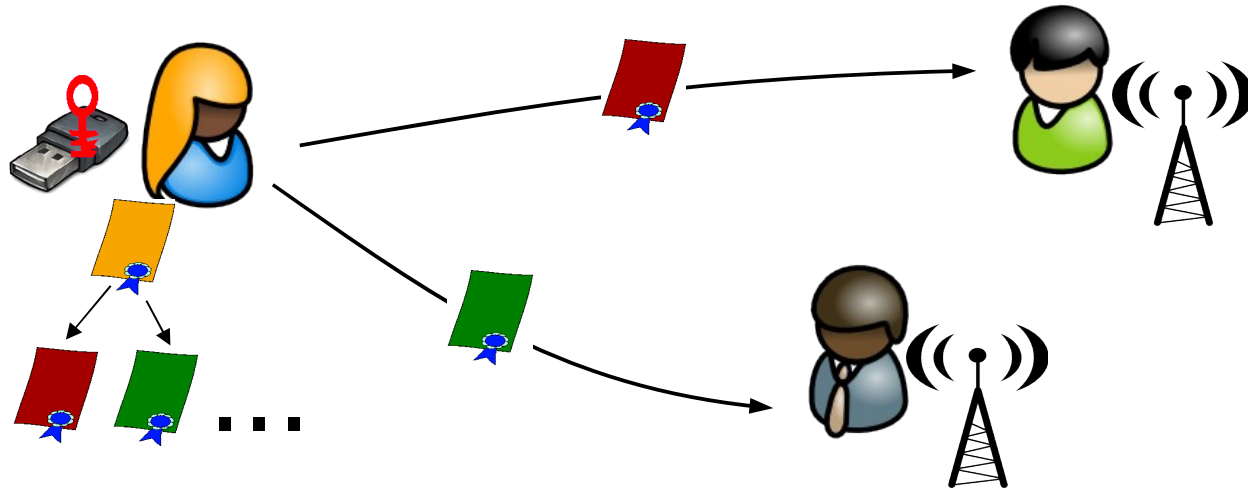
- Several hotspots and a number of users
- Subscribed users obtain USB-dongle
- Only users with valid subscription & dongle are able to log-in
- User can not share her subscription with other users (at the same time)
- User wants to use hot-spot anonymously and such that different log-ins are unlinkable
- *(If USB-dongle gets lost, user wants to be able to lock his account)*

Hotspot – Solution: Registration



- User subscribes for the hotspot usage and receives a dongle
- Dongle locally creates public/private key pair
- User obtains credential on public key of the dongle stating that she is allowed to use hotspots

Hotspot – Solution: Log-In



User derives fresh credential for each log-in

- including a proof that she possess the corresponding usb-dongle
- User is anonymous and different log-ins are unlinkable
- Sharing of subscription is prevented by tying the credential to the usb-dongle
- Lost credentials/dongles get revoked by the Issuer



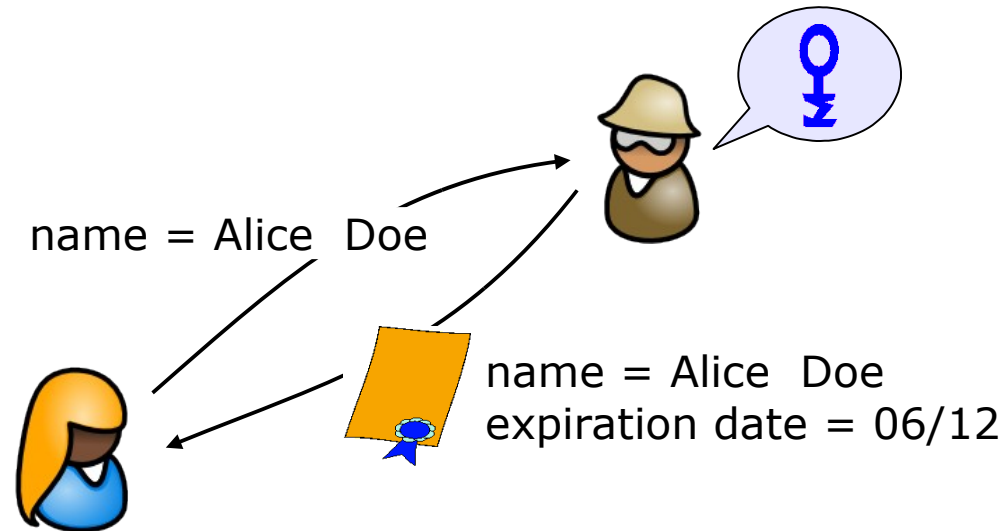
Use Case: Library



Scenario:

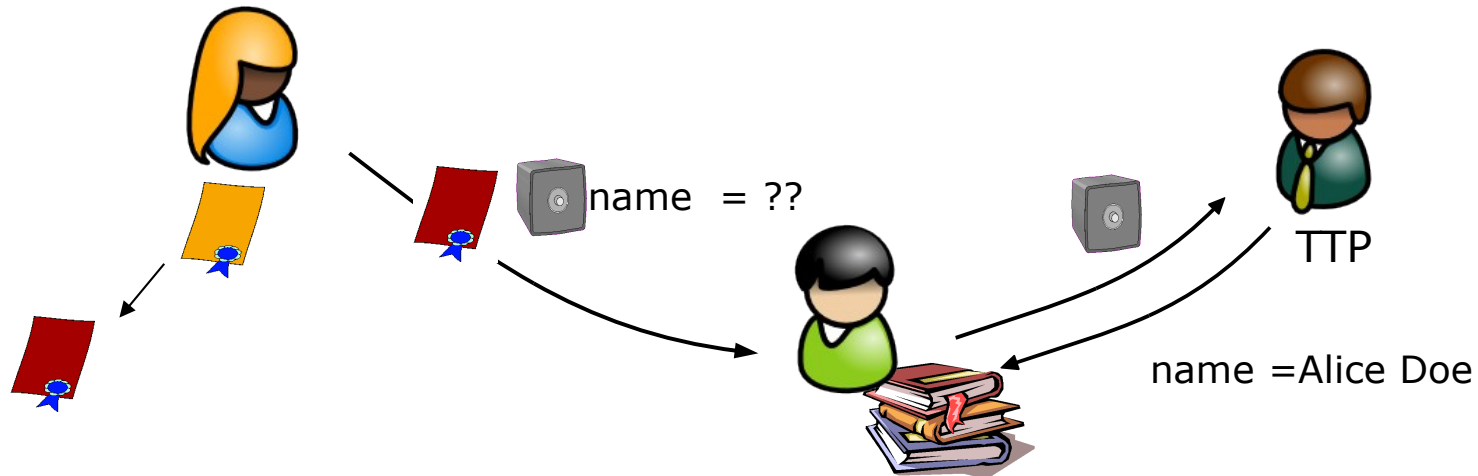
- (Several) libraries and a number of users
- Only registered users are allowed to lend books
- User wants to lend books anonymously and unlinkable
 - ... But if a user doesn't return a book in time, the library wants to know his identity

Library – Solution: Registration



- User obtains credential on her name (or some other identifying information)
- credential states that she is eligible to use the library for the next year

Library – Solution: Log-In



User derives fresh credential each time she lends a book where

- she proves that the credential is not expired yet
- does not reveal her ID/name but provides a verifiable encryption of it (+proof that it is the same as in credential)
- User is anonymous and different book lends are unlinkable
- If user doesn't return the book, library contacts TTP which reveals the ID/name of the user



Demo: Teenage Chatroom



- Proof of age with government issued credentials (or Mobile Operator)
- Examples: Teenage Chatroom, Gaming,
- Scenario:
 - User gets letter at mailing address with activation code
 - User goes to government website
 - Enters activation code
 - Obtains credential with Name, Address, Birth date, etc
 - User goes to Teenage Chatroom
 - Chatroom does attribute based AC: $11 < \text{age} < 16$
 - User transforms credential into one that just proves $11 < \text{age} < 16$
 - User chats away happily (if proof succeeded)

Demo Time



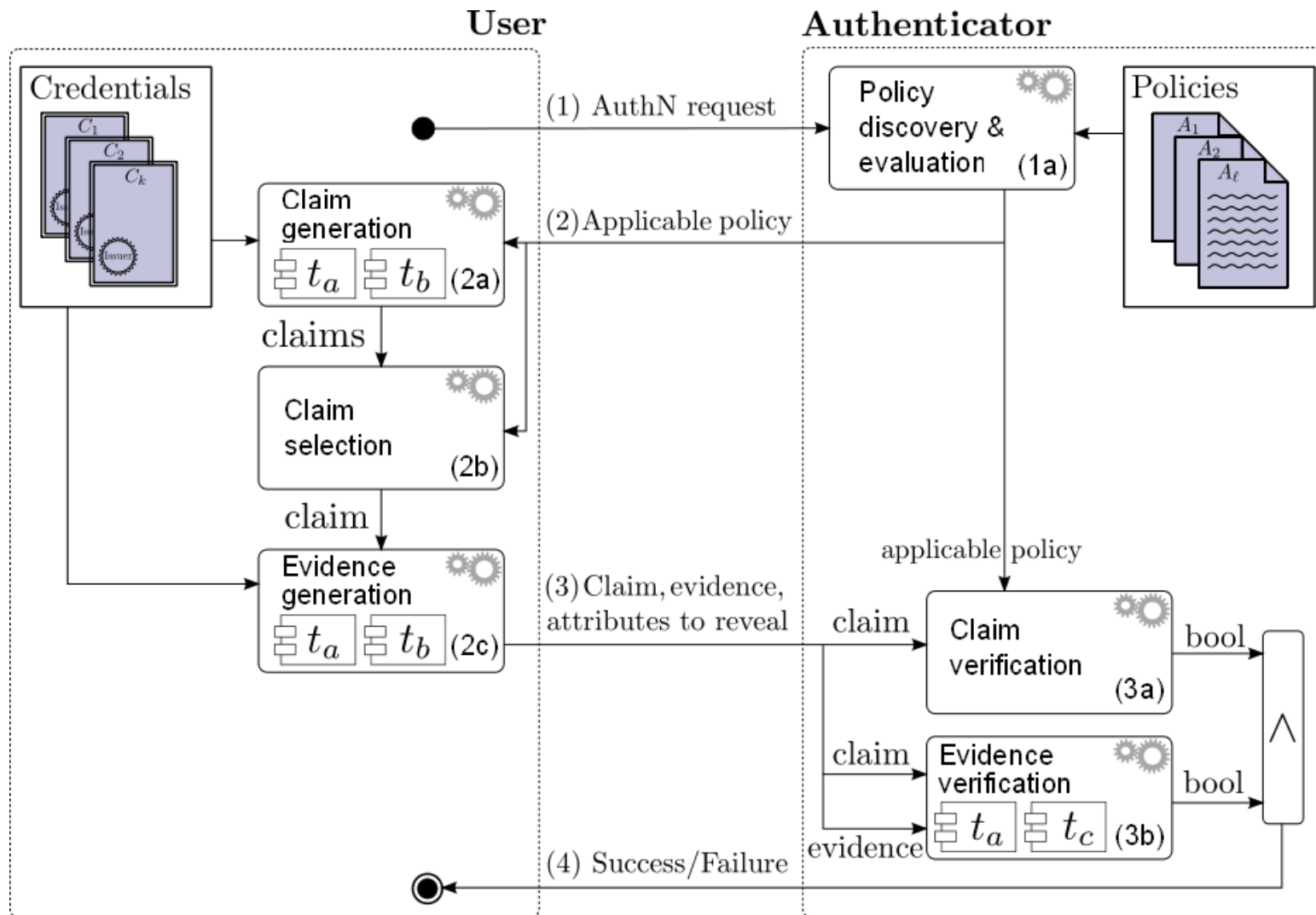
The idemix Library



Implementation available :-)

- Identity Mixer is an implementation of Private Credentials
- Provides a library with all the crypto
 - Issuing credentials
 - Transforming credentials according to a specified statement (policy)
 - Includes many of the features discussed
- Provides a credential-based AC engine
 - Relying party specifies attributes & credentials requirements
 - User matches that to available credentials and generates „evidence“
- Get it at www.PrimeLife.eu/opensource and use it
 - ..as do a number of projects already :-)

Authentication/Access Control Engine



Card-based access requirements language (CARL)

Policy and proof presentation in CARL and SAML/XACML

- Policy: requirements on owned cards, e.g.,
own p::Passport **issued-by** admin.ch, fgov.be, governo.it
own c::Creditcard **issued-by** visa.com, amex.com
reveal c.number, c.expdate
where p.name = c.name ^ p.bdate < today-18Y
 ^ c.expdate > today ^ p.expdate > today+1M
- Authentication = *claim* over owned cards + *evidence*, e.g.,
own p::Passport **issued-by** admin.ch
own c::Creditcard **issued-by** visa.com
reveal c.number = "1234567890"
reveal c.expdate = "31/12/2012"
where p.name = c.name ^ p.bdate < 22/03/1993 ^ p.expdate >
22/04/2011

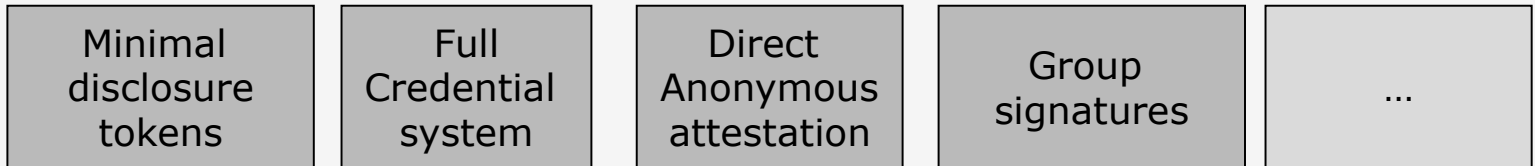
IBM Identity Mixer: Framework

Policy Layer



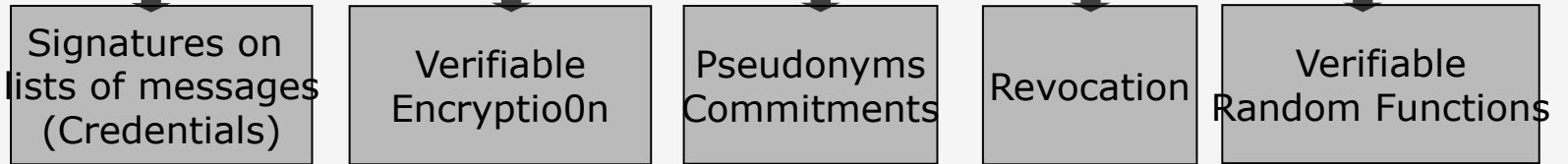
Crypto Token Layer

Composed schemes

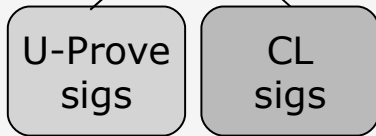


Efficient zero-knowledge proofs

Building blocks



Instantiations



“Token Transforming” Language

```
Declaration{ id1:unrevealed:string; id2:unrevealed:string; id3:unrevealed:int;
             id4:unrevealed:enum; id5:revealed:string; id6:unrevealed:enum }
```

```
ProvenStatements{
```

```
  Credentials{ randName1:http://www.ch.ch/passport/v2010/chPassport10.xml =
                { FirstName:id1, LastName:id2, CivilStatus:id4 }
                randName2:http://www.ibm.com/employee/employeeCred.xml =
                { LastName:id2, Position:id5, Band:5, YearsOfEmployment:id3 }
                randName3:http://www.ch.ch/health/v2010/healthCred10.xml =
                { FirstName:id1, LastName:id2, Diet:id6 } }
```

```
  Inequalities{ {http://www.ibm.com/employee/ipk.xml, geq[id3,4]} }
```

```
  Commitments{ randCommName1 = {id1,id2}; randCommName2 = {id6} }
```

```
  Representations{ randRepName = {id5,id2; base1,base2} }
```

```
  Pseudonyms{ randNymName; http://www.ibm.com/employee/ }
```

```
  VerifiableEncryptions{ {PublicKey1, Label, id2} }
```

```
  Message { randMsgName = “Term 1:We will use this data only for ...” }
```

```
}
```

ABC4Trust & PrimeLife Tutorial

Questions?

www.abc4trust.eu

www.primelife.eu

www.zurich.ibm.com/security/idemix

