January 11, 2011

Private Credentials

And Their Application to Voting

Jan Camenisch

Project Leader Security
Member, IBM Academy of Technology
IBM Research – Zurich

Houston, we have a problem!

“Neil Armstrong’s Footsteps are still there”

(Robin Wilton, Sun Microsystems)

Computers don't forget!

- Storage becomes ever cheaper
- Data mining ever better

And we leave traces, lots of traces!

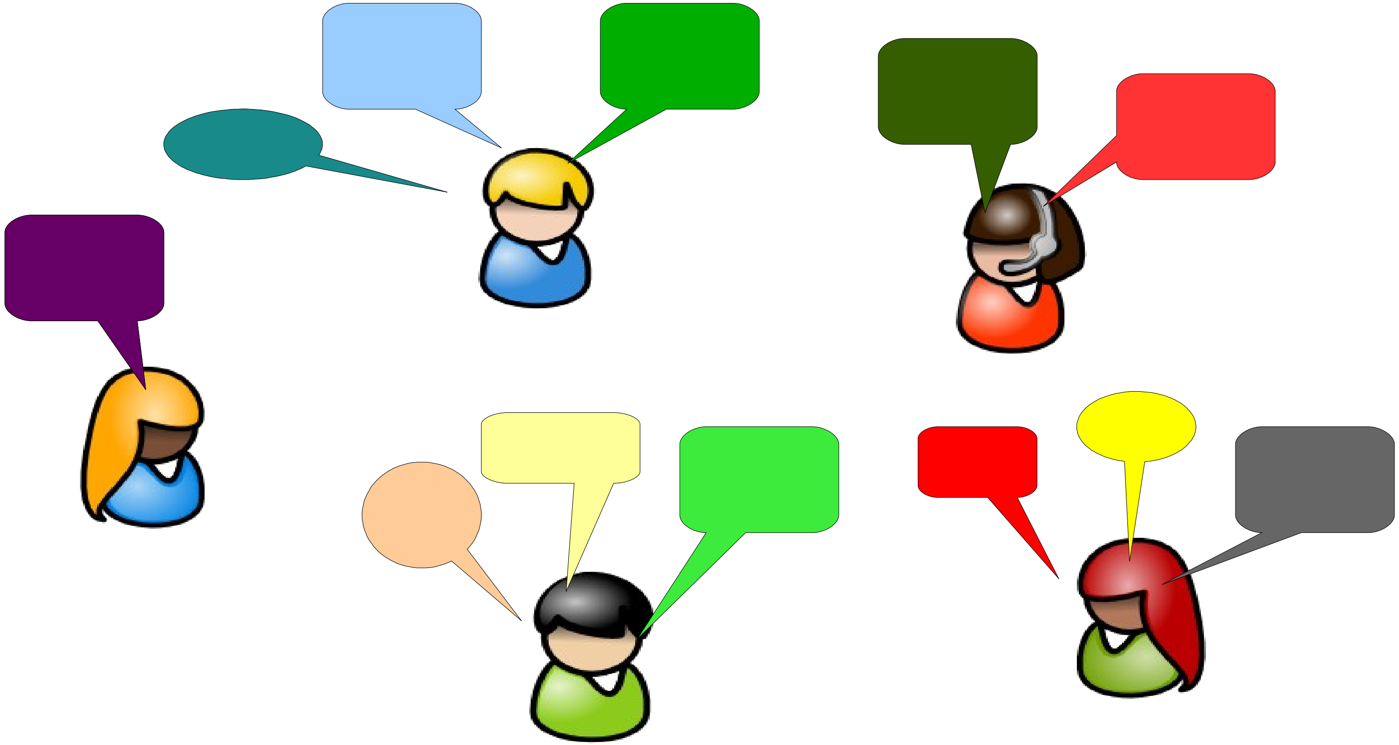
EUROPEAN UNION
IDENTIFICATION CARD
ISSUED FOR

FULL NAME
SARAH JANE BLOGGS
PLACE OF BIRTH
LONDON
NATIONALITY
BRITISH CITIZEN
DATE OF BIRTH
16-09-84
SIGNATURE

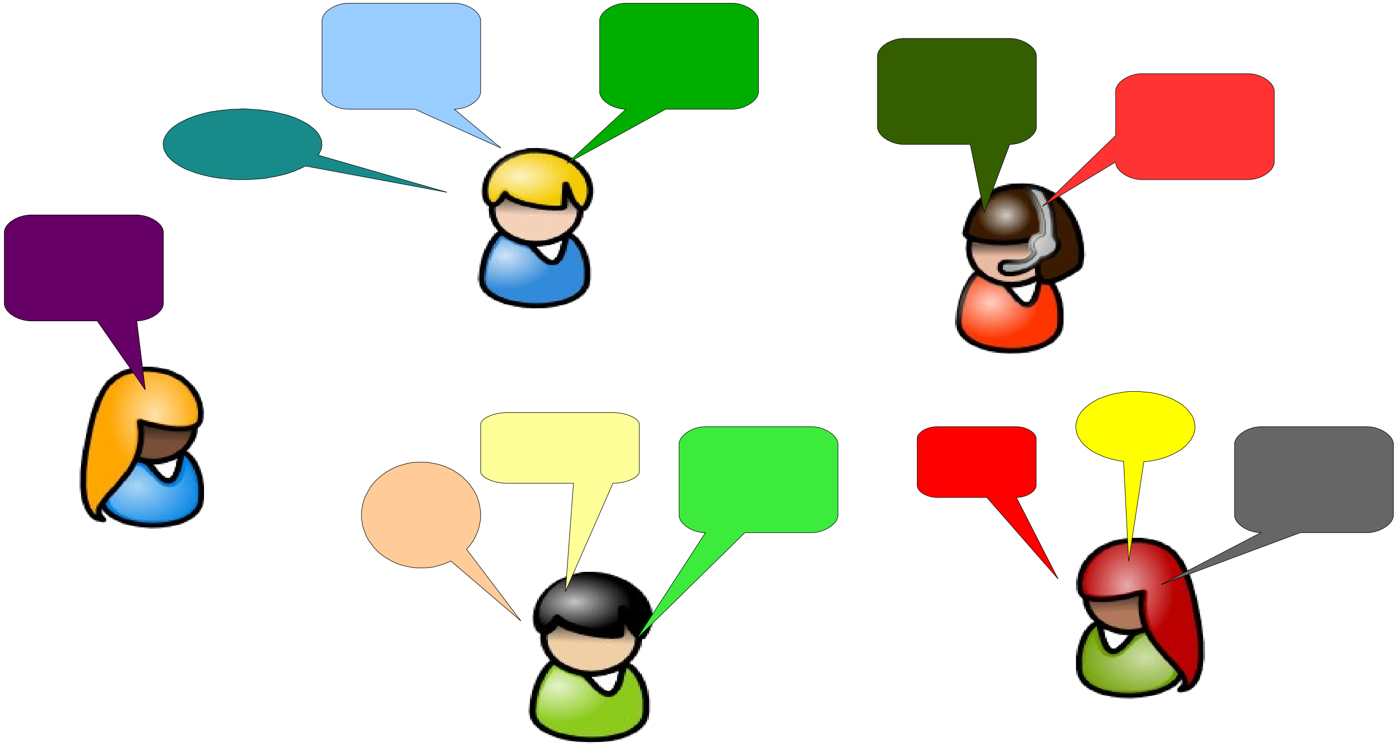
SERIAL NUMBER ISSUE CENTRE CODE REF
34545 4545676-4555554

Not only computers but also people...

… people who like to talk



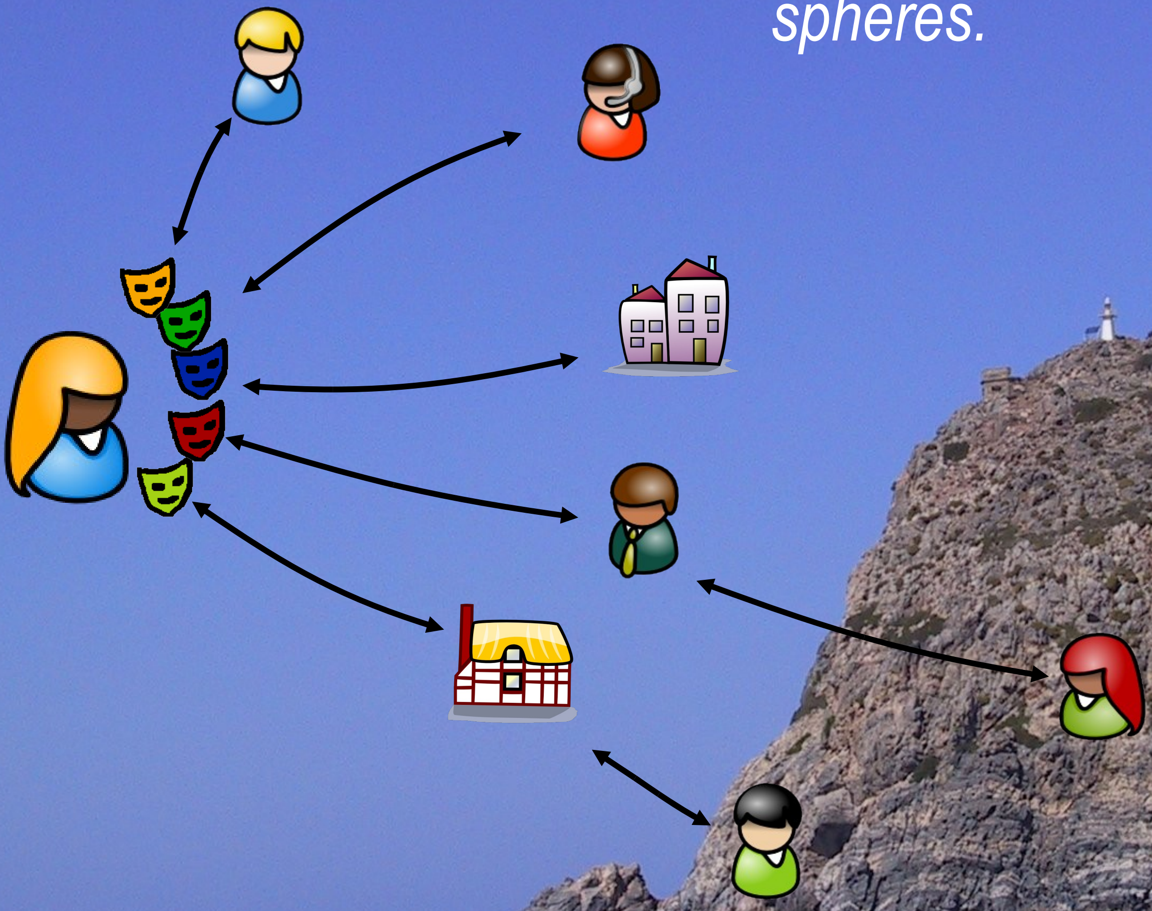
… people who like to talk



- Distributing Information is easier
- Controlling it much harder
- Establish trust and security even harder

Our Vision

In the Information Society, users can act and interact in a safe and secure way while retaining control of their private spheres.



David, please help!?

Mix Networks

Oblivious Transfer

Searchable Encryption

Onion Routing

Confirmer signatures

Anonymous Credentials

Group signatures

Pseudonym Systems

OT with Access Control

e-voting

Priced OT

Blind signatures

Private information retrieval

Secret Handshakes

Homomorphic Encryption

(Crypto) PETs Can Help! - A More Structured Approach

PET to be built-in everywhere

- Network Layer Anonymity
  - ... in mobile phone networks
  - ... in the Future Internet as currently discussed
  - ... access points for ID cards
- Identification Layer
  - Access control & authorization
- Application Layer
  - “Standard” e-Commerce
  - Specific Apps, e.g., eVoting, OT, PIR, ...
  - Web 2.0, e.g., Facebook, Twitter, Wikis, ...

Overview

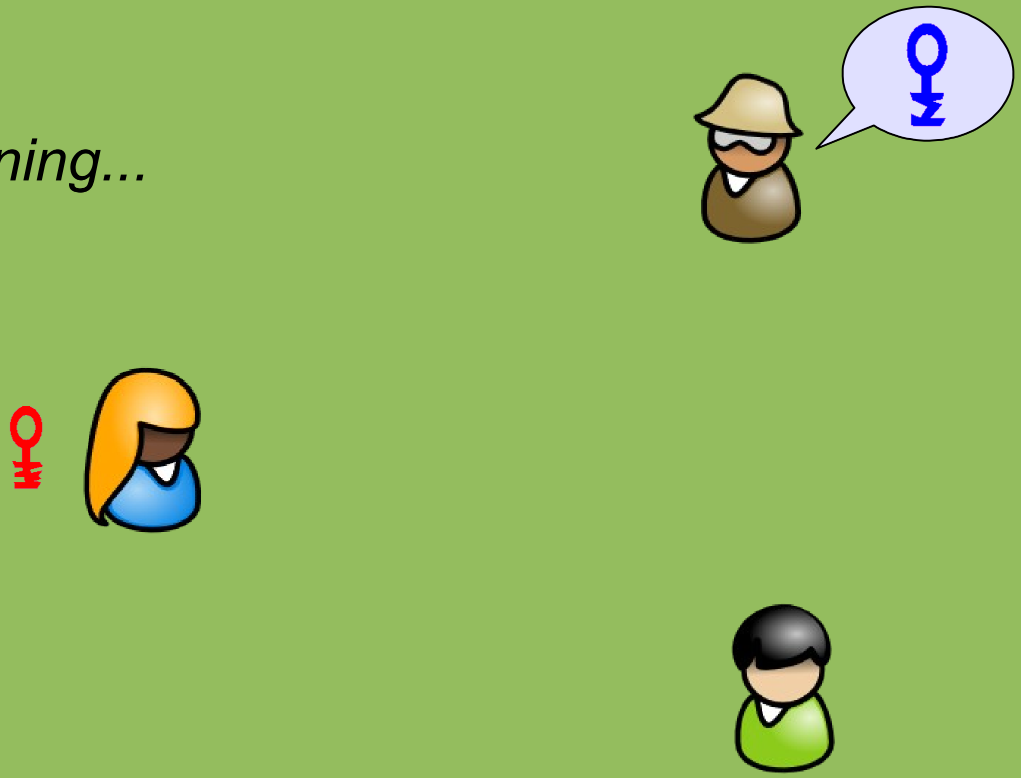
- PETs – Identification Layer
- Private Credentials
  - High-Level Basic
  - Crypto
  - High-Level Advanced
- How to use Crypto PETs
- Private Credentials and Voting

What PETs Can Do

The Identification Layer

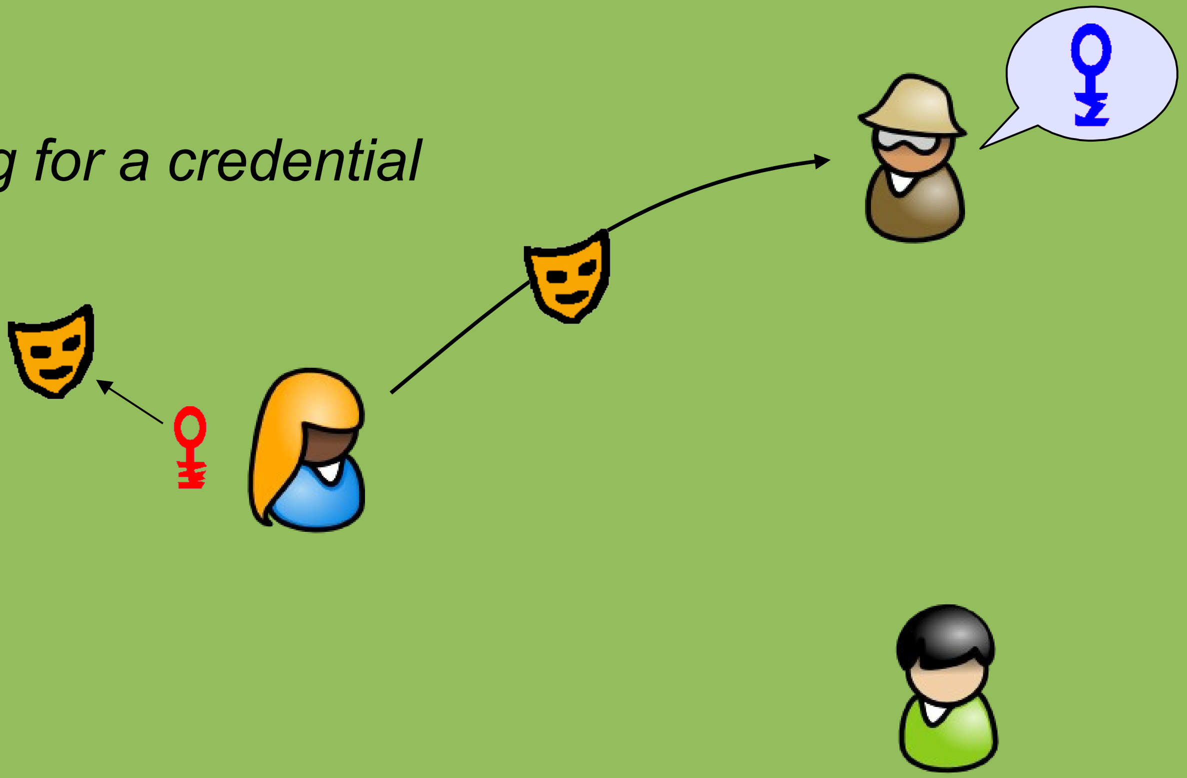
Private Credentials: How to Build Them

In the beginning...



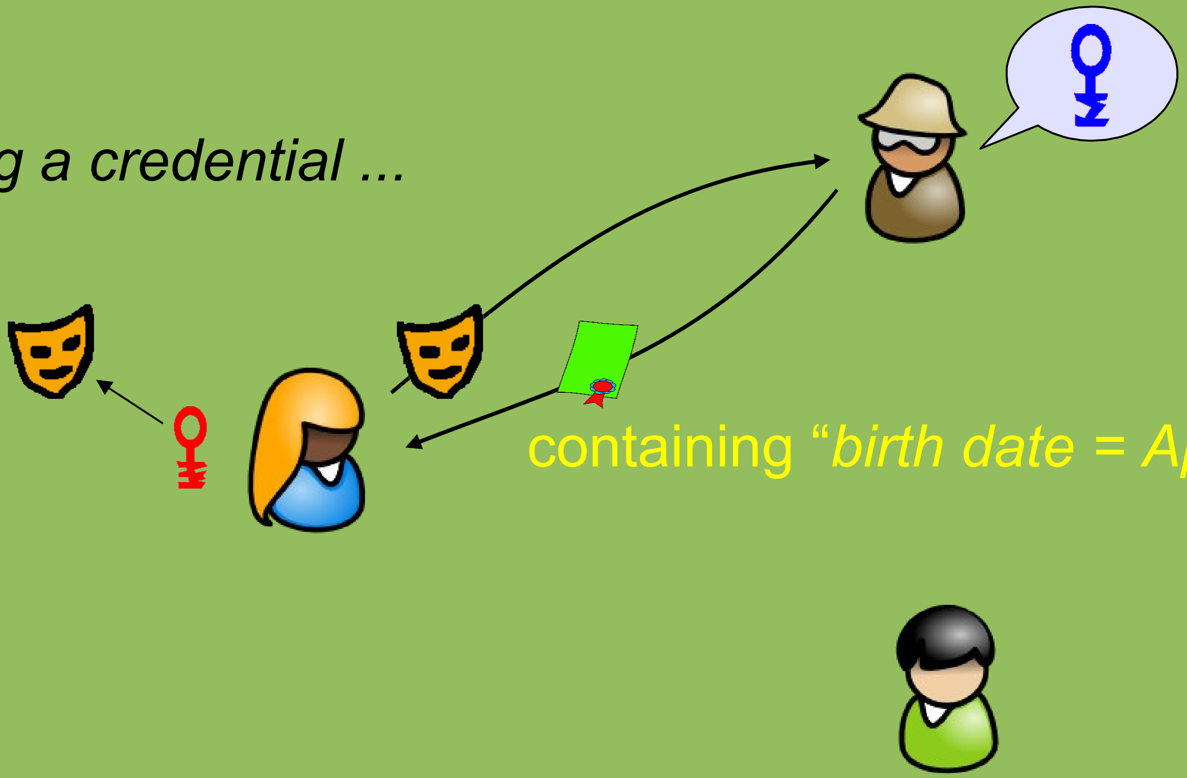
State of the Art: How to Build Them

asking for a credential



State of the Art: How to Build Them

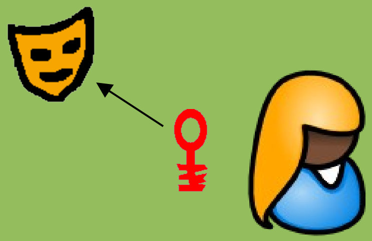
getting a credential ...

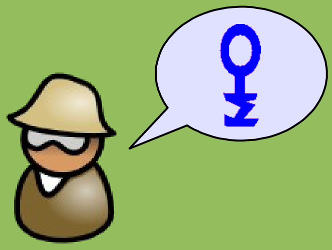


containing “*birth date = April 3, 1987*”

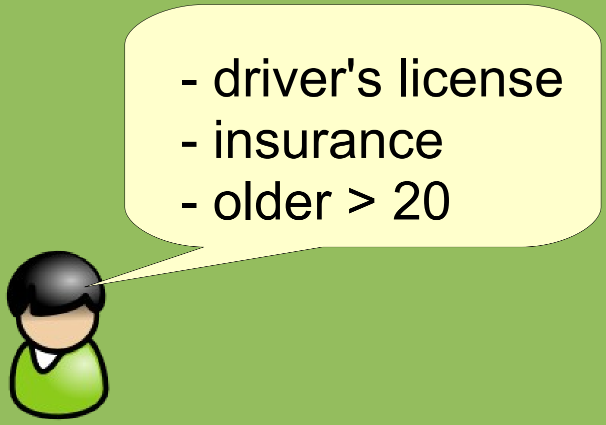
State of the Art: How to Build Them

showing a credential ...





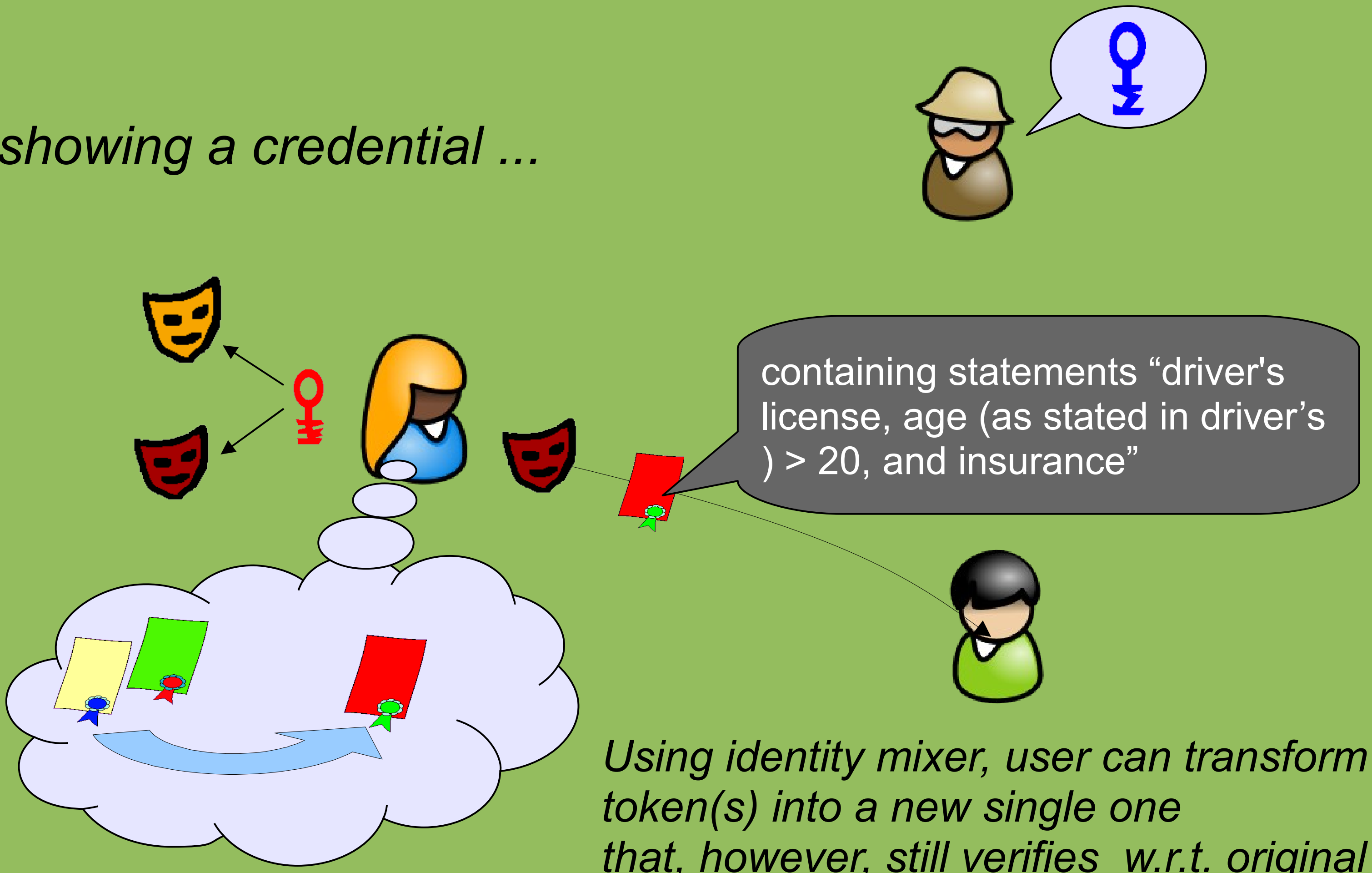
goes off-line



- driver's license
- insurance
- older > 20

State of the Art: How to Build Them

showing a credential ...

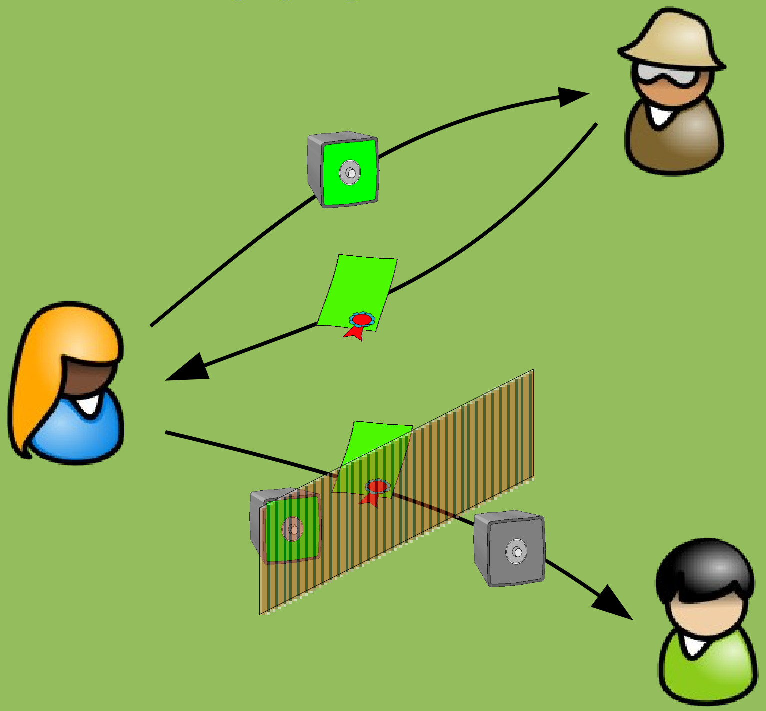


containing statements “driver's license, age (as stated in driver’s) > 20, and insurance”

Using identity mixer, user can transform (different) token(s) into a new single one that, however, still verifies w.r.t. original signers' public keys.

Two Approaches

ZK Proofs

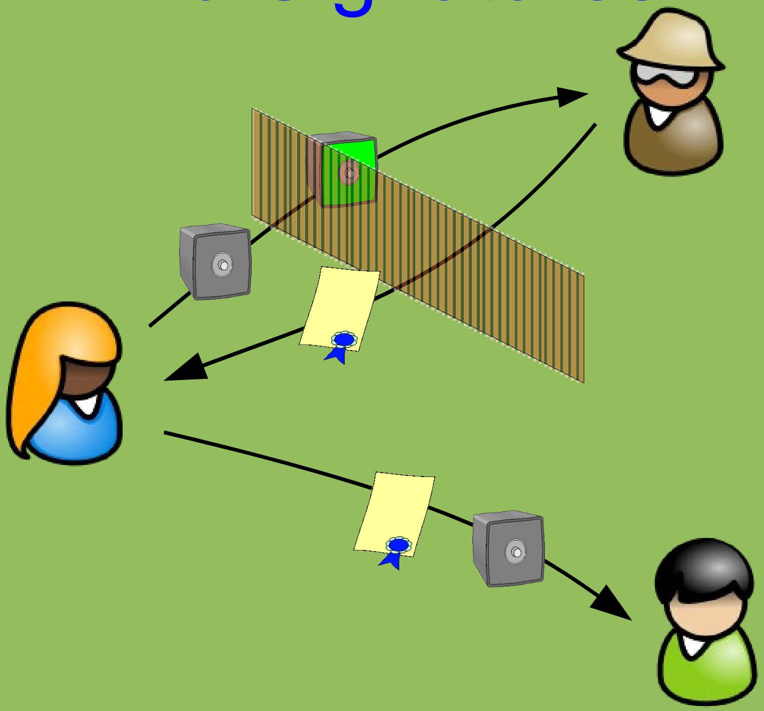


can be used multiple times

Damgaard,Camenisch&Lysyanskaya

Strong RSA, DL-ECC,..

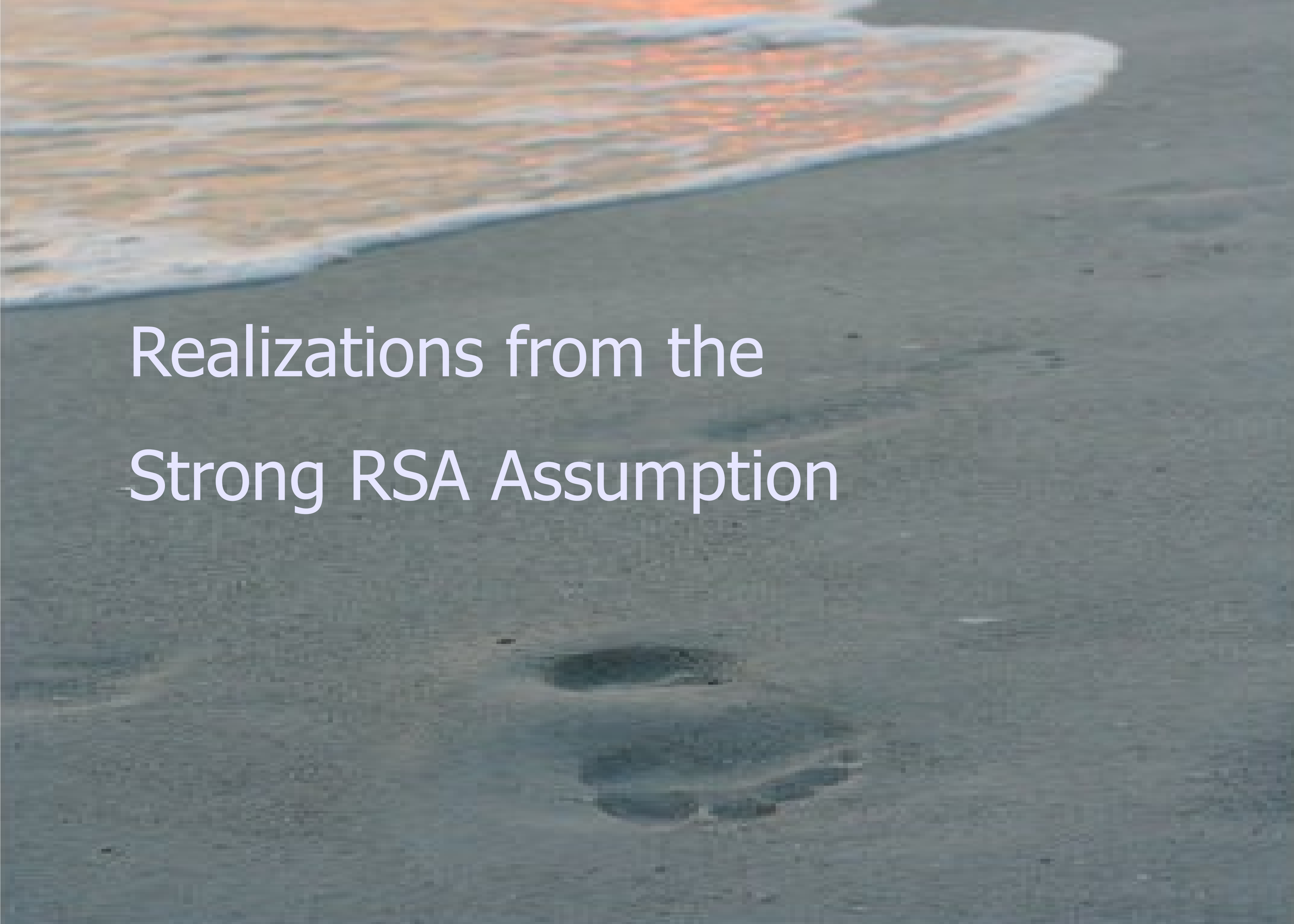
Blind Signatures



can be used only once

Chaum, Brands, et al.

Discrete Logs, RSA,..



Realizations from the Strong RSA Assumption

The Strong RSA Assumption

Flexible RSA Problem: *Given RSA modulus* n *and* $z \in QR_n$ *find integers* e *and* u *such that*

$$u^e = z \bmod n$$

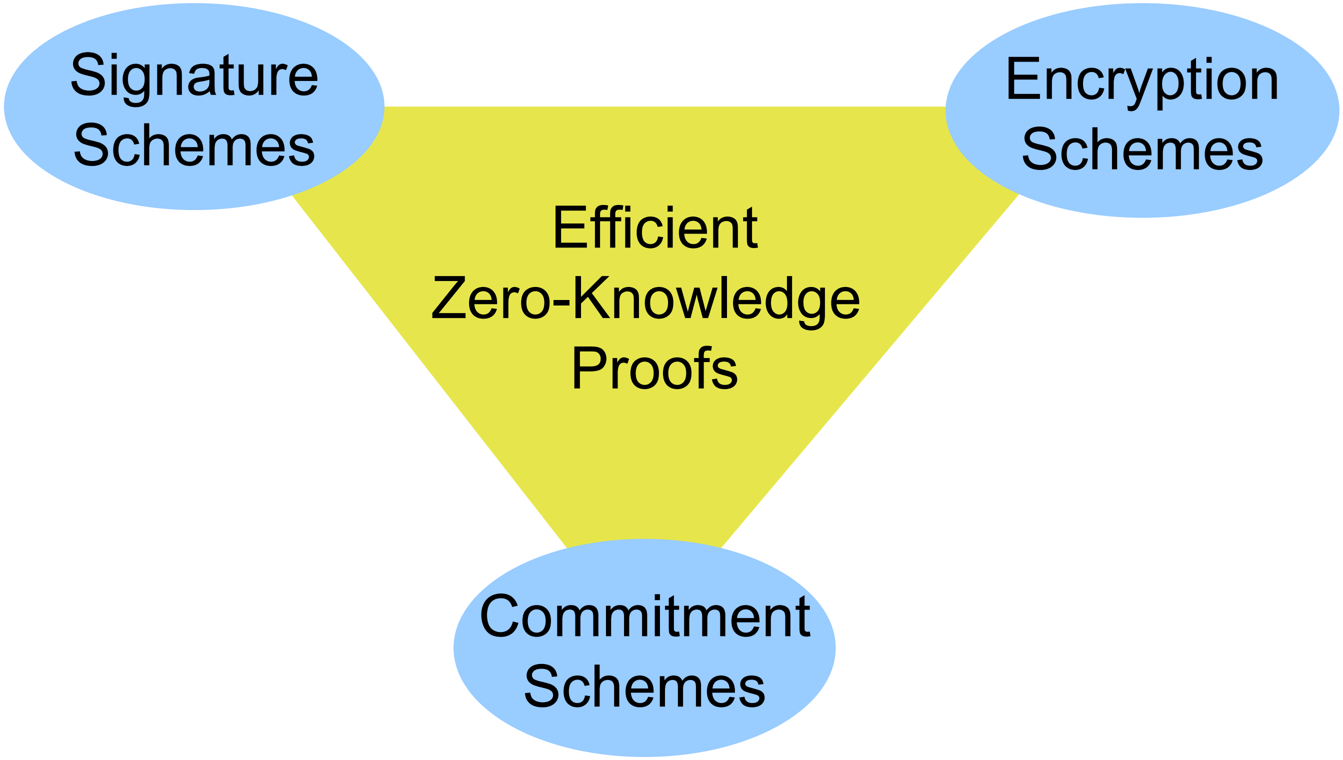
- Introduced by Barić & Pfitzmann '97 and Fujisaki & Okamoto '97
- Hard in generic algorithm model [Damgård & Koprowski '01]
- Turned out to be useful in security analysis of many protocols

A Useful Lemma

Lemma [CS02]: *Given RSA modulus* n *and* $g,h \in QR_n$ *it is hard to find integers* a,b,c *and* u *such that*

$$u^c = g^a h^b \bmod n \quad \text{and} \quad c \nmid a \text{ or } c \nmid b$$

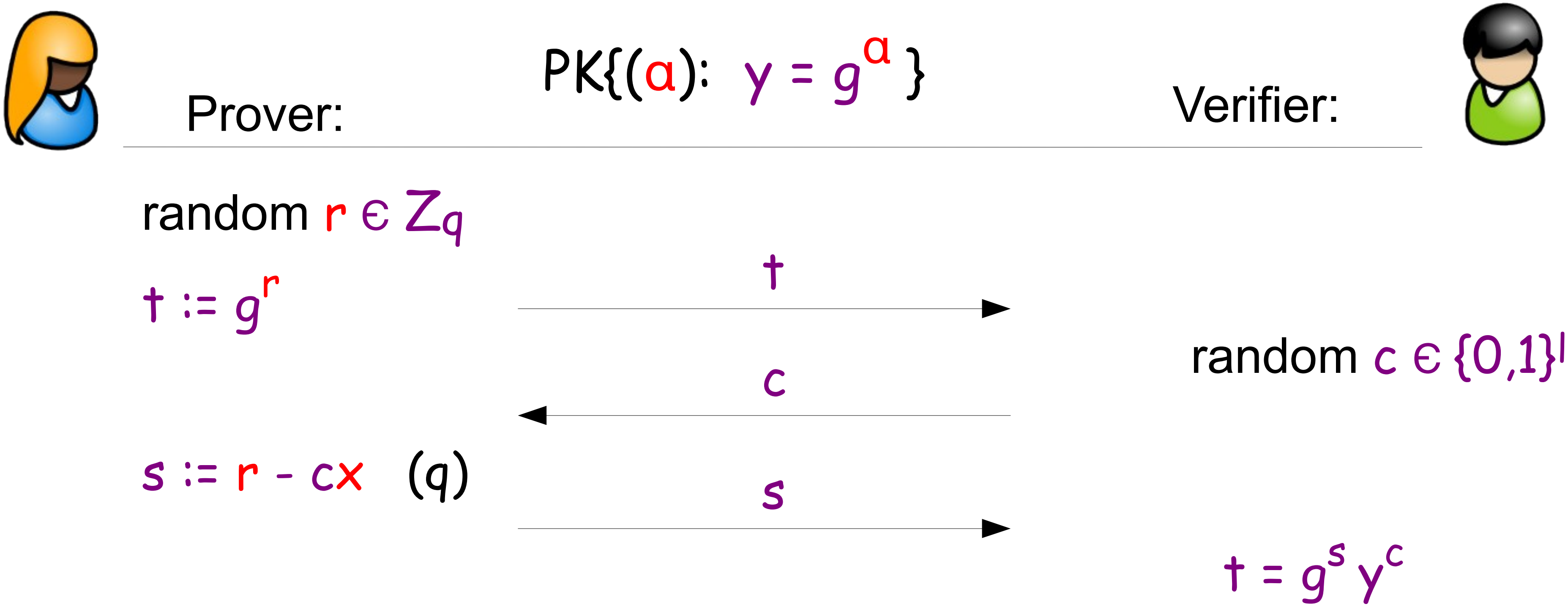
Building Blocks



This One We Know All

Given group $\langle g \rangle$ and element $y \in \langle g \rangle$.

Prover wants to convince verifier that she *knows* $x = \log_g y$ such that verifier only learns y and g.
Let l be a security parameter.



$PK\{(\alpha): y = g^{\alpha}\}$

Prover:

Verifier:

random $r \in Z_q$

$t := g^r$

t

random $c \in \{0,1\}^l$

c

$s := r - cx \quad (q)$

s

$t = g^s y^c$

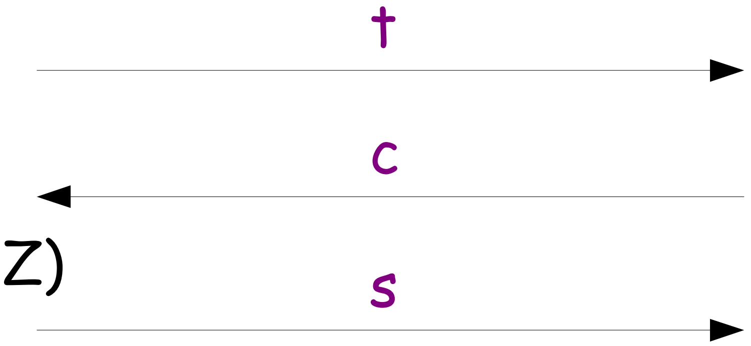
What if the Order of the Group is not Known

$PK\{(\alpha):\ y = g^{\alpha}\}$

Prover: Verifier:

random $r \in Z$

$t := g^{r}$



t

c

s

random $c \in \{0,1\}^{l}$

$s := r - cx$ (in Z)

$t = g^{s} y^{c}$

Knowledge Extractor:

(t,$c1$,$c2$) and (t,$c1$,$c2$) $\rightarrow\ t = g^{s1} y^{c1} = g^{s2} y^{c2}$

$\rightarrow\ y^{c1-c2} = g^{s2-s1}$

… but cannot compute $\alpha = (s2-s1)/(c1-c2)$ as order is unknown!

Strong RSA Assumption to the Rescue

… but cannot compute $\alpha = (s2-s1)/(c1-c2)$

$$y^{c1-c2} = g^{s2-s1}$$

Under the *Strong RSA assumption* (use our little Lemma):

(c1-c2) must divide (s2-s1)

$(s2-s1) = \alpha\ (c1-c2) \rightarrow\ y^{c1-c2} = g^{s2-s1} \rightarrow y = bg^{\alpha}$

If n is product of safe prime, one can get rid of b

$$y = g^{\alpha}$$

Thus verifier must not know the order of the group!!!!

If the Order is not Known: Proving length

Prover: $PK\{(\alpha):\ y = g^{\alpha} \wedge \alpha \in \pm\{0,1\}^{ls}\}$ Verifier:

random $r \in Z$

$t := g^{r}$ $\xrightarrow{t}$

$\xleftarrow{c}$ random $c \in \{0,1\}^{lc}$

$s := r - cx$ (in Z) $\xrightarrow{s}$ $t = g^{s} y^{c}$

If we check that $s \in \{0,1\}^{ls}$

then $(s2-s1) = \alpha\,(c1-c2) \in \pm\{0,1\}^{ls}$

and thus $\alpha \in \mp\{0,1\}^{ls}$

Note that $ls = lx + lc + lz$, i.e, $x \in \pm\{0,1\}^{ls - lc - lz}$

So there is some fudge here!!

Summary: Efficient ZK Proofs for/about DLs

Logical combinations:

$PK\{(\alpha,\beta): \ y = g^{\alpha} \wedge z = g^{\beta} \wedge u = g^{\beta}h^{\alpha}\}$

$PK\{(\alpha, \beta): \ y = g^{\alpha} \vee z = g^{\beta}\}$

Non-interactive (Fiat-Shamir heuristic / Random Oracle):

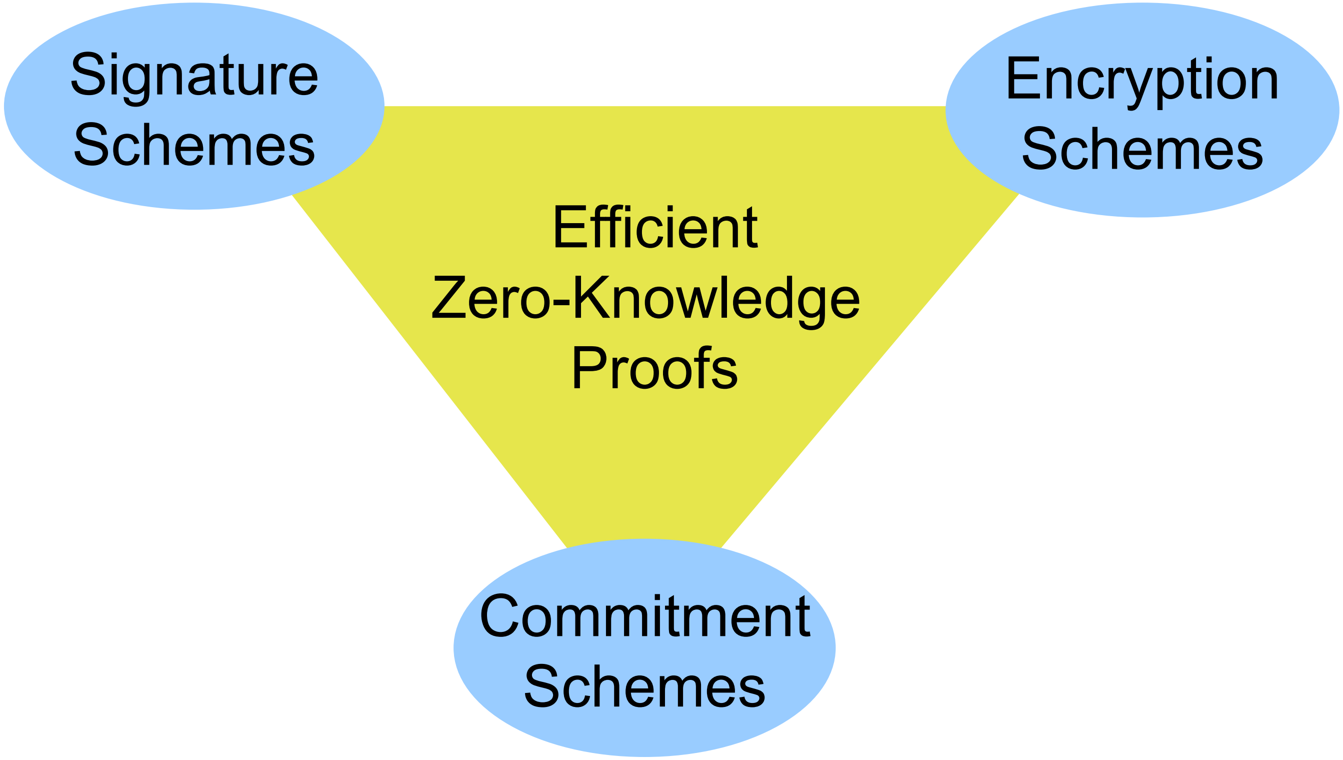
$SPK\{(\alpha): \ y = g^{\alpha}\}(m)$

Intervals and different groups (under SRSA):

$PK\{(\alpha): \ y = g^{\alpha} \wedge \alpha \in [A,B]\}$

$PK\{(\alpha,b): \ y = g^{\alpha} \wedge \mathbf{z}=\mathbf{g}^{\alpha} \wedge \mathbf{w}=\mathbf{g}^{\alpha}\mathbf{h}^{b} \wedge \alpha \in [0,\min\{\#(g),\#(\mathbf{g})\}]\}$

Building Blocks



Signature Scheme based on SRSA [CL01]

Public key of signer: RSA modulus n and a_i, b, d $\in QR_n$

Secret key: factors of n

To sign k messages $m1, \ldots, mk \in \{0,1\}^\ell$:

I. choose random prime $e > 2^\ell$ and integer $s \approx n$

II. compute c such that

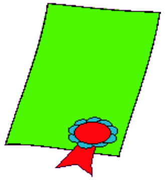
$$d = a_1^{m1} \cdot \ldots \cdot a_k^{mk}\, b^s\, c^e \mod n$$

III. signature is (c,e,s)

Signature Scheme based on SRSA [CL01]

A signature (c,e,s) on messages $m1, \ldots, mk$ is valid iff:

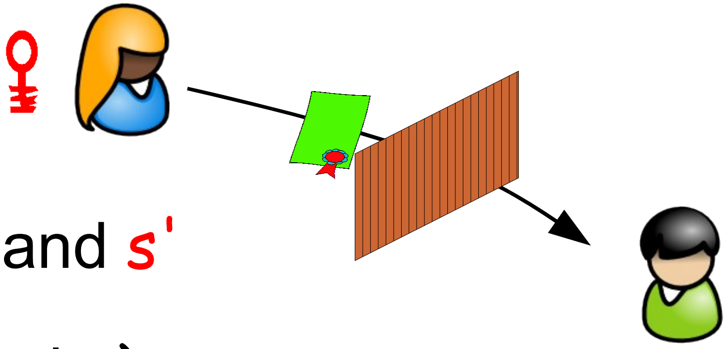
- $m1, \ldots, mk \in \{0,1\}^{\ell}$:
- $e > 2^{\ell}$
- $d = a_1^{m1} \cdot \ldots \cdot a_k^{mk} \, b^s \, c^e \bmod n$



Theorem: *Signature scheme is secure against adaptively chosen message attacks under SRSA assumption.*

Proof of Knowledge of a Signature

Observe:



- Let c' = $c\ b^{s'}$ mod n with randomly and s'
- then $d = \ c'^{e}\ a_1^{m1} \cdot \ldots \cdot a_k^{mk}\ b^{s^*}$ (mod n),

  i.e., (c', e, s^*) is a also a valid signature!

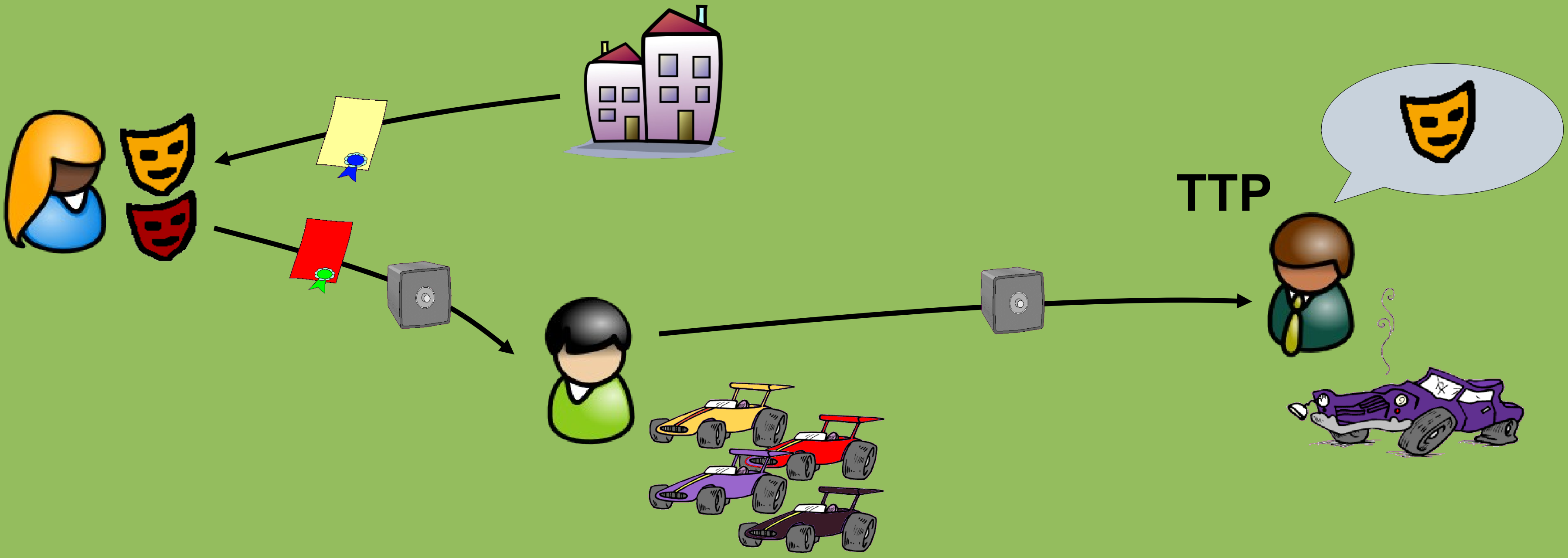
Therefore, to prove knowledge of signature on some m

- provide c'
- PK{$(e, m1, \ldots, mk, s)$: $d := c'^{e} a_1^{m1} \cdot \ldots \cdot a_k^{mk}\, b^{s}$

  $\wedge\ mi \in \{0,1\}^{\ell}\ \wedge\ e \in 2^{\ell+1} \pm \{0,1\}^{\ell}$ }

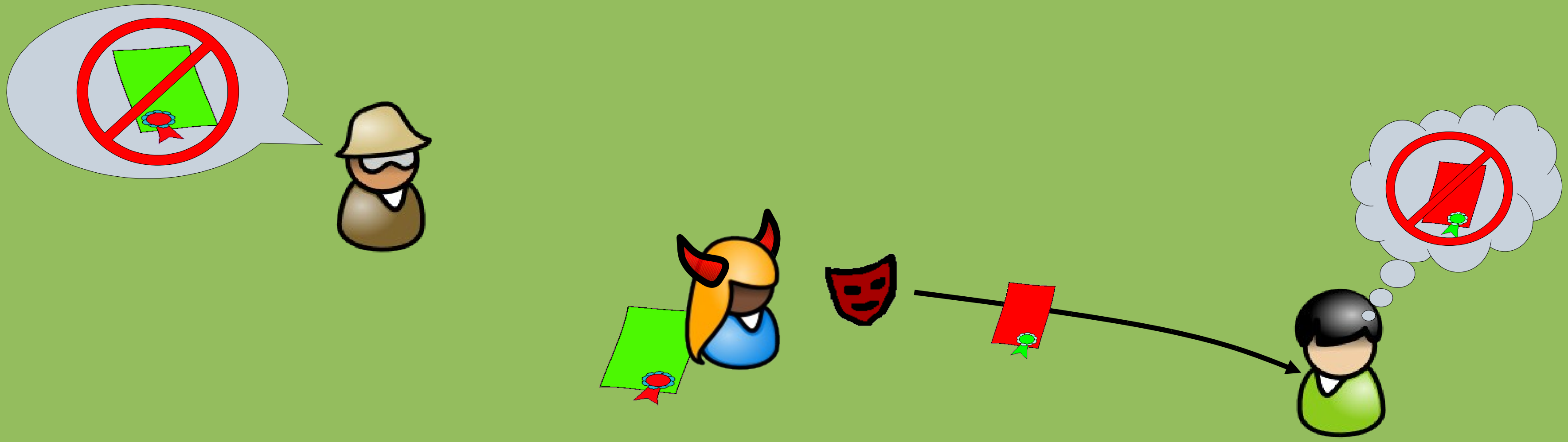
Back to What We Can Do

Other Properties: Attribute Escrow (Opt-In)



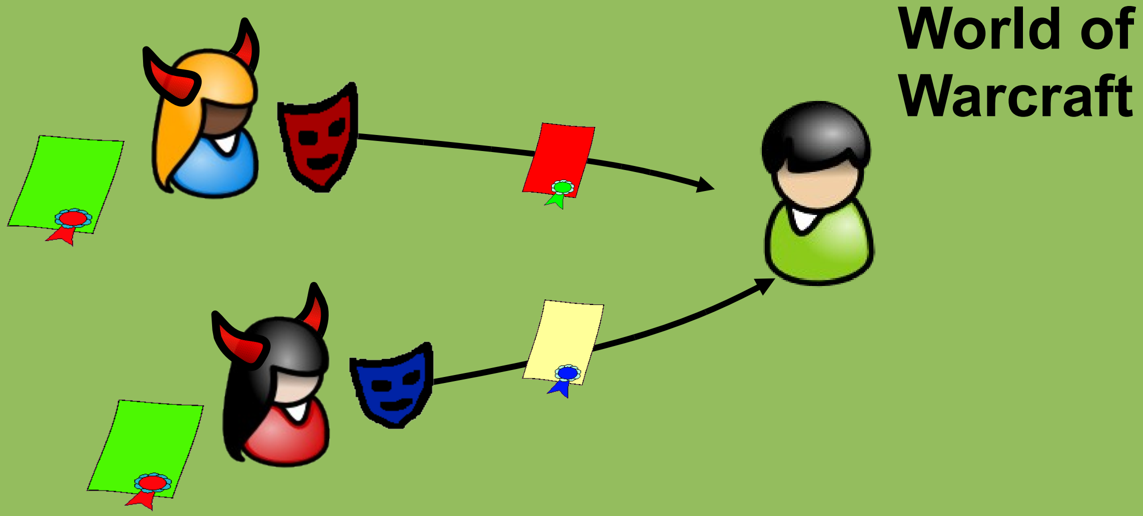
- If car is broken: ID with insurance needs be retrieved
- Can verifiably encrypt any certified attribute *(optional)*
- TTP is off-line & can be distributed to lessen trust

Other Properties: Revocation



- If Alice was speeding, license needs to be revoked!
- There are many different use cases and many solutions
  - Variants of CRL work (using crypto to maintain anonymity)
    - Accumulators
    - Signing entries & Proof, ...
  - Limited validity – certs need to be updated
  - ... For proving age, a revoked driver's license still works

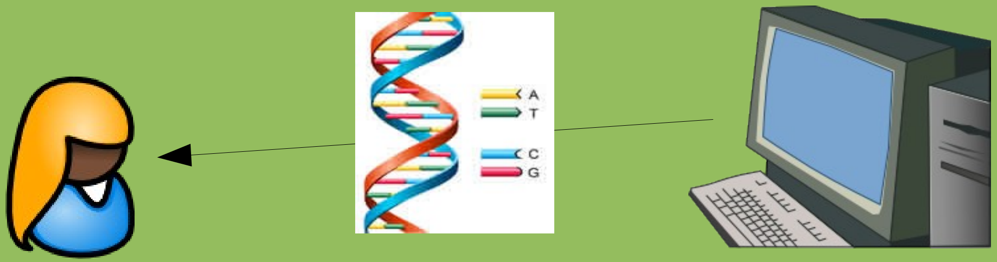
Other Properties: Cheating Prevention



Limits of anonymity possible *(optional)*:

- If Alice and Eve are on-line together they are caught!
- Use Limitation – anonymous until:
  - If Alice used certs > 100 times total...
  - ... or > 10'000 times with Bob
- Alice's cert can be bound to hardware token (e.g., TPM)

Privacy Preserving Access Control [CDN09]



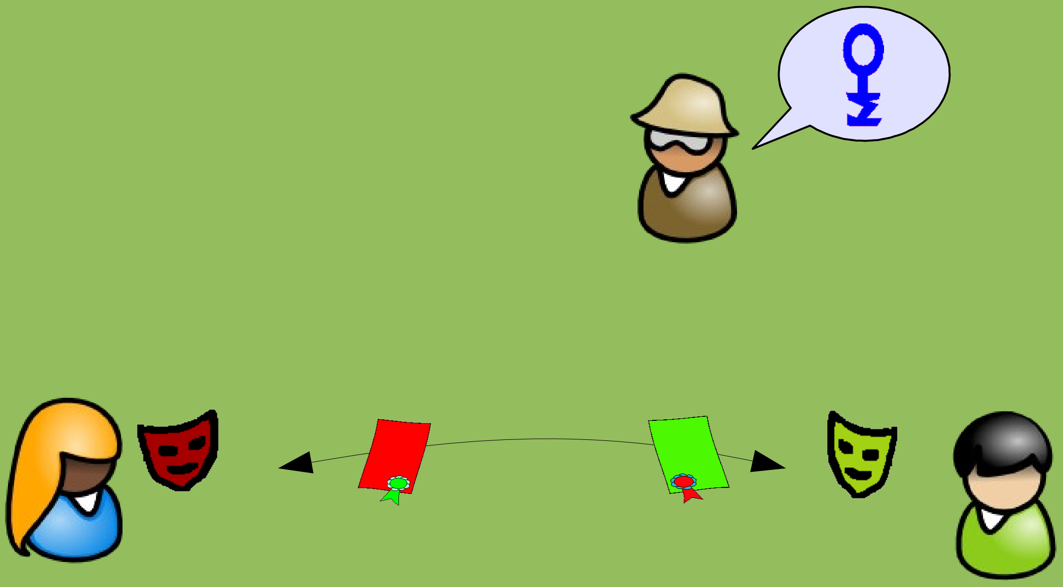
DNA Database

Simple case: DB learns not who accesses DB

Better: Oblivious Access to Database (OT with AC)

- Server must not learn *who* accesses
- *which* record
- Still, Alice can access only records she is *authorized* for

Secret Handshakes [CCGS09]

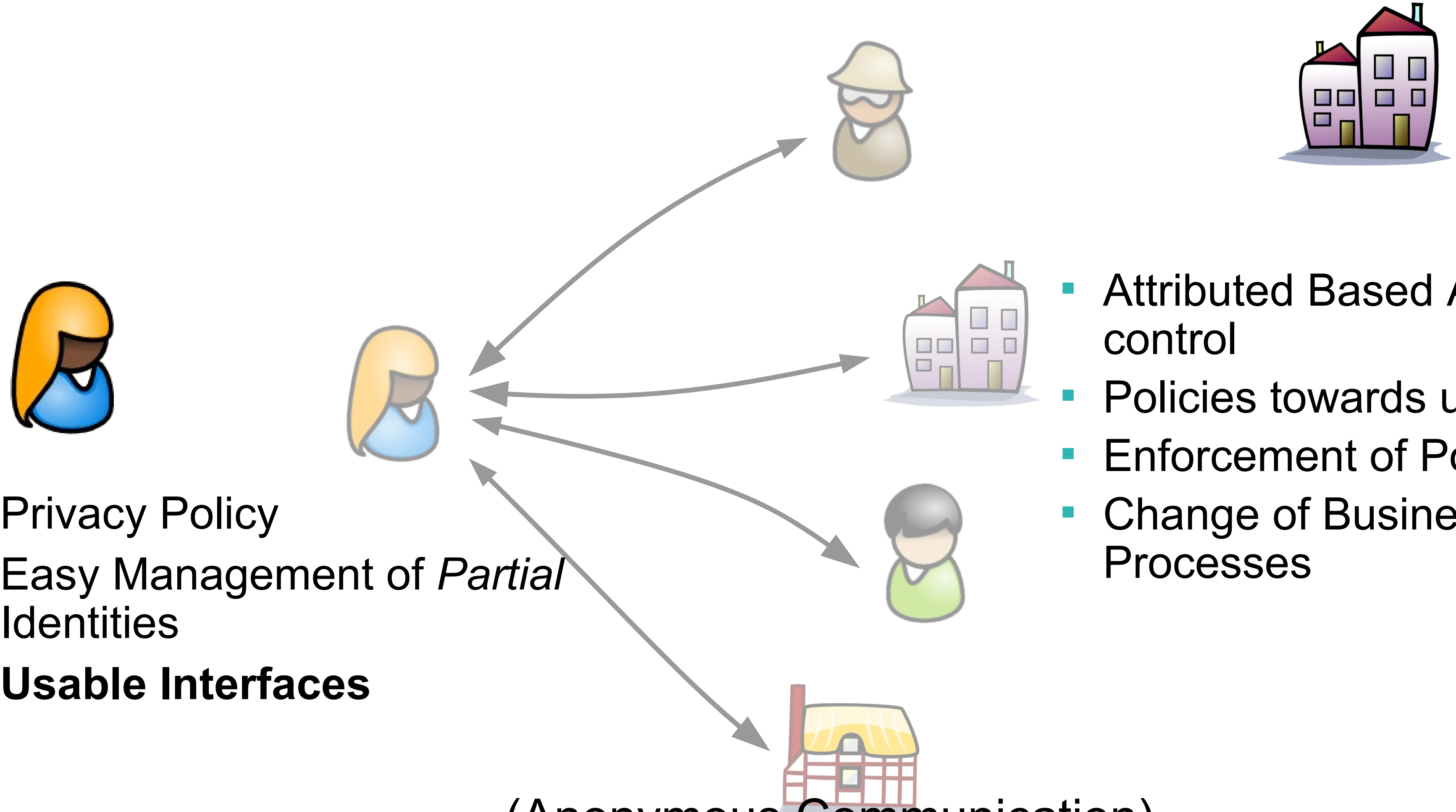


- Alice and Bob both define some predicate PA and PB
- Alice learns whether Bob satisfies PA if she satisfies PB

How to use Crypto PETs

needs more than just crypto.... ;-)

Crypto is the Easiest Part ....

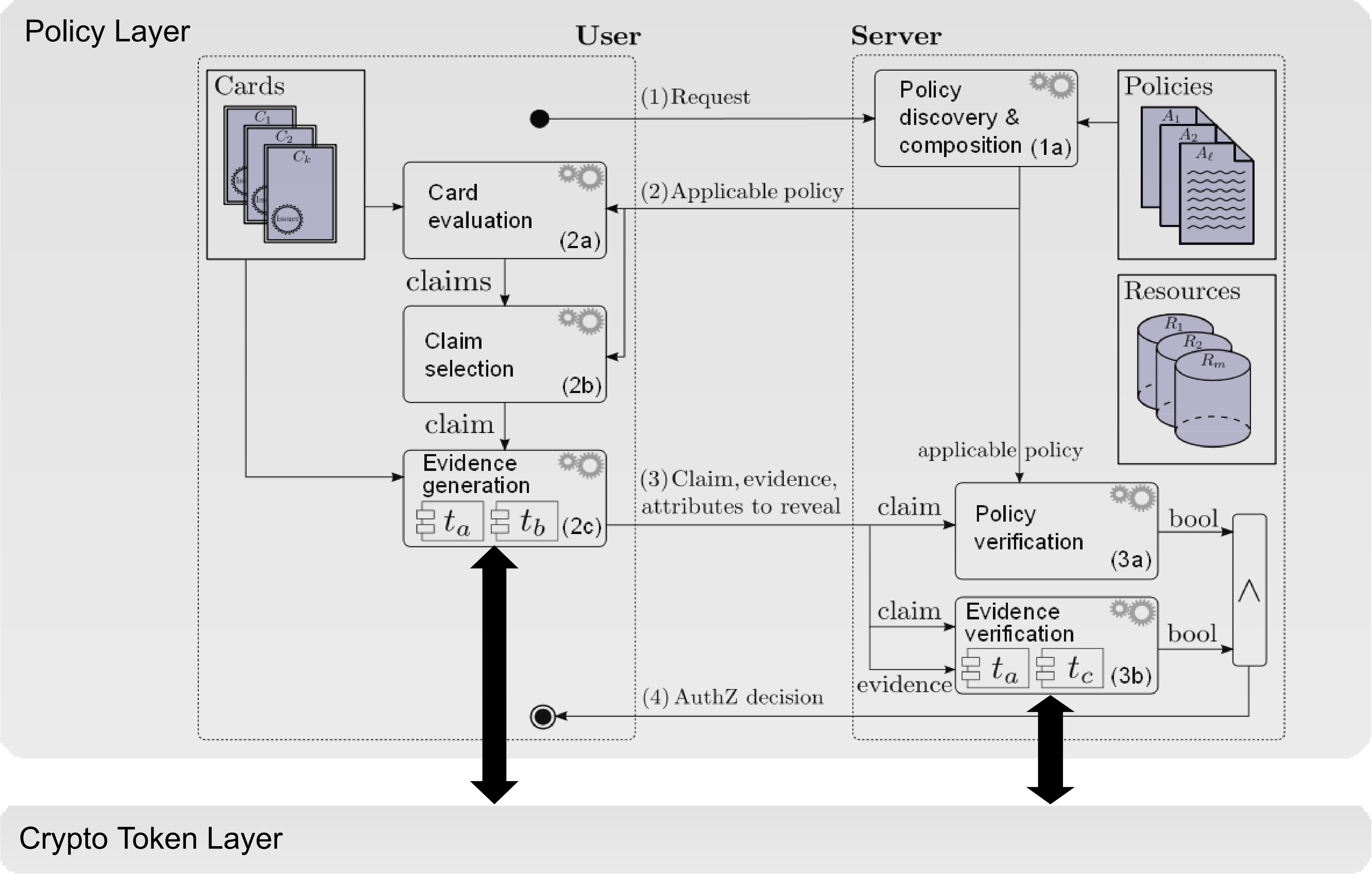


- Privacy Policy
- Easy Management of *Partial* Identities
- **Usable Interfaces**

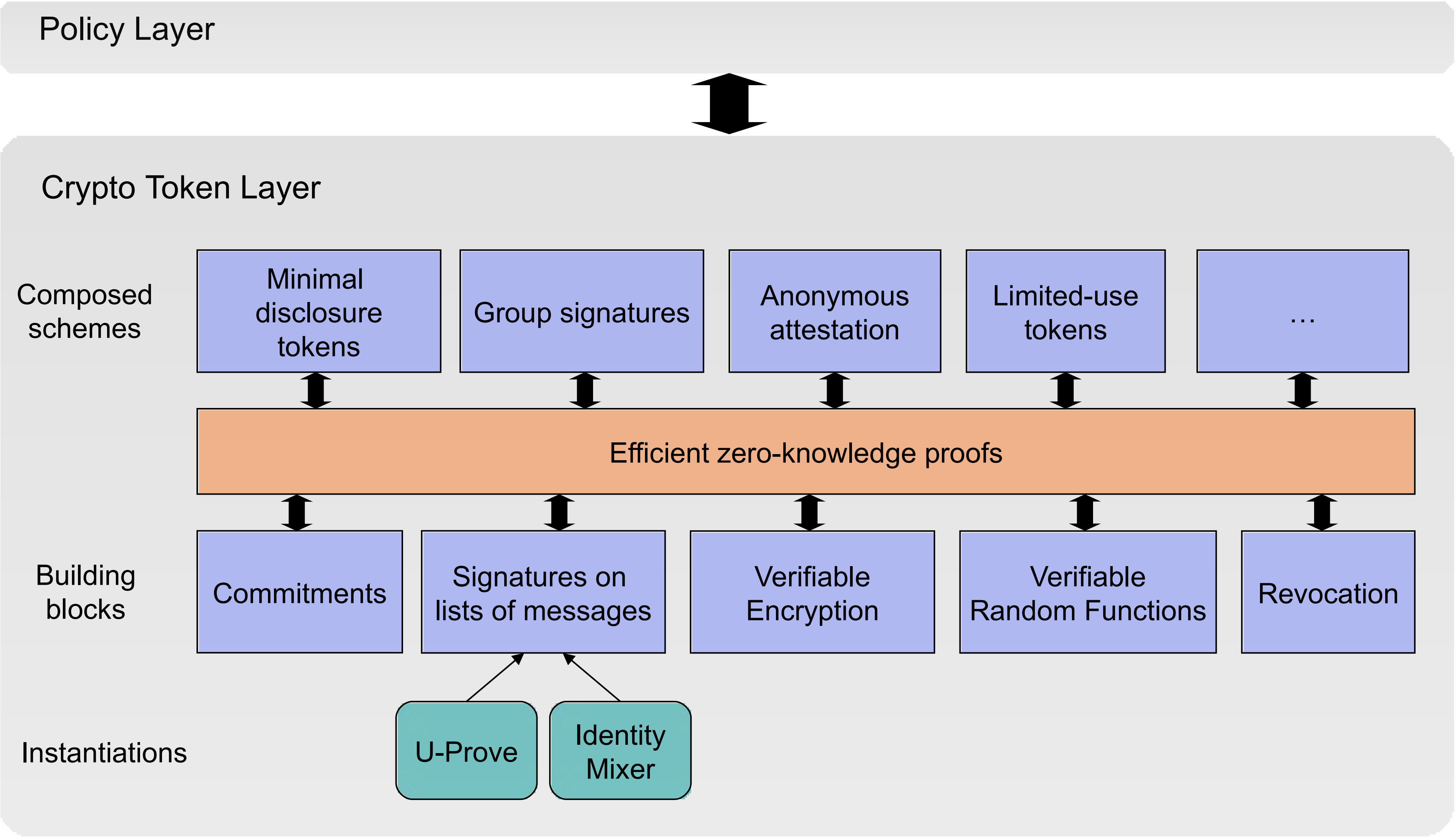
- Attributed Based Access control
- Policies towards users
- Enforcement of Policies
- Change of Business Processes

(Anonymous Communication)

Card-based access control: policy architecture



ABC crypto architecture



Policy Layer

Crypto Token Layer

Composed schemes

Minimal disclosure tokens

Group signatures

Anonymous attestation

Limited-use tokens

…

Efficient zero-knowledge proofs

Building blocks

Commitments

Signatures on lists of messages

Verifiable Encryption

Verifiable Random Functions

Revocation

Instantiations

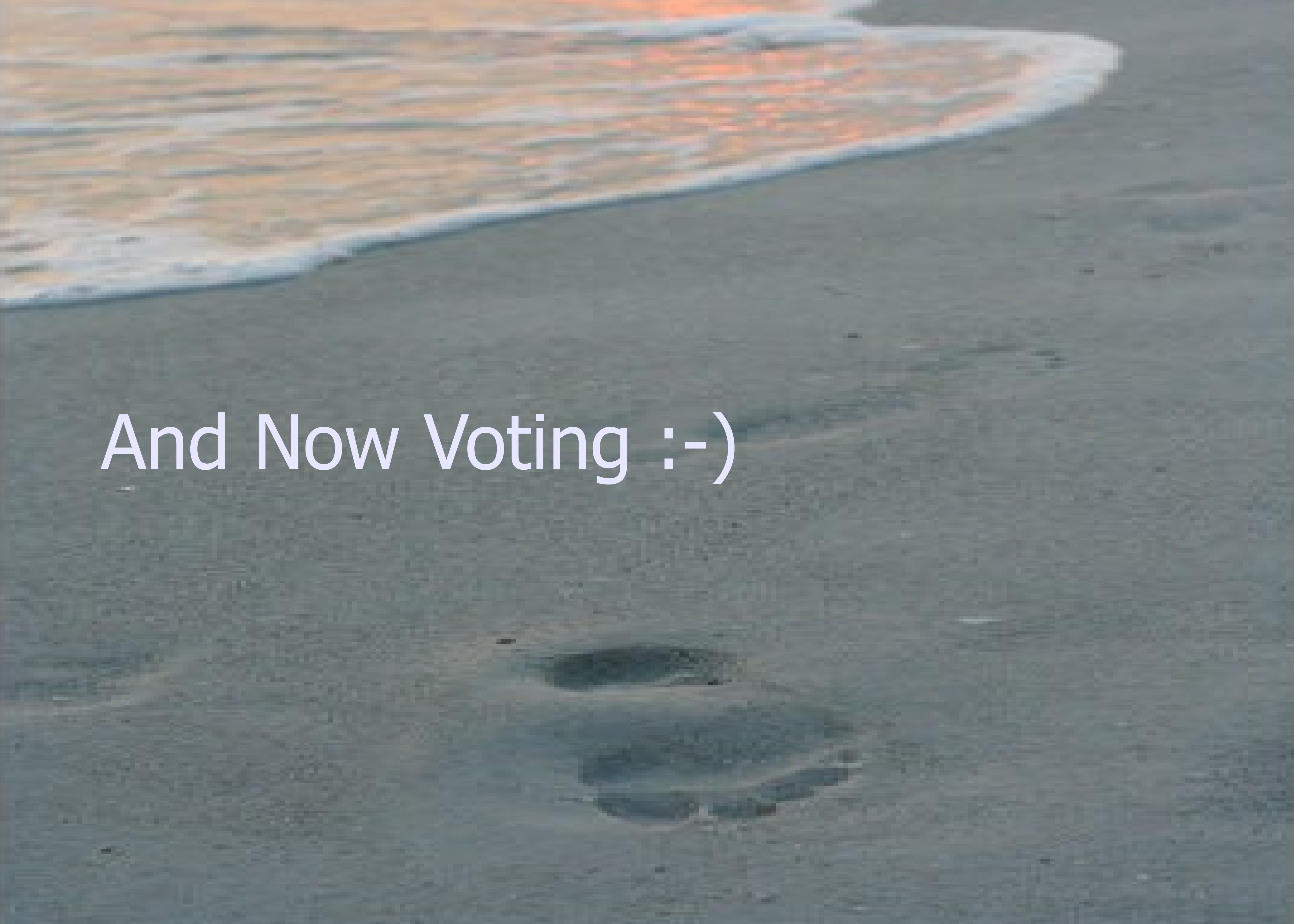
U-Prove

Identity Mixer

Proof Language [BicCam10]

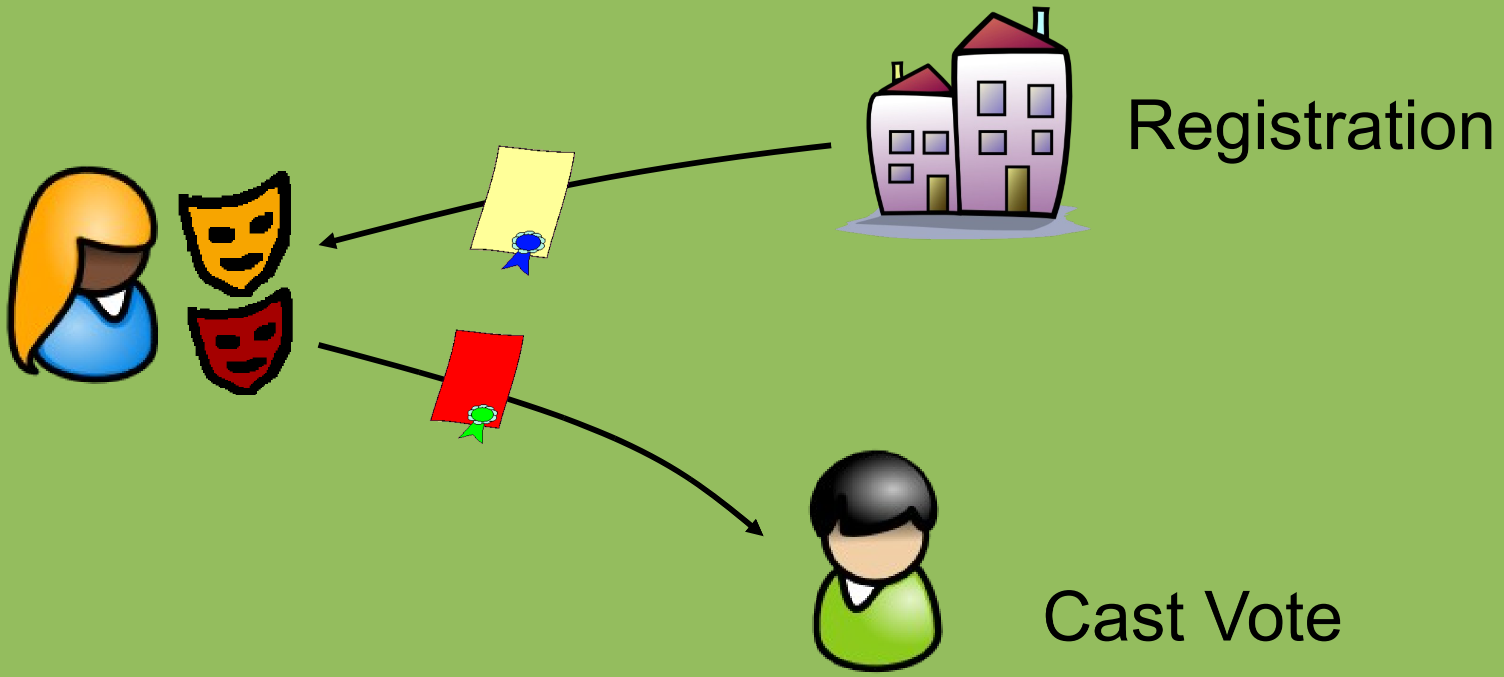
```
Declaration{ id1:unrevealed:string; id2:unrevealed:string; id3:unrevealed:int;
             id4:unrevealed:enum; id5:revealed:string; id6:unrevealed:enum }

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 ..." }
}
```

(see http://www.primelife.eu/results/opensource/55-identity-mixer)



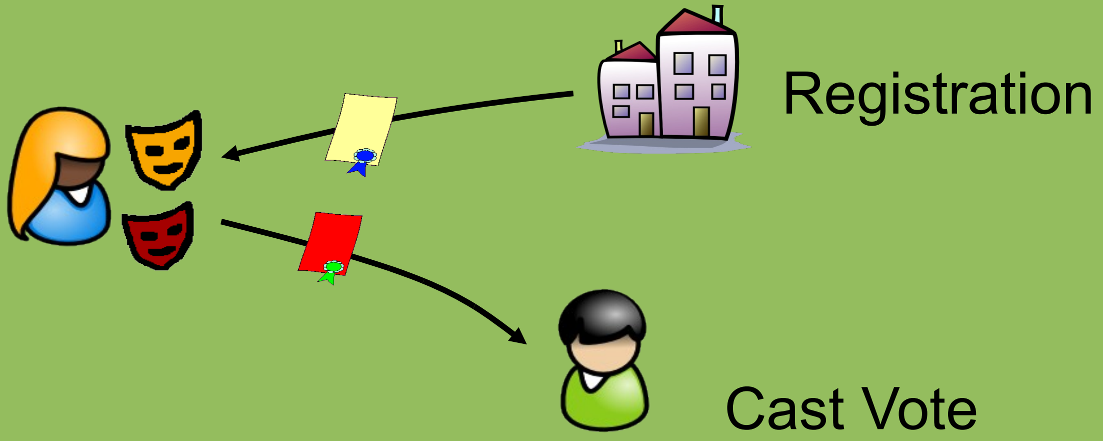
And Now Voting :-)

Voting-Basic Approach



- Register once (could be your eID card)
- Vote: prove that you have registered
- Problem: malicious people could vote several times! ??

Voting - Refined



Solution: prevent malicious people from voting several times!

- Generate *domain* pseudonym for each vote
  - Based on master secret key and domain
  - Thus they are unique for each domain
- Vote: Prove two things
  - Possession of registration credential and
  - Correctness of domain pseudonym

Essentially as blind-signatures approach with reusable registration

Conclusions

Showed you only some of the tools

- More signature schemes (DL, ECC, ...)
- Encryption schemes
- ......

Lots of cool crypto that is about to make it into practice :-)

- See Primelife.eu/results/opensource

Loads of Open Problems

- Still lots of new crypto
- Framework of crypto tools
- User interfaces, standards, ...
- Explain the crypto so that others understand what it's good for