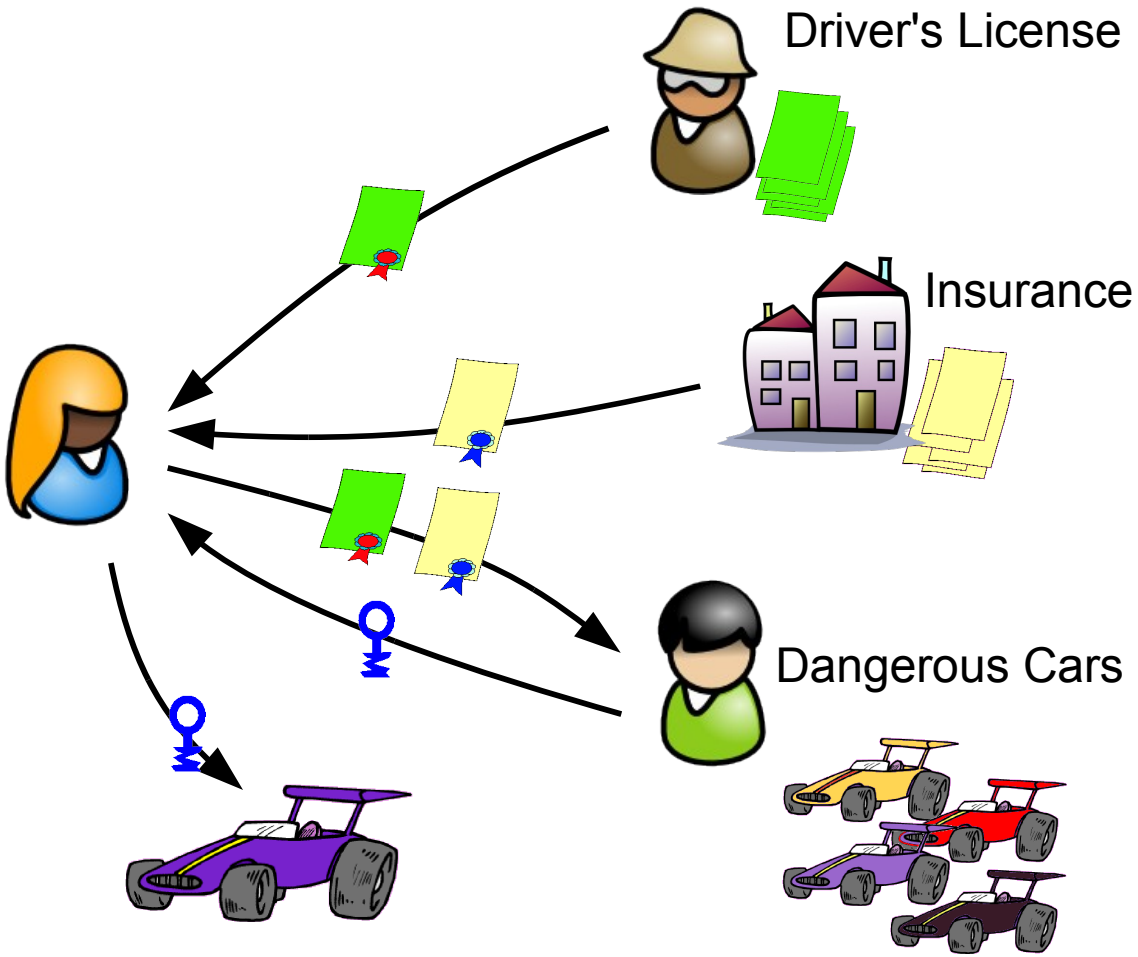
# Privacy by Crypto II

Jan Camenisch & Anna Lysyanskaya

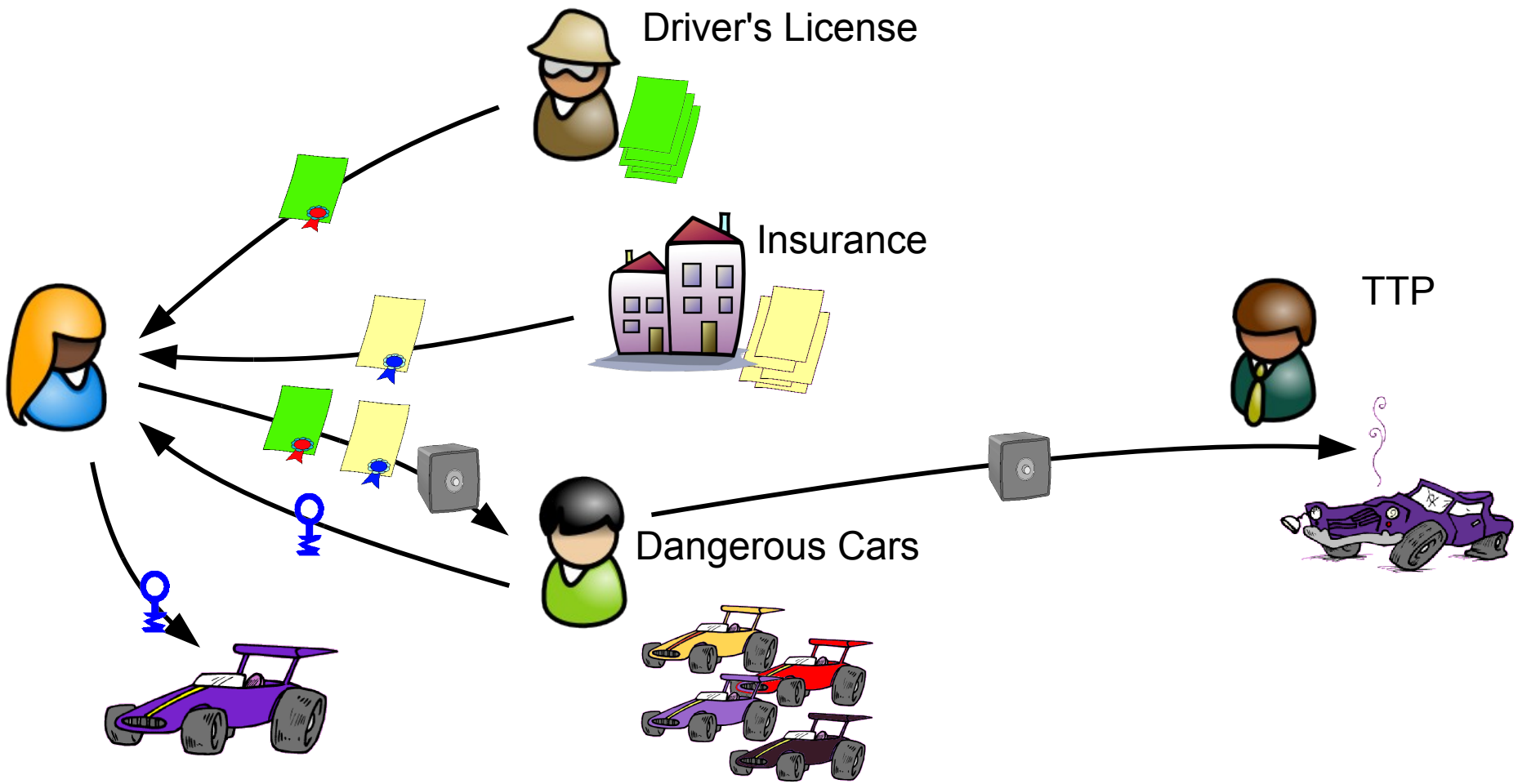
Joint work with many others :-)

# Anonymous Credential Systems

[Chaum85]



# Anonymous Credential Systems



# Extensions (will not be discussed)

- revocation of credentials
- encoding of attributes & selective disclosure
- k-spendable credentials

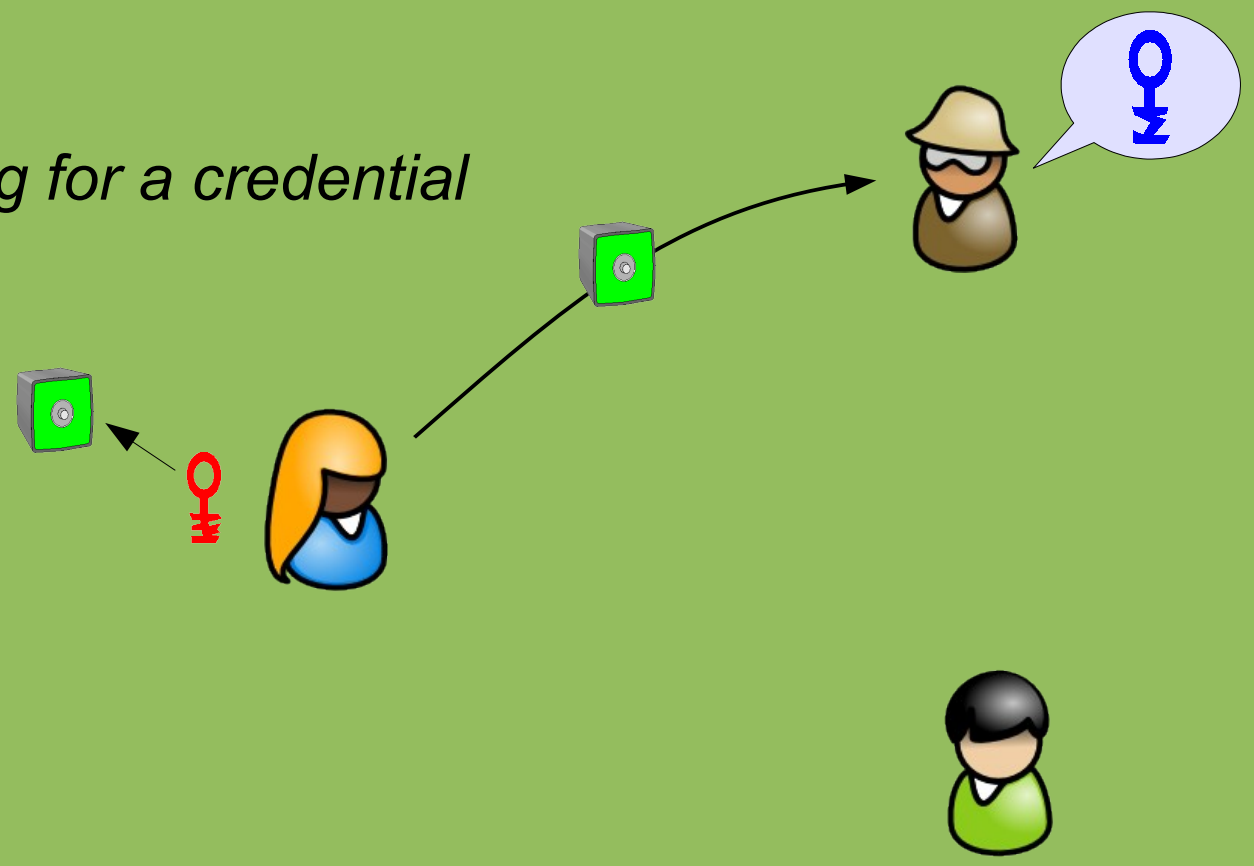
- money laundering
- clone/sharing – detection & prevention
  - cf Direct Anonymous Attestation

# State of the Art: How to Build Them

*In the beginning...*

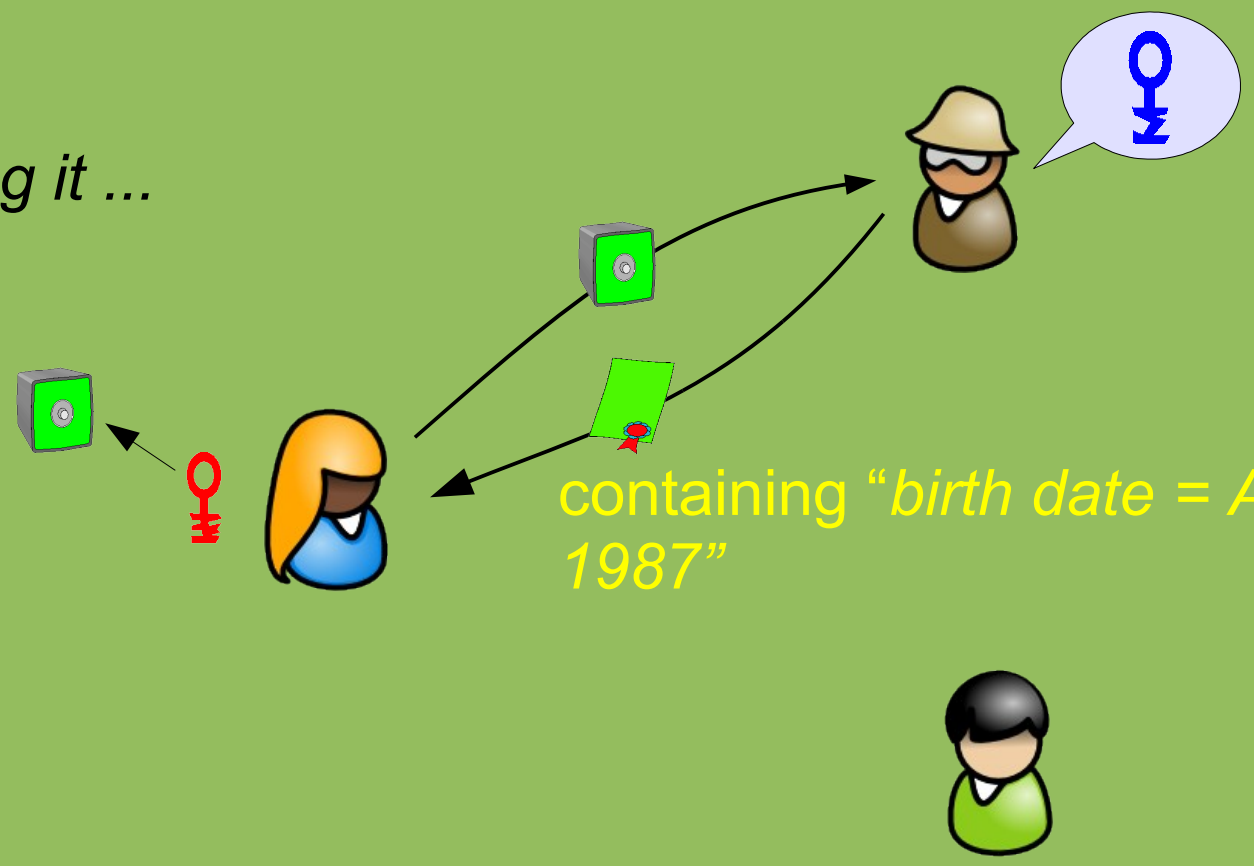
# State of the Art: How to Build Them

*asking for a credential*



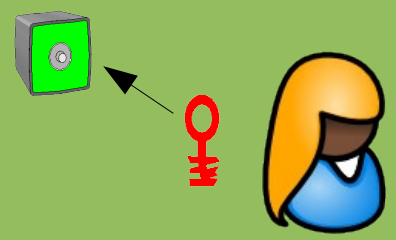
# State of the Art: How to Build Them

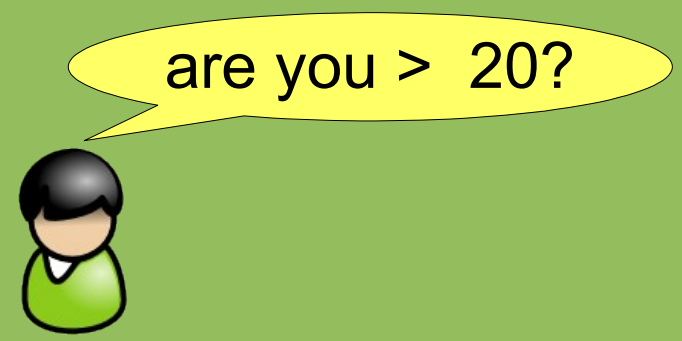
*getting it ...*



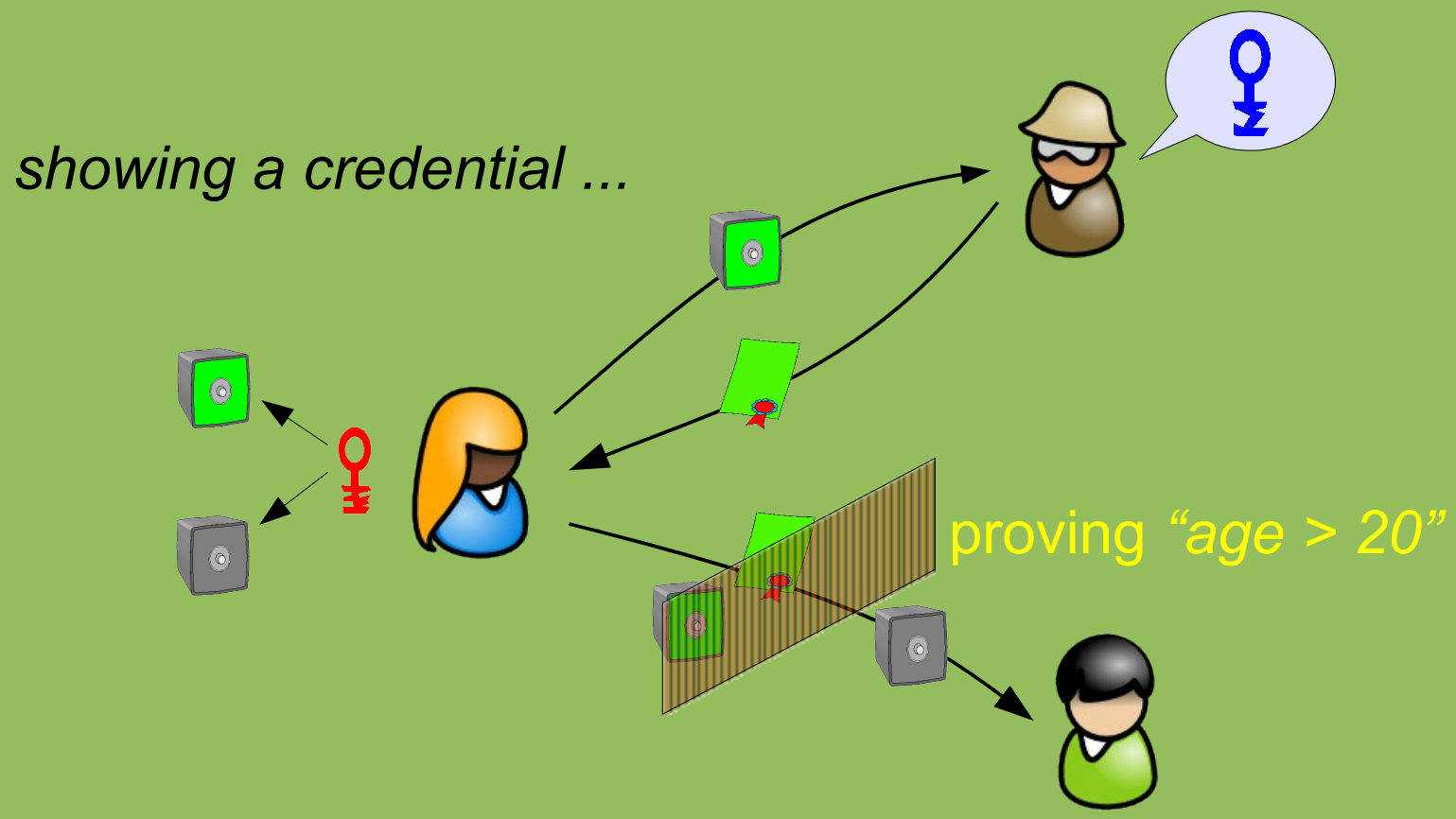
# State of the Art: How to Build Them

*showing it...*



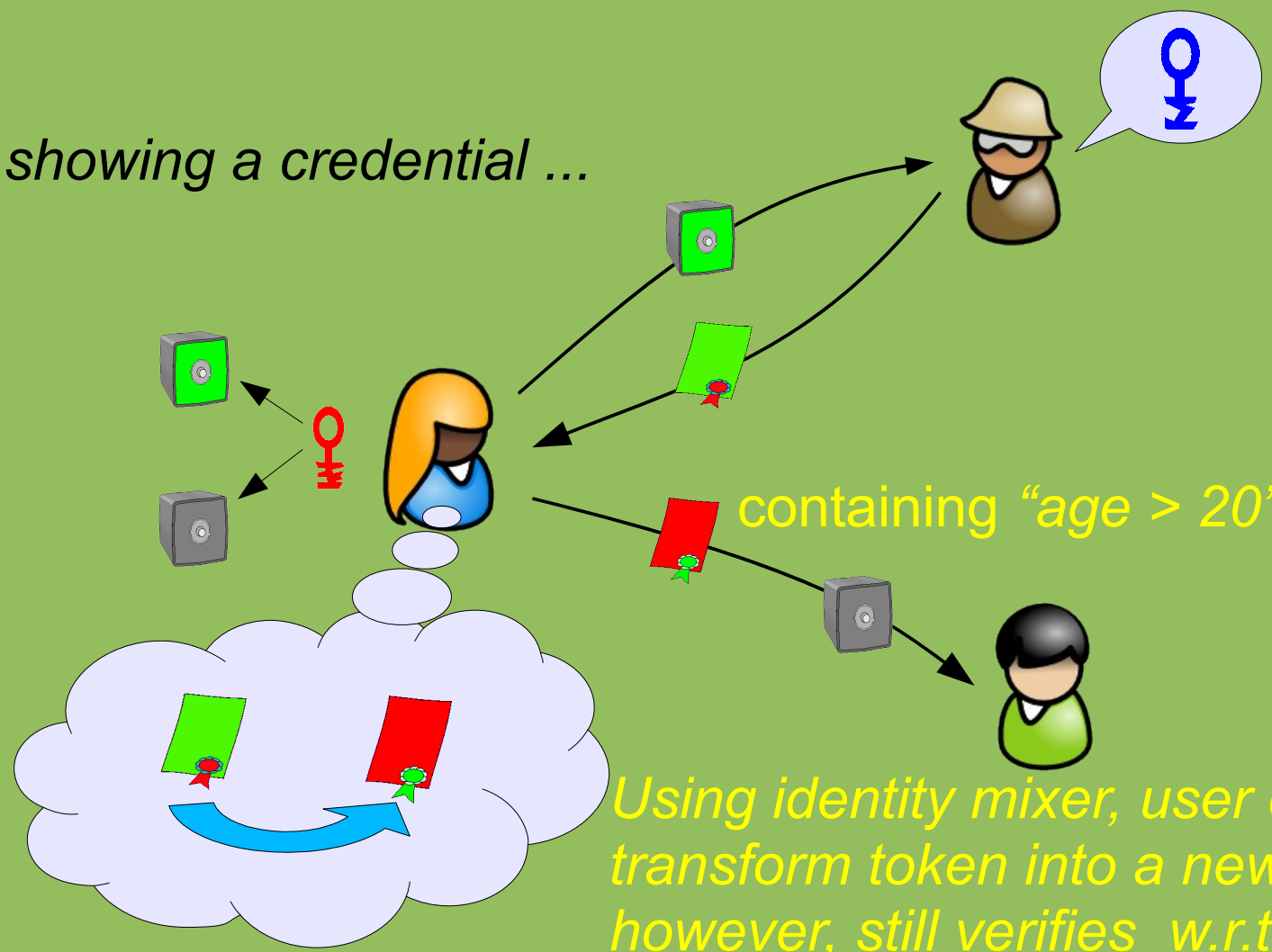


# State of the Art: How to Build Them



# State of the Art: How to Build Them

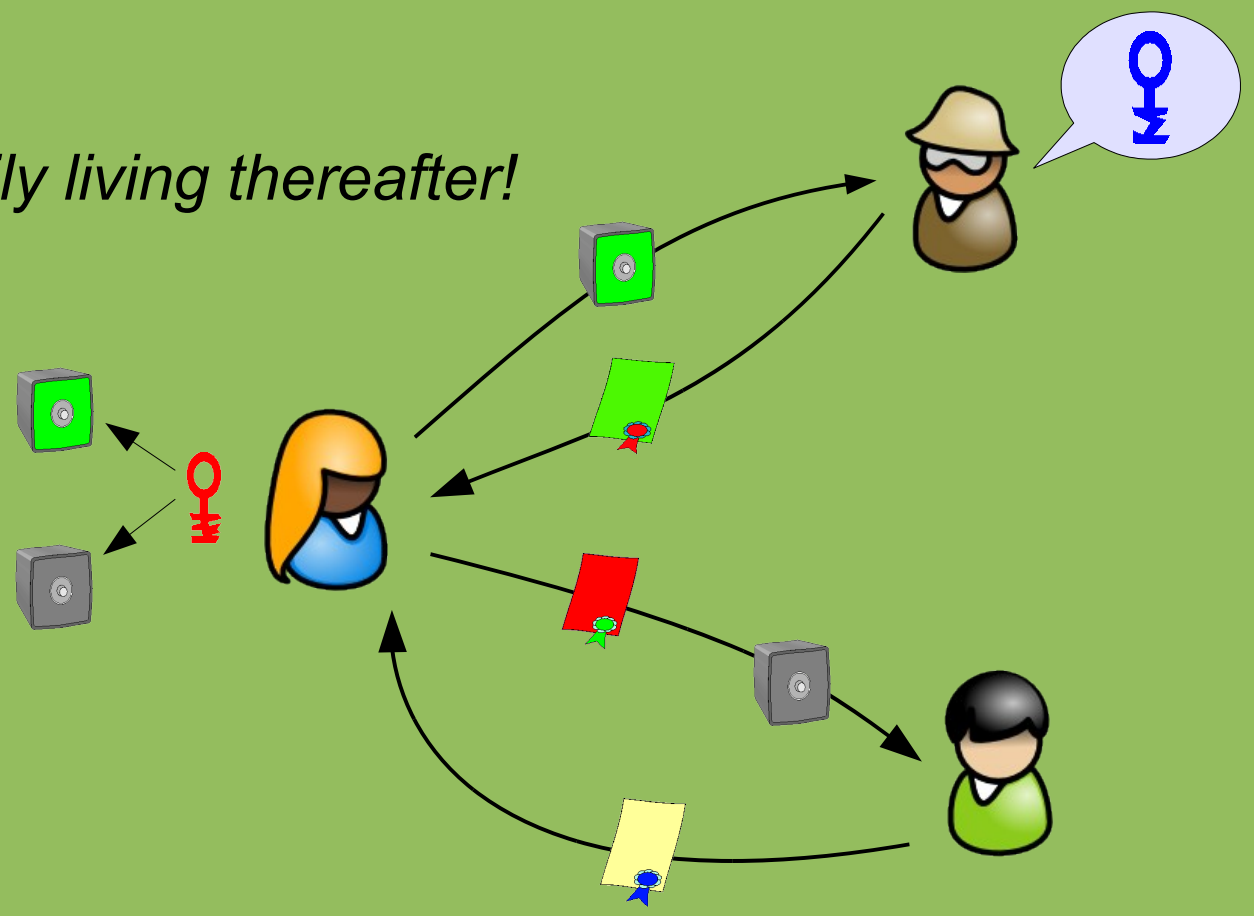
*showing a credential ...*




*Using identity mixer, user can transform token into a new one that, however, still verifies w.r.t. original signer's public key*

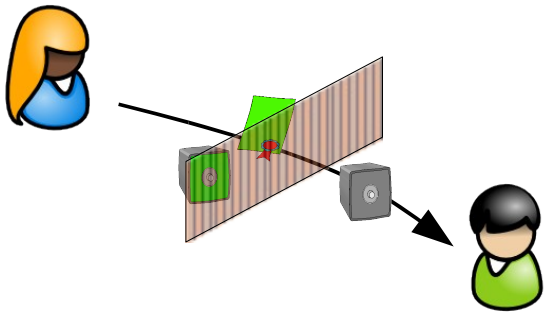
# State of the Art: How to Build Them

*happily living thereafter!*



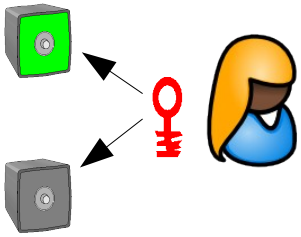
# Properties to be Fulfilled

- Protection of user's privacy
  - anonymity
  - unlinkeability (multi-use)



- Unforgeability of credentials

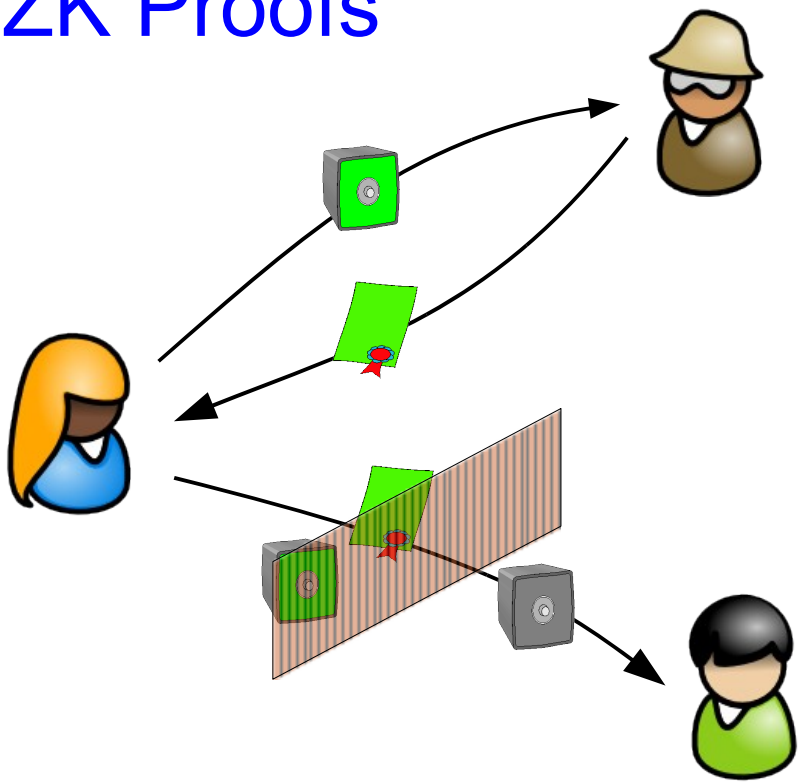
- Consistency of credentials (no pooling)



- Anonymity revocation

# Two Approaches:

## ZK Proofs

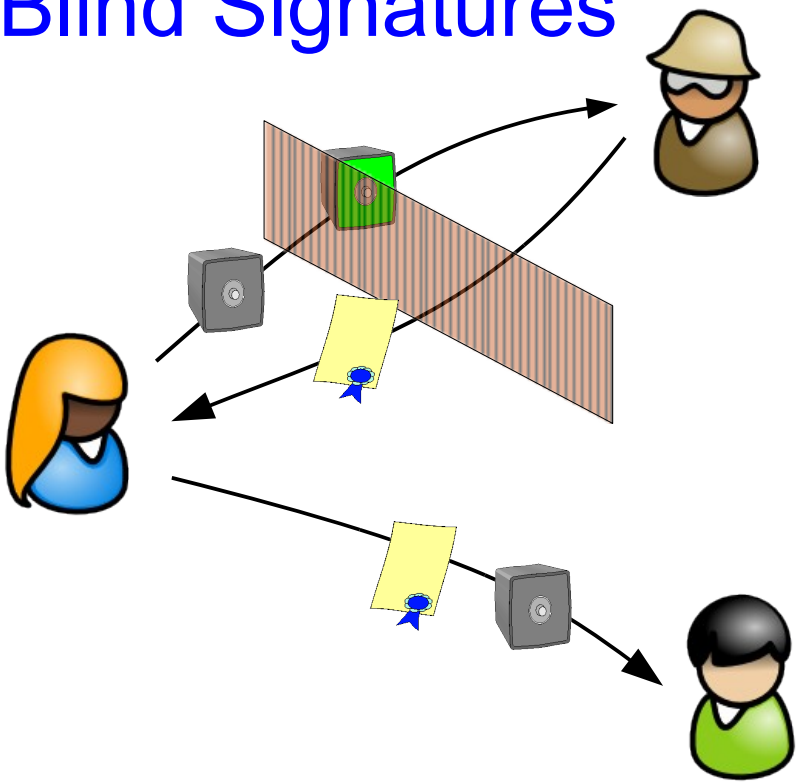


*can be used multiple times*

CL-Signatures

Strong RSA, DL-ECC

## Blind Signatures



*can be used only once*

Brands, et al.

Discrete Logs, RSA

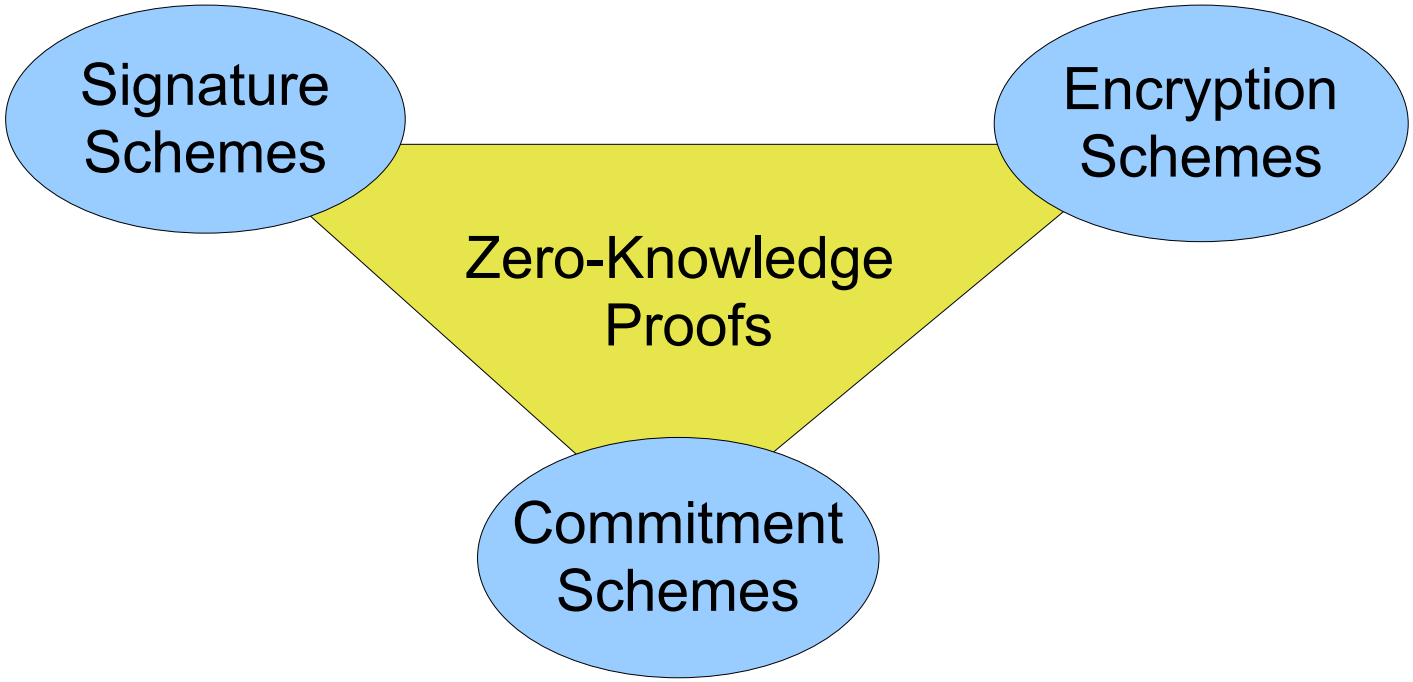
# Some History

- Chaum ’85: introduced scenario
- Chaum & Evertse ’87: solution based on a semi-trusted party
- Damgård ’90: theoretical solution
- Brands ’95-’99: one-show credentials with different attributes
- LRSW ’99: practical solution for one-show credentials
- Camenisch-Lysyanskaya '00: efficient multi-show w/ attributes
- Camenisch-Lysyanskaya '04: Discrete log based.

Special cases: e-cash, group signatures, identity escrow

# How to Really Build Them

# Required Technologies




..... challenge is to do all this efficiently!

# A Quick Number Theory Reminder

# Number Theory: Discrete Logarithm

Group G

- $G = \langle g \rangle = \{1 = g^0, g^1 \ldots, g^{q-1}\}$
- $q$: order (so we are computing mod q in exponent)
- $g$: generator

- Given $x$ it is easy to compute: $y = g^x$, $z = g^{1/x}$, ....

Discrete Log Assumption:

- Given $y$ it is hard to compute $x$ s.t. $y = g^x$

Diffie-Hellman Assumption:

- Given $y, g, h$ it is hard to compute $z = h^x$ where $x = \log_g y$

Decisional Diffie-Hellman Assumption

- Given $g, y, h, z$ (with $y = g^{x1}$ and $z = h^{x2}$) it is hard to decide if $x1 = x2$

# Number Theory: RSA & Factoring

Computing $n = pq$ with $p$ and $q$ primes is easy

Given $n$, finding $p$ and $q$ s.t. $n = pq$ is hard (Factoring Assumption)

Consider group $Zn^* = \{i : 0<i<n, \gcd(i,n)=1\}$

RSA: given $n$ and $e$ computing

- $y = g^e \bmod n$ is easy
- $y = g^{1/e} \bmod n$ is hard (RSA Assumption)

Strong RSA Assumption: given $n$ and $z$ finding

- $u,e$ s.t. $z = u^e \bmod n$ is hard

# Zero-Knowledge Protocols

# Zero-Knowledge Proofs of Knowledge

[Schnorr '91,Chaum & Pedersen '92,....]

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

Prove *knowledge* of $x = \log_g y$ such that verifier only learns $y$ and $g$.

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

| Prover: | | Verifier: |
|---|---|---|
| random $r$ | | |
| $t := g^r$ | $\xrightarrow{\quad t \quad}$ | |
| | | random $c$ |
| | $\xleftarrow{\quad c \quad}$ | |
| $s := r - cx$ | $\xrightarrow{\quad s \quad}$ | |
| | | $t = g^s y^c$ |

# Zero-Knowledge Proofs of Knowledge

PK{($x$): $y = g^x$ }

***Proof of Knowledge:***

From any prover that can make verifier accept one can extract value $x$ such that $y = g^x$ .

***Zero Knowledge:***

For any verifier there exists a simulator-prover that can not privy to $x$ such that verifier accepts conversation.

# Zero-Knowledge Proof-Protocols

Non-interactive (Fiat-Shamir heuristic):

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

Logical combinations:

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

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

Intervals and groups of different order (under SRSA):

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

$$PK\{(\alpha):\ y = g^{\alpha} \wedge \mathcal{Z} = \mathcal{g}^{\alpha} \wedge \alpha \in [0,\min\{ord(g),ord(\mathcal{g})\}]\}$$

# Example Zero-Knowledge Proof

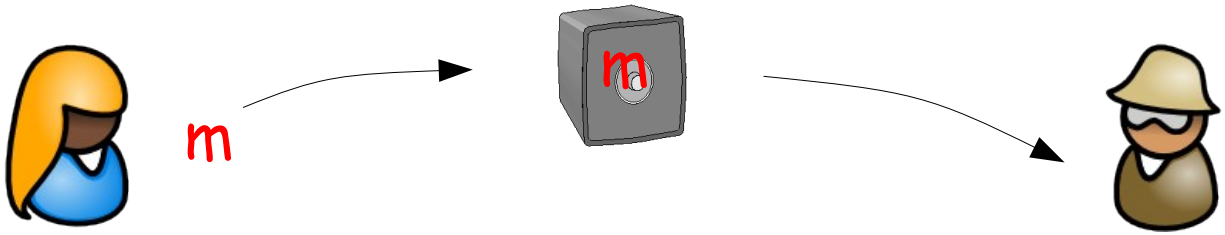
[Schnorr '91,Chaum & Pedersen '92,....]

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

| Prover: | | Verifier: |
|---|---|---|
| random $r_\alpha$, $r_\beta$ | | |
| $t_y := g^{r_\alpha}$, $t_z = g^{r_\beta}$ | $\xrightarrow{t_y,\ t_z}$ | |
| | | random $c$ |
| | $\xleftarrow{c}$ | |
| $s_\alpha := r_\alpha - c \cdot \alpha$ | | |
| $s_\beta := r_\beta - c \cdot \beta$ | $\xrightarrow{s_\alpha,\ s_\beta}$ | $t_y = g^{s_\alpha} y^c$ |
| | | $t_z = g^{s_\beta} z^c$ |

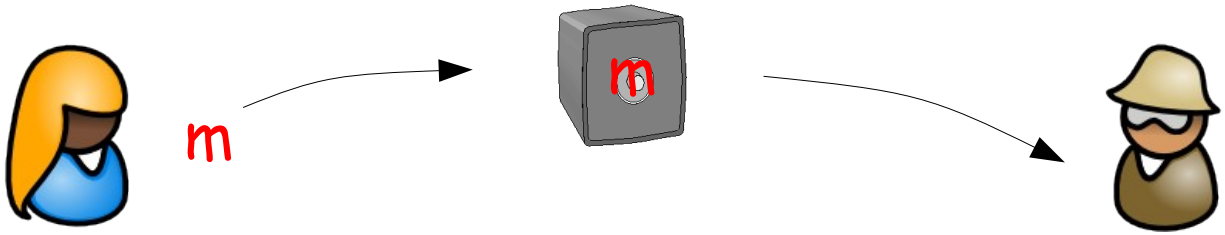
# Commitment Scheme

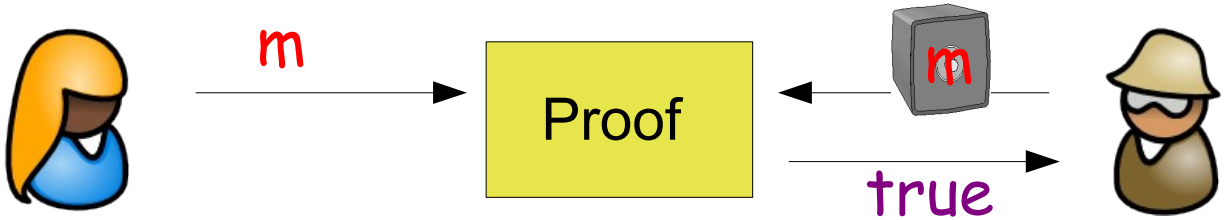
# Commitment Scheme



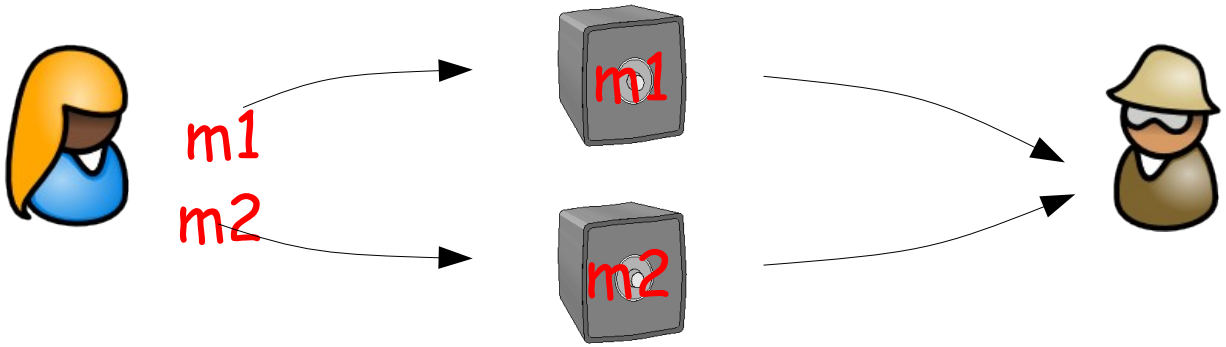

At some later point...

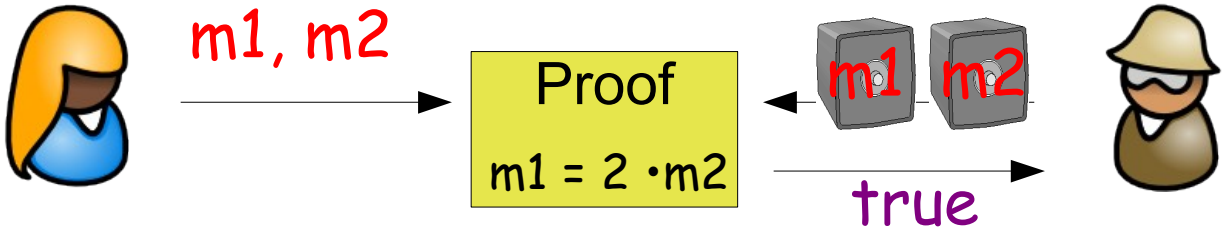
# Commitment Scheme



# Commitment Scheme



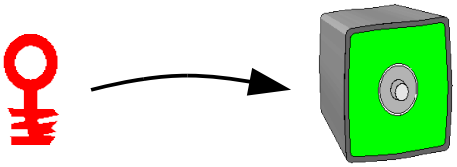
# Commitment Schemes

[Pedersen]

Group $G = \langle g \rangle = \langle h \rangle$ of order $q$

To commit to element $x \in Z_q$:

- choose $r \in Z_q$ and
- compute $c = g^x h^r$



To commit to integer $x \in Z$ (Damgård, Fujisaki):

- similarly, if order of $G$ is not known, e.g., subgroup of the integers modulo an RSA modulus $n$, e.g., $G = QR_n$
- choose $r \in \{1,\ldots,n2^k\}$ and compute $c = g^x h^r$

# Signature Scheme

# 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]

# Signature Scheme based on the SRSA Assumption I

[Camenisch & Lysyanskaya '01]

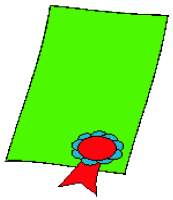
Public key of signer: RSA modulus $n$ and $a_i, b, d \in QR_n$,

Secret key: factors of $n$

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

- choose random prime $e > 2^\ell$ and integer $s \approx n$
- compute $c$ such that

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



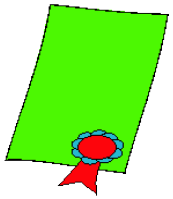
- signature is $(c,e,s)$

# Signature Scheme based on the SRSA Assumption II

[Camenisch & Lysyanskaya '01]

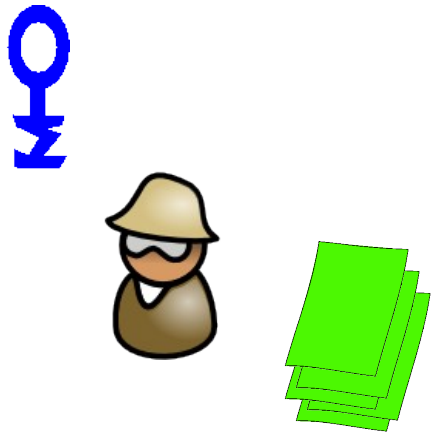
A signature $(c,e,s)$ on messages $m1, \ldots, mk$ is valid iff:

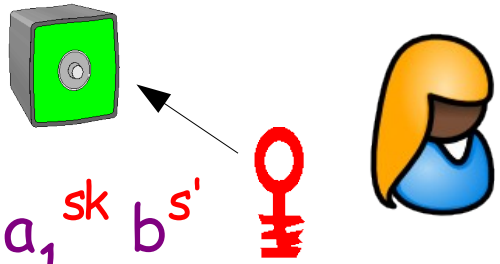
- $m1, \ldots, mk \in \{0,1\}^{\ell}$:
- $e > 2^{\ell}$
- $d = a_1^{m1} \cdot \ldots \cdot a_k^{mk} \; b^s \; c^e \bmod n$



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

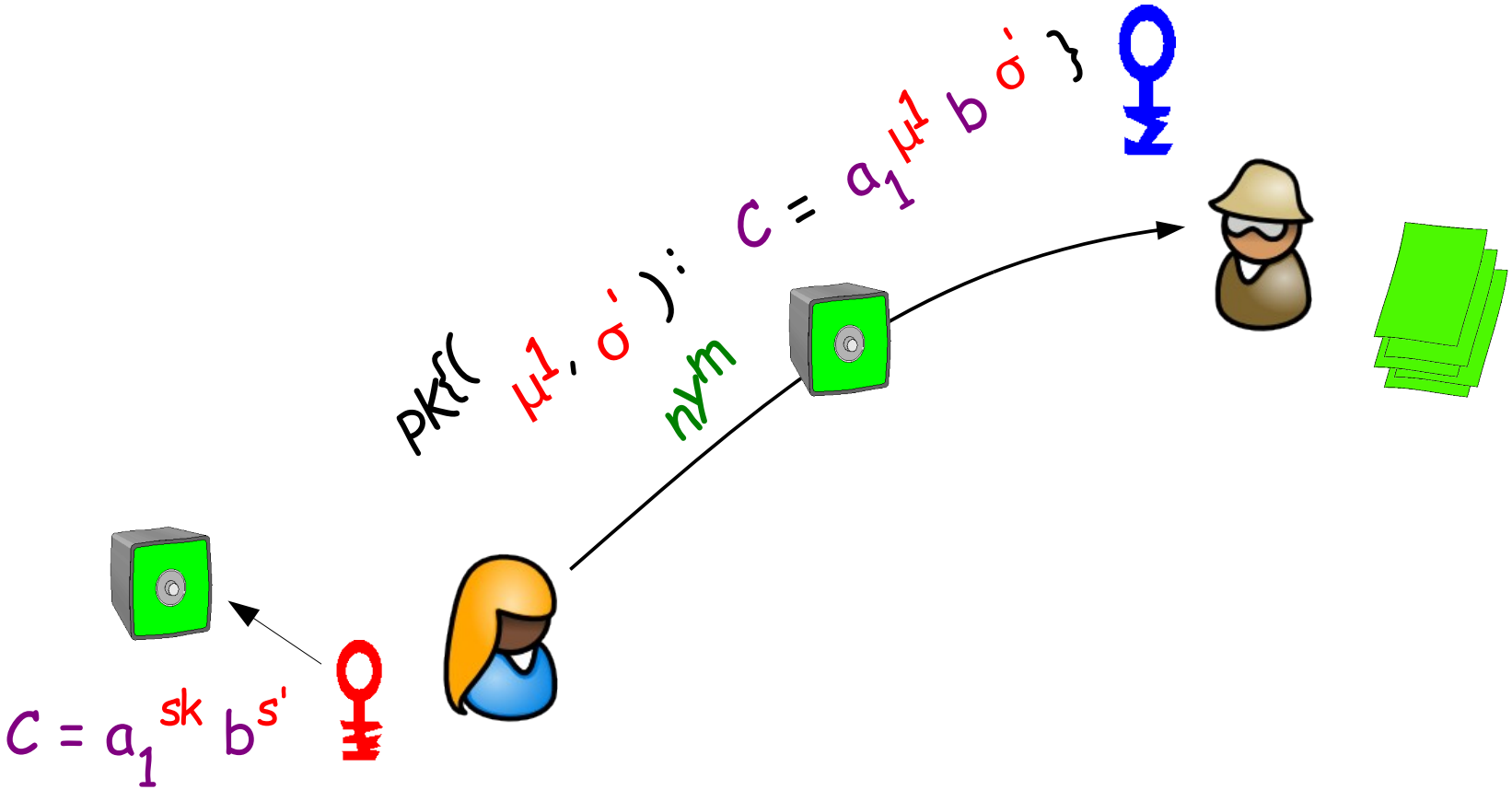
# Getting a Signature on a Secret Message



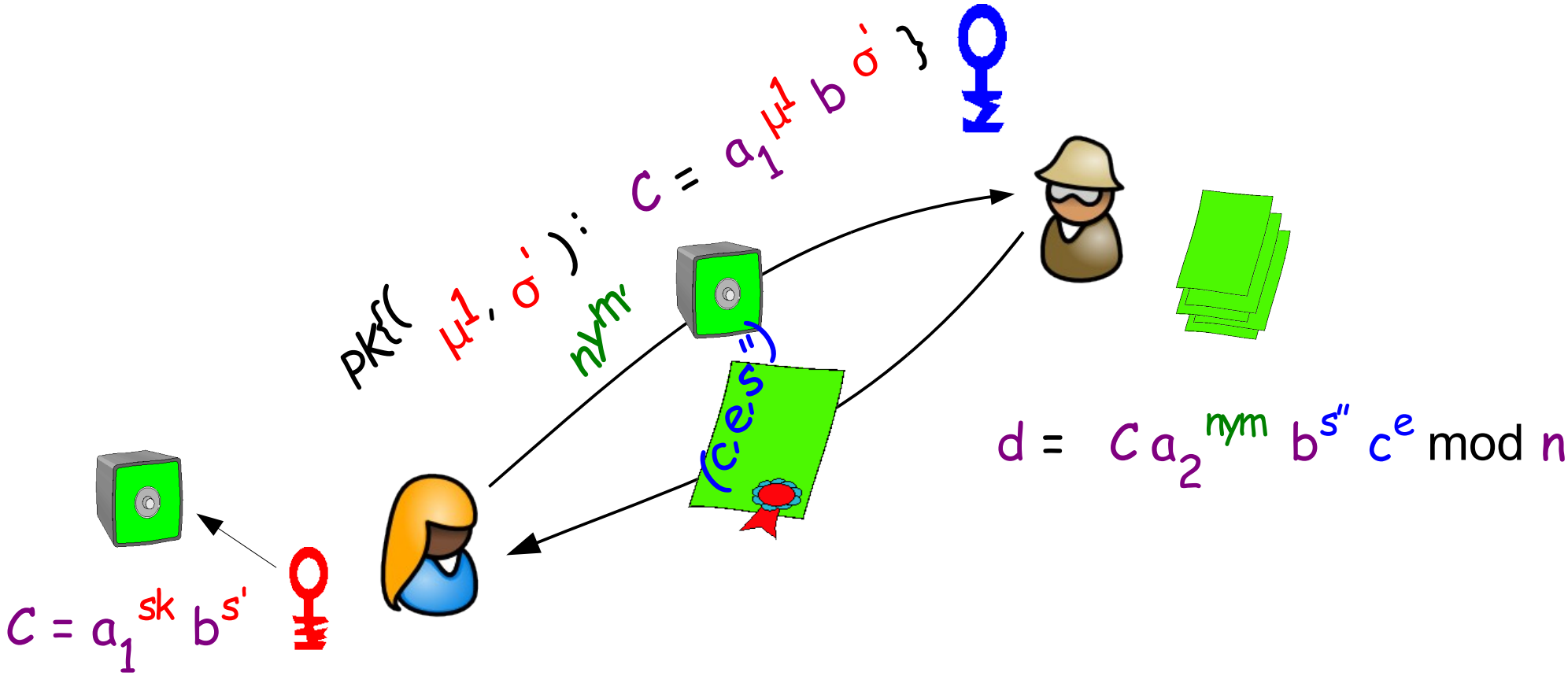


$$C = a_1^{sk}\, b^{s'}$$

# Getting a Signature on a Secret Message

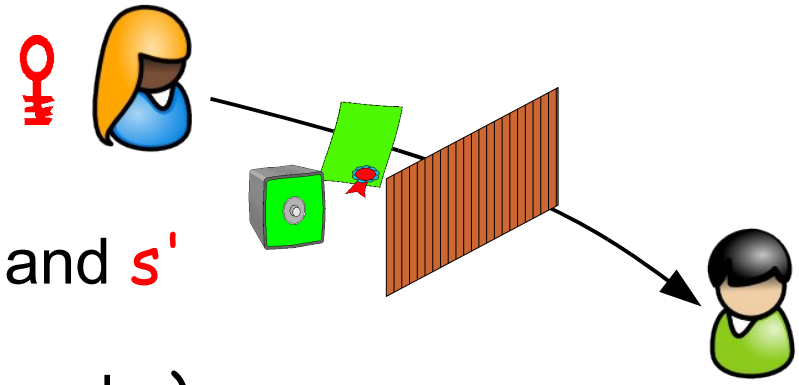


# Getting a Signature on a Secret Message




$$d = a_1^{sk}\, a_2^{nym}\, b^{s''+s'}\, c^e \pmod{n}$$

# Proof of Knowledge of a Signature



Observe:

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

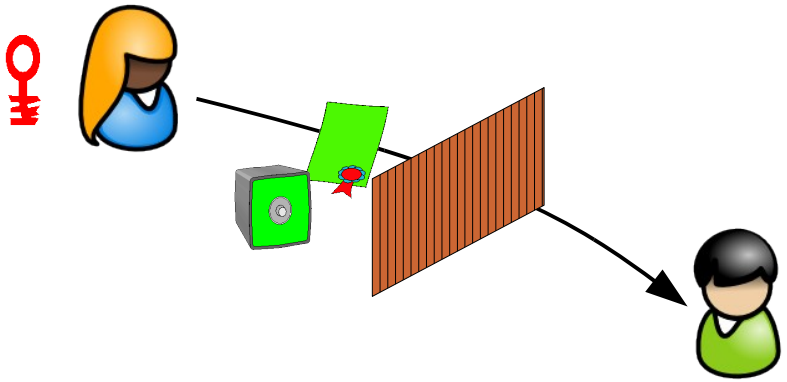
  i.e., $(c', e, s^*)$ is a also a valid signature!

Therefore, to prove knowledge of signature on some $m$

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

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

# Revealing some attributes



Assume we wanna reveal m2 and m3

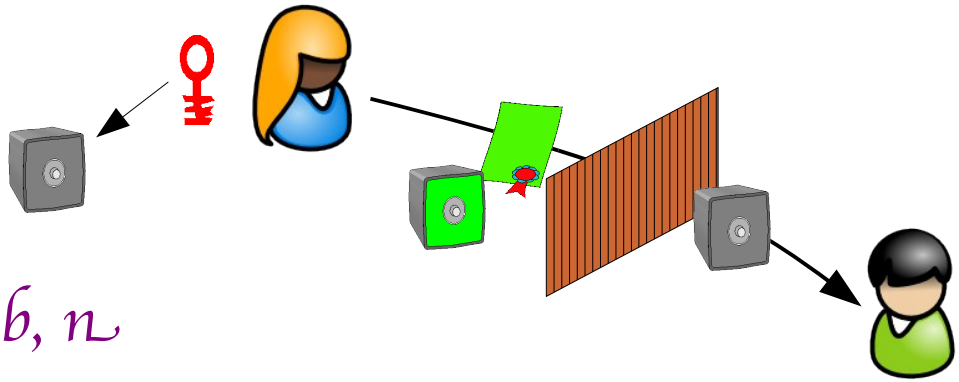
Thus, send m2 and m3 to verifier and proof

- Provide newly randomized c'
- PK{(e, m1, m4,..., mk, s) :

$$d/(a_2^{m2} a_3^{m3}) := c'^{e}\, a_1^{m1}\, a_4^{m4} \cdot \ldots \cdot a_k^{mk}\, b^{s}$$

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

# Proof of Knowledge of a Signature



Using second Commitment

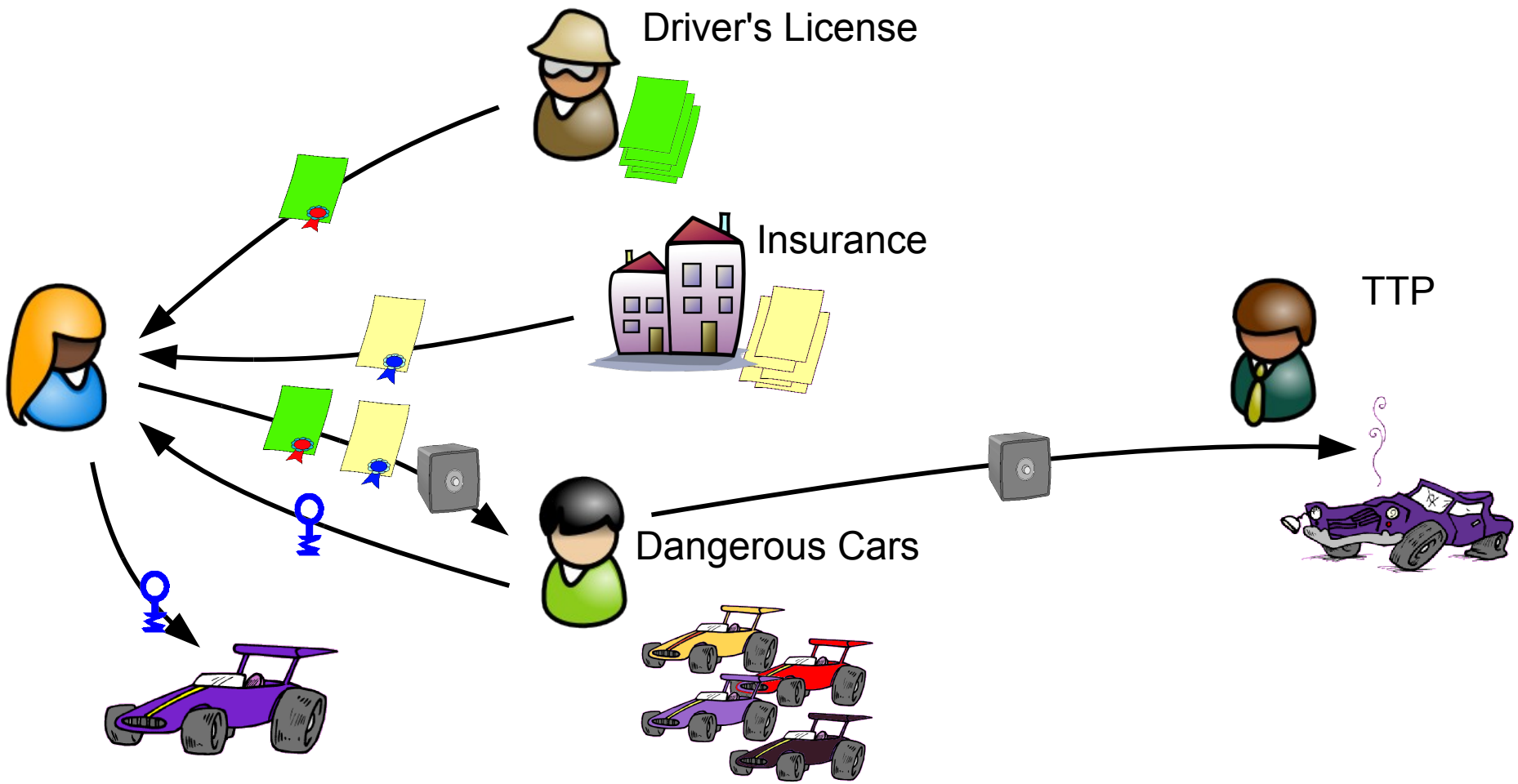
- assume second group $\mathcal{T}n, a_i, b, n\mathcal{L}$
- 2nd public key (commitment) $C = a_1^{sk}\, b^{s*}$

To prove knowledge of signature on some m

- provide c'
- PK{(e, m1, ..., mk, s,s* ) :

$$C = a_1^{m1} b^{s*} \;\wedge\; d := c'^{e}\, a_1^{m1} \cdot \ldots \cdot a_k^{mk}\, b^{s} \;\}$$

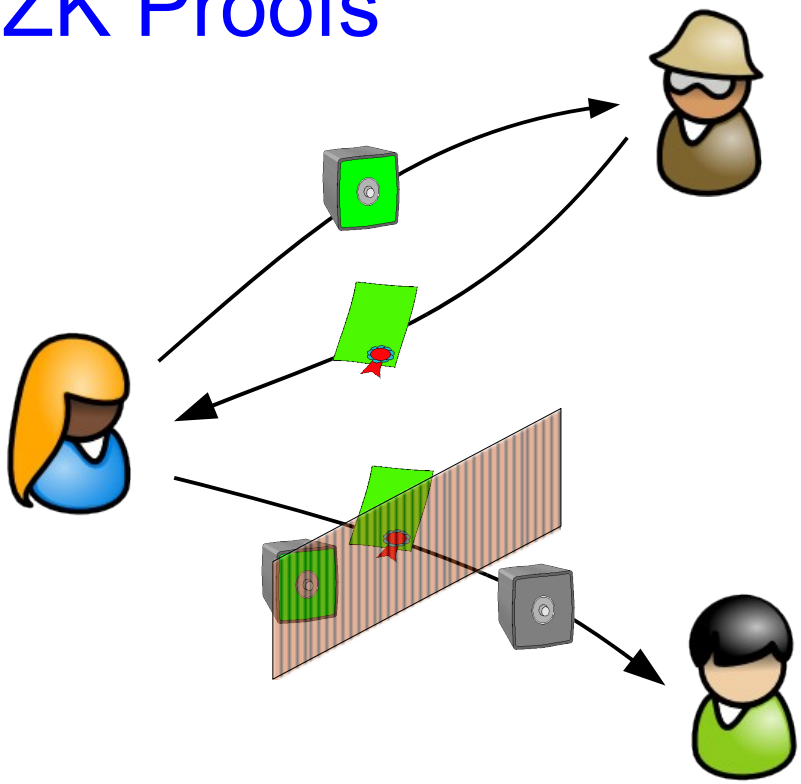
# Now All Pieces 4 Anonymous Credential Systems!



# How U-Prove Works

# Two Approaches:

## ZK Proofs

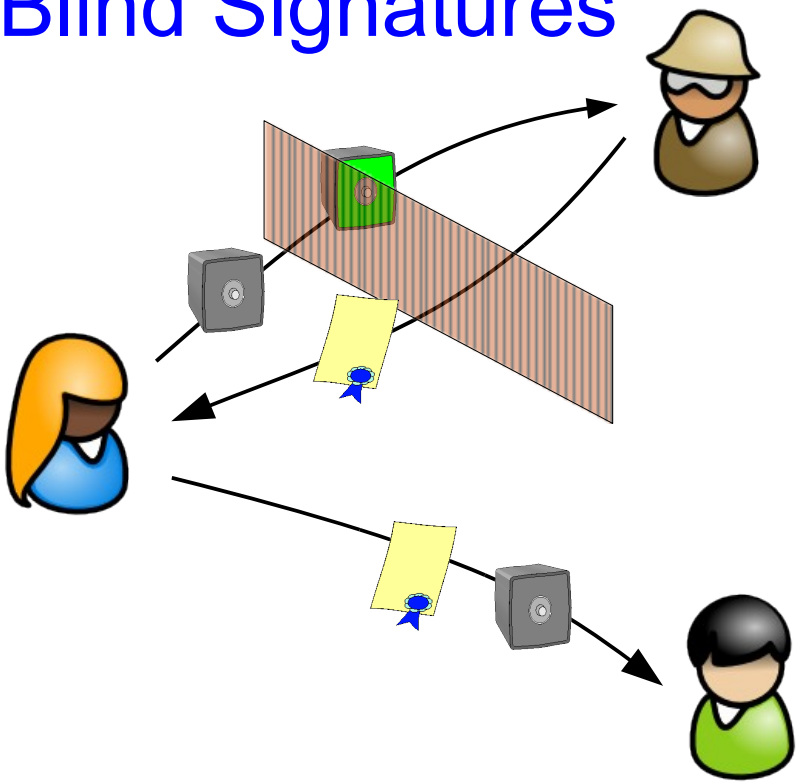


*can be used multiple times*

CL-Signatures

Strong RSA, DL-ECC

## Blind Signatures



*can be used only once*

Brands, et al.

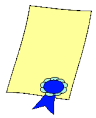
Discrete Logs, RSA

# U-Prove Signature

- Group $G = \langle g \rangle = \langle h \rangle$ of order $q$
- Public Key: $y$ and $z_i$, $g_i$, Secret key $x$ Keys $y = g^x$ and $z_i = g_i^x$
- Signature on messages $m1, \ldots, mk \in Z_q$ (with constant $c$)
  - Random $a \in Z_q$
  - $h = (g_0\, g_1^{m1} \cdot \ldots \cdot g_k^{mk}\, g_{k+1}^{c})^a$ and

    $w = (z_0\, z_1^{m1} \cdot \ldots \cdot z_k^{mk}\, z_{k+1}^{c})^a \qquad (= h^x)$
  - $\pi = SPK\{(x) : y = g^x \wedge w = h^x\}$

Signature = $(a,h,w,\pi)$



# U-Prove Proving Knowledge Signature

Want to show credentials and reveal m2 and m3

- Compute

  $\pi' = SPK\{(a, m1, m4, \ldots, mk) :$

  $$g_0 g_2^{m2} g_3^{m3} g_{k+1}^{c} = h^{1/a} g_1^{-m1} g_4^{-m4} \cdot \ldots \cdot g_k^{-mk} \}$$

- Send $\pi'$, $\pi$, $h$, $w$ to verifier (who then verifies $\pi'$, $\pi$ )

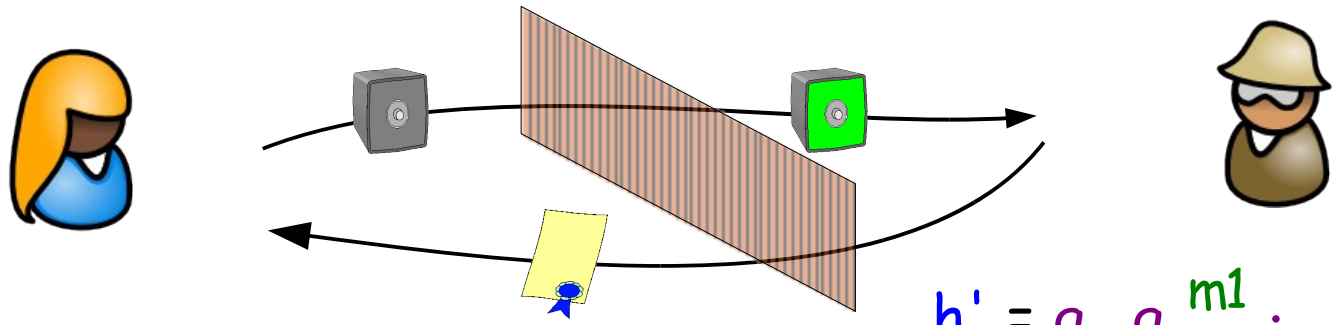
Notice:

- As we send $\pi$ , $h$, $w$ to verifier this can only be done once per signature :-(

- Need get as many sigs as uses, as signature issuing protocol is interactive, so is quite costly

# U-Prove Signature Issuing Protocol

$y = g^x$



$h' = g_0 \, g_1^{m1} \cdot \ldots \cdot g_k^{mk} \, g_{k+1}^{c}$

$w' = h^x$

Choose a

$h = h'^a$

$PK\{(x) : y = g^x \wedge w' = h'^x\}$

$\pi = SPK\{(x) : y = g^x \wedge w = h^x\}$

# If You Wanna Really Build it

Anonymous Communication

Attribute-based access control

Lots of policies

HCI

Standards, standards, standards, ...

..... see www.prime-project.eu for more information

# Conclusions

- Efficient Private Credentials and more are practical!
- EU Projects PRIME and PrimeLife used/are using this

  www.prime-project.eu.org and www.primelife.eu

- Software implementation (idemix) can be obtained from PrimeLife.eu

# Thanks for your attention!

More Information:
jca@zurich.ibm.com
www.zurich.ibm.com/security/idemix