Anonymous Credentials – All Details

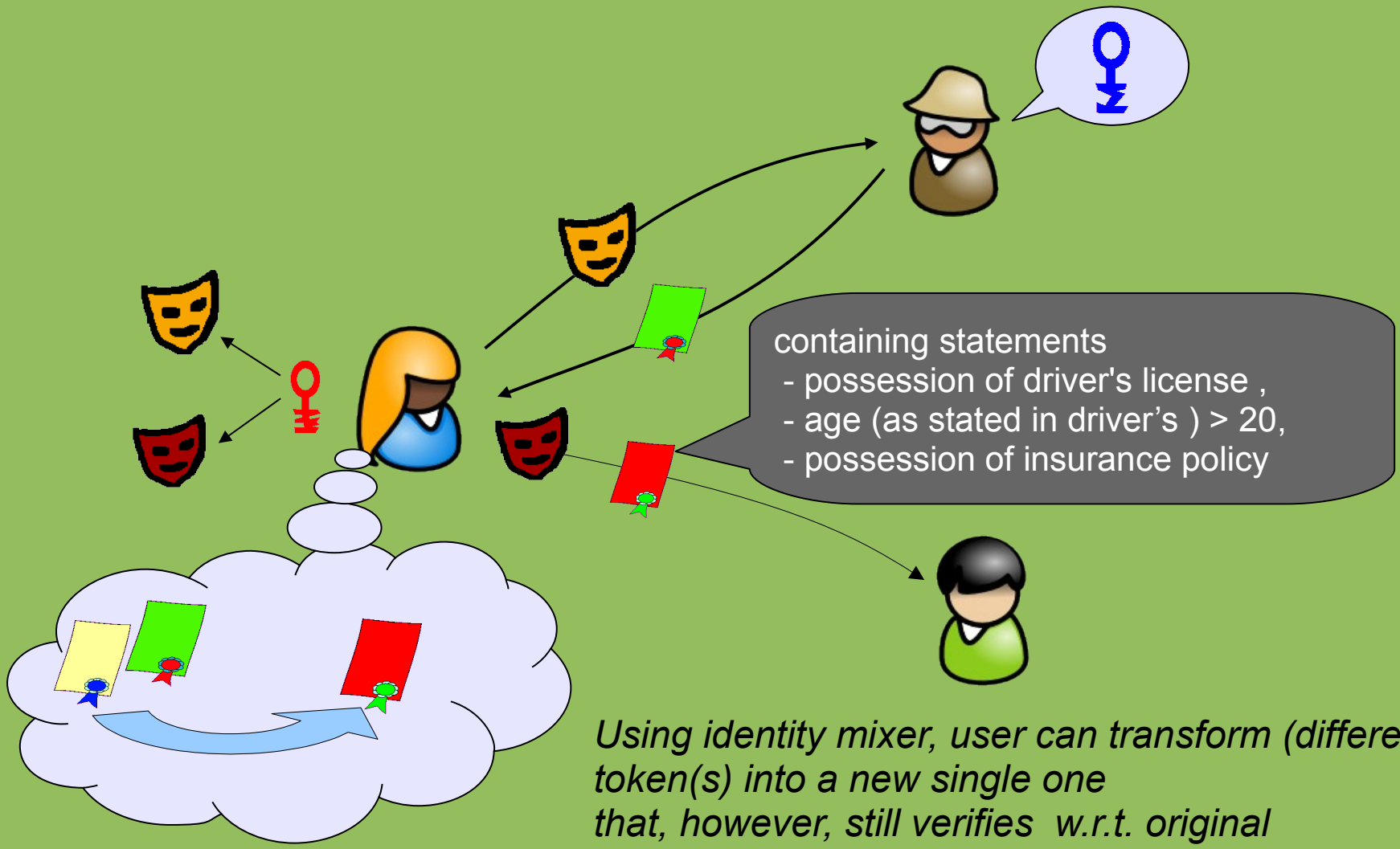
Crypto for Privacy Tutorial



Jan Camenisch

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

Recall Private Credentials



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.

We need efficient crypto mechanisms for:

1) Commitment scheme (one SK, many PKs)

2) Signature scheme to issue certificates
   - Sign (some) messages (possibly blind)
   - Prove statements about messages without revealing them
     → Proof protocols for *proving* possession of certificates

3) Additional crypto mechanisms
   - Verifiable encryption under TTP's key (tracing etc)
   - Cryptographic accumulators (revocation of credentials)
   - .... and specialized protocols, e.g., for private access control


Proof Protocols

Zero Knowledge Proofs

Logical combinations:

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

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

Efficient: computing essentially the expression by prover and verifier

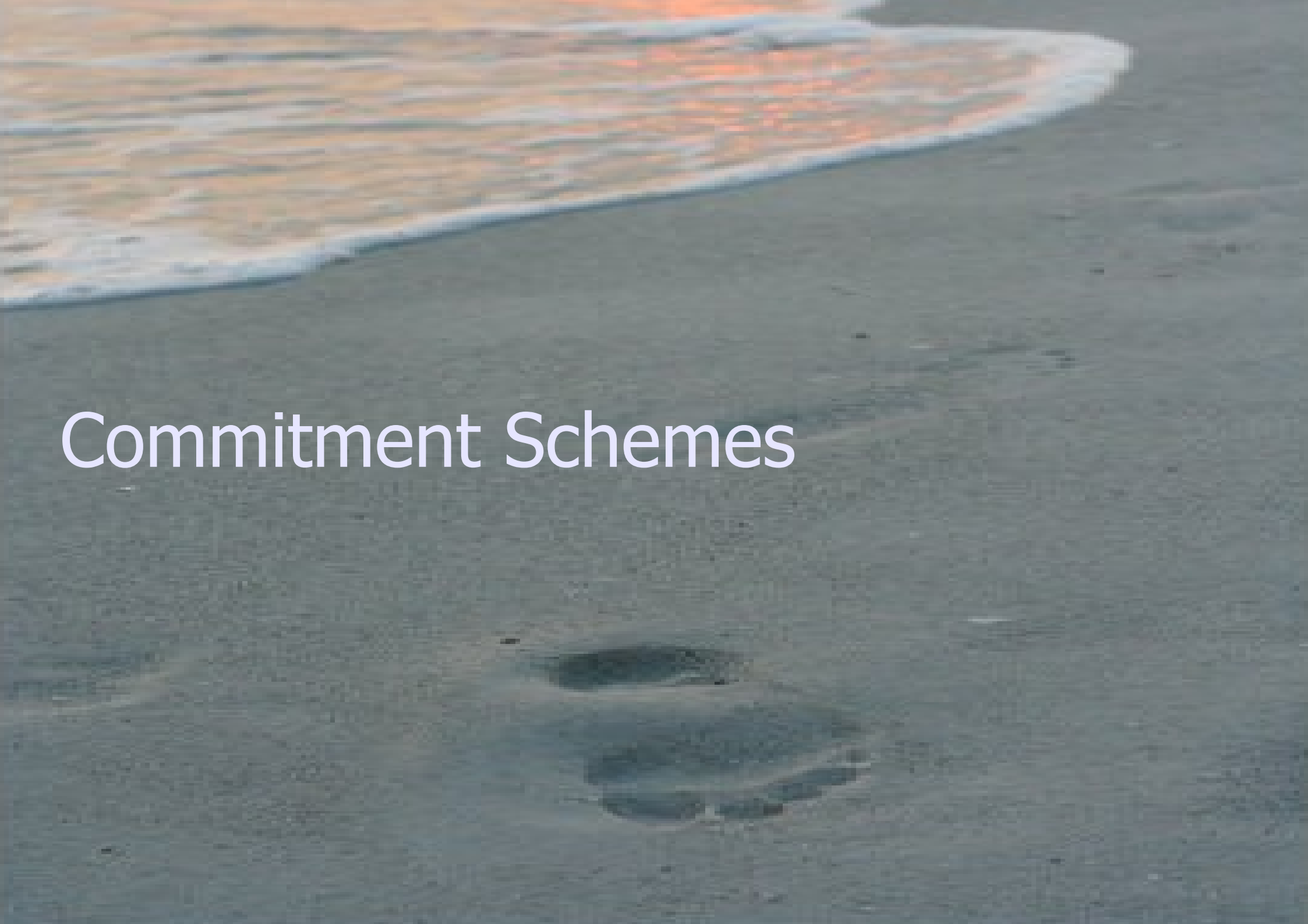
Intervals and groups of different order (under SRSA):

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

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

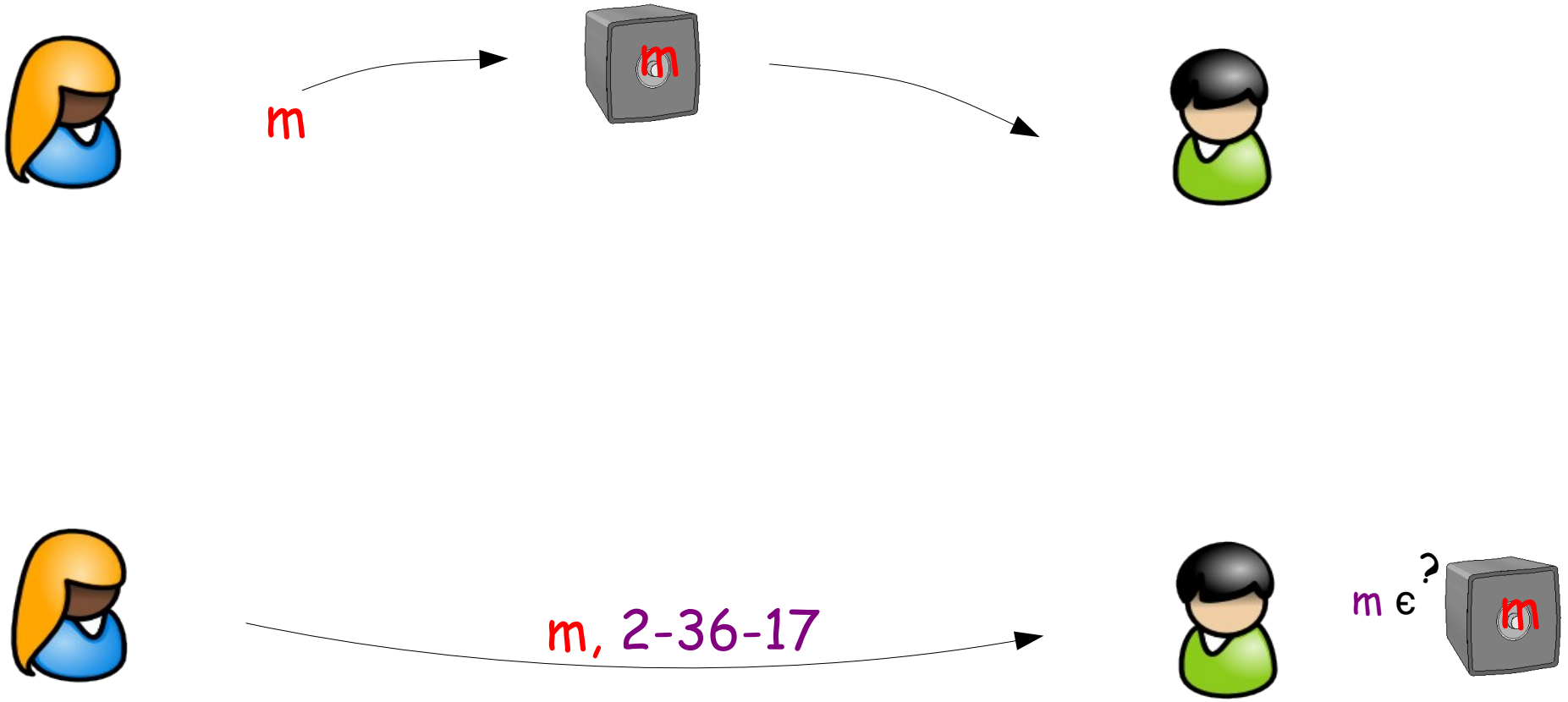
Non-interactive (Fiat-Shamir heuristic):

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



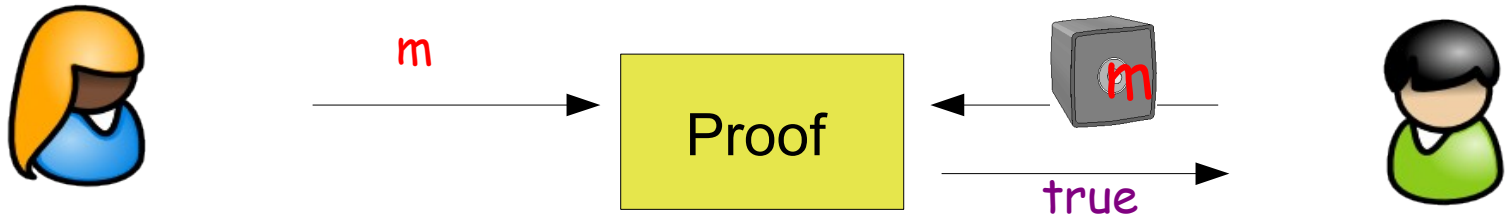
Commitment Schemes

Commitment Scheme

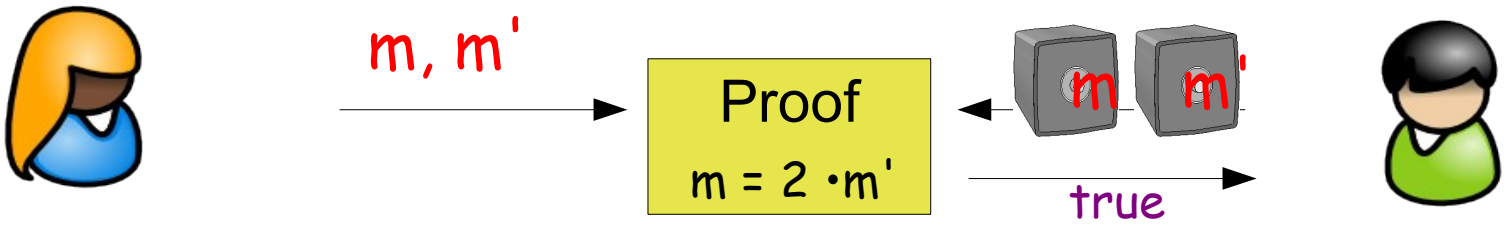


Commitment Scheme: Features

Proof of Knowledge of Contents



Proof of Relations among Contents



Commitment Schemes

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

To commit to element $x \in Z_q$:

- Pedersen: perfectly hiding, computationally binding

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

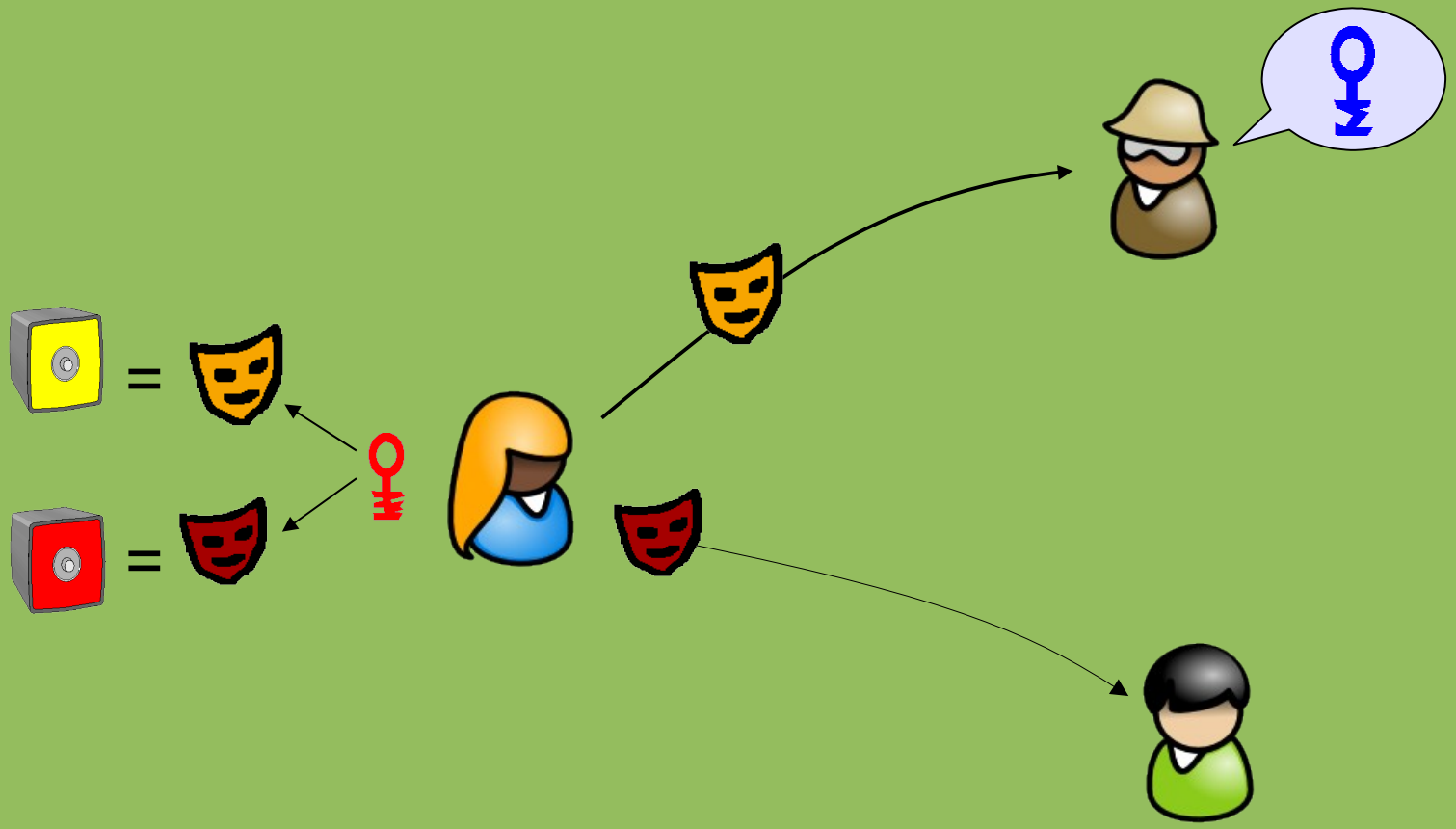
- ElGamal: computationally hiding, perfectly binding:

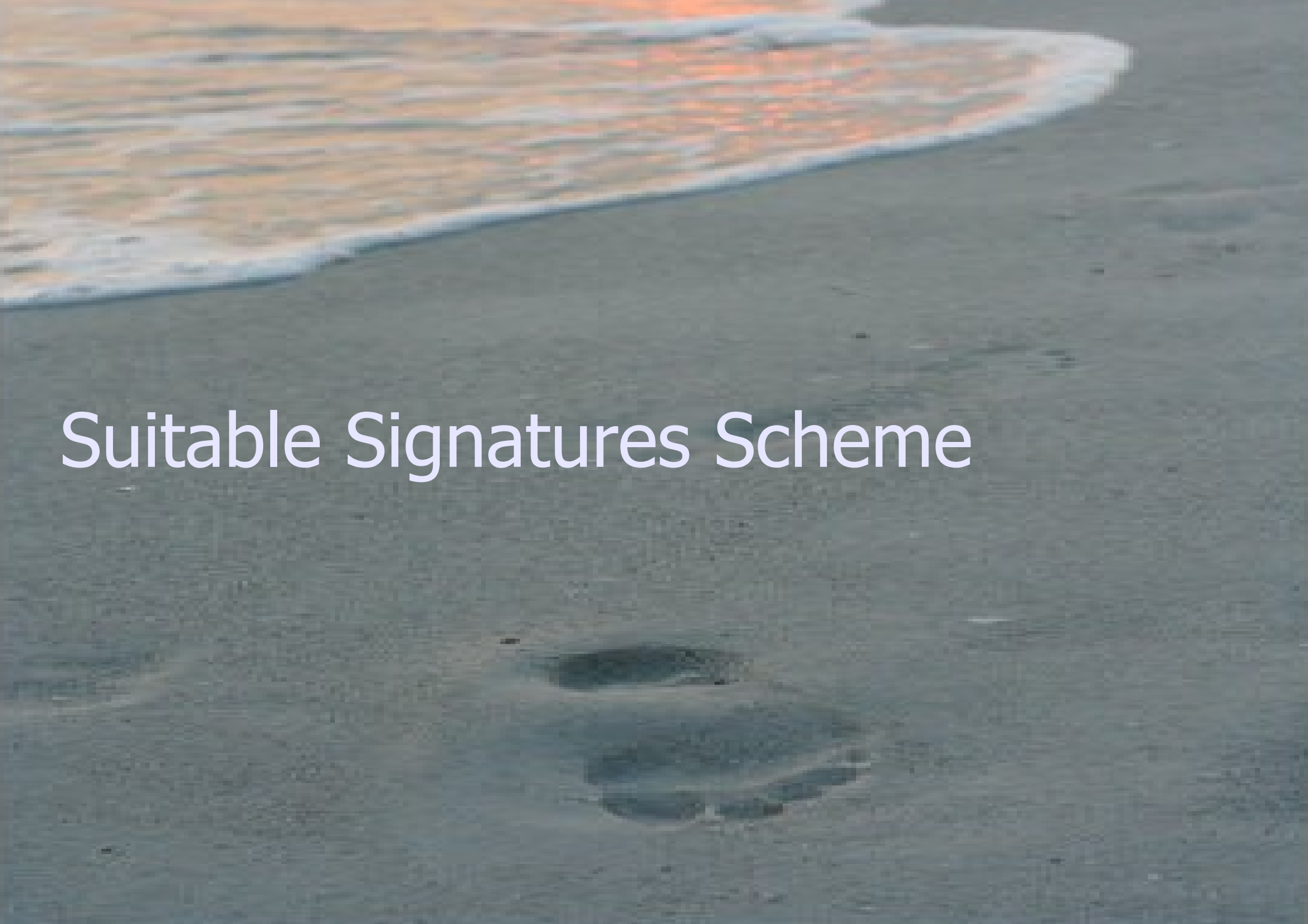
  choose $r \in Z_q$ and compute $c = (g^x h^r, g^r)$

To open commitment:

- reveal x and r to verifier
- verifier checks if $c = g^x h^r$

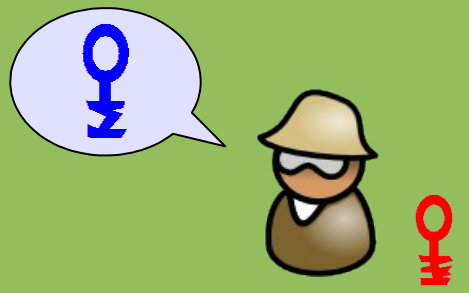
How we use commitments




Suitable Signatures Scheme

Signature Scheme: Algorithms

Key Generation:
Signer generates
- public key (verification key)
- secret key (signing key)



KeyGen = (,)



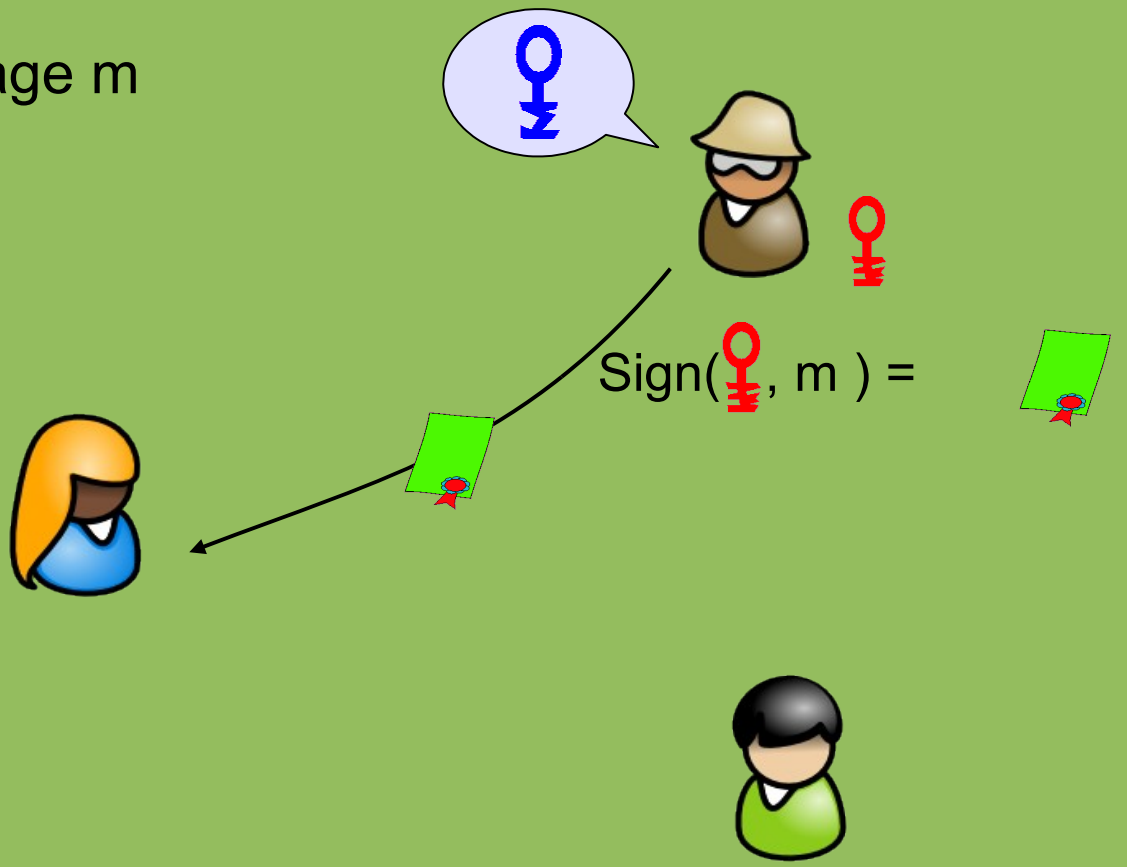


Signature Scheme: Algorithms

Signing Algorithm

Signer signs message m

- input: secret key and message m
- output: signature on m

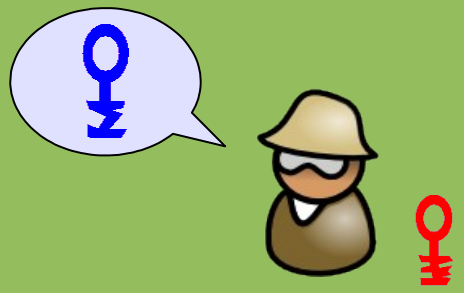


Signature Scheme: Algorithms

Verification Algorithm

Anyone can verify signature on a message

- input: public key , signature , and message m
- output: yes/no





Verify(, ) = yes



Verify(, ) = yes

(remark: no privacy yet ....)

Digital Signature Schemes 4 Privacy

- Sign blocks of messages m1, … , mk
- Compatible with proof protocols
- Some known schemes:
  - Brands/U-Prove (Discrete Log/Blind Signature)
  - Camenisch-Lyskanskaya (Strong RSA)
  - Camenisch-Lyskanskaya (Bilinear Maps; LRSW, q-SDH)
  - ....a number of others, but not really practical yet
    - P-Signatures – Belinkiy et al. (q-SDH)
    - Lattice-based ones (Gordon et al.)



CL Signature Scheme

SRSA Variant

RSA Signature Scheme (for reference)

Rivest, Shamir, and Adlemann 1978

Secret Key: two random primes p and q

Public Key: $n = pq$, prime e,
and collision-free hash function

$$H: \{0,1\}^* \rightarrow \{0,1\}^{\ell}$$

Computing signature on a message $m \in \{0,1\}^*$

$$d = 1/e \bmod (p-1)(q-1)$$

$$s = H(m)^d \bmod n$$

Verification signature on a message $m \in \{0,1\}^*$

$$s^e = H(m) \quad (\bmod n)$$

Signature Scheme based on SRSA [CL01]

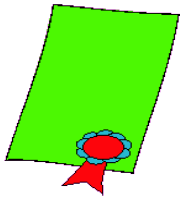
Public key of signer: RSA modulus n and $a_i, b, d \in QR_n$

Secret key: factors of n

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

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

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

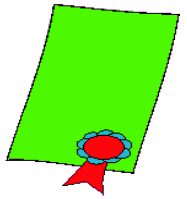


- signature is (c,e,s)

Signature Scheme based on 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 Strong RSA assumption.*



Ext. Schnorr Signature Scheme

(also called signature of knowledge)

From Protocol To Signature



Protocol PK{(α): $y = g^{\alpha}$ }

Prover: Verifier:



random $r \in Z_q$

$t := g^r$ — t →

← c — random $c \in \{0,1\}^l$

$s := r - cx$ (q) — s →

$t = g^s y^c$

From Protocol To Signature

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



Signing a message m:

- chose random $r \in Z_q$ and
- compute $(c,s) := (H(g^r||m), r - cx \quad (q))$

Verifying a signature (c,s) on a message m:

- check $c = H(g^s y^c||m)$?



Security:

- Discrete Logarithm Assumption holds
- Hash function H(.) behaves as a random oracle.



U-Prove Signature Scheme

U-Prove seen as a Signature Scheme [Brands93,..]

Public key of signer: Group $G = \langle g \rangle$ and $g_0, g_1, \ldots, g_k, g_{k+1}, y = g^x$

Secret key: x

To sign k messages $m1, \ldots, mk \in Zq$:

I. Choose random $a \in Zq$

II. Compute $h = (g_0 g_1^{m1} \cdot \ldots \cdot g_k^{mk} g_{k+1}^{ctype})^a$ and $z = h^x$

III. Compute $(c, s) = SPK\{(x): y = g^x \wedge z = h^x\}(z,h)$

IV. Signature is $(a, h, z, (c, s))$

where ctype is a fixed non-zero value derived from public key and an application identifier.

Note: a could be fixed value if we are only interested in sign. scheme.

U-Prove seen as a Signature Scheme [Brands93,..]

Public key of signer: Group $G = \langle g \rangle$ and $g_0, g_1, \ldots, g_k, g_{k+1}, y$

To verify signature $(a, h, z, (c, s))$ on k messages $m1, \ldots, mk \in Z_q$:

I. check $h = (g_0 g_1^{m1} \cdot \ldots \cdot g_k^{mk} g_{k+1}^{ctype})^a$

II. verify $(c, s) = SPK\{(x): y = g^x \wedge z = h^x\}(z,h)$

Security:

- requires DL to be hard ...

- … but assumption is that scheme is secure (no reduction known)

Alternative Signature Schemes

Signature schemes that follow the same paradigm

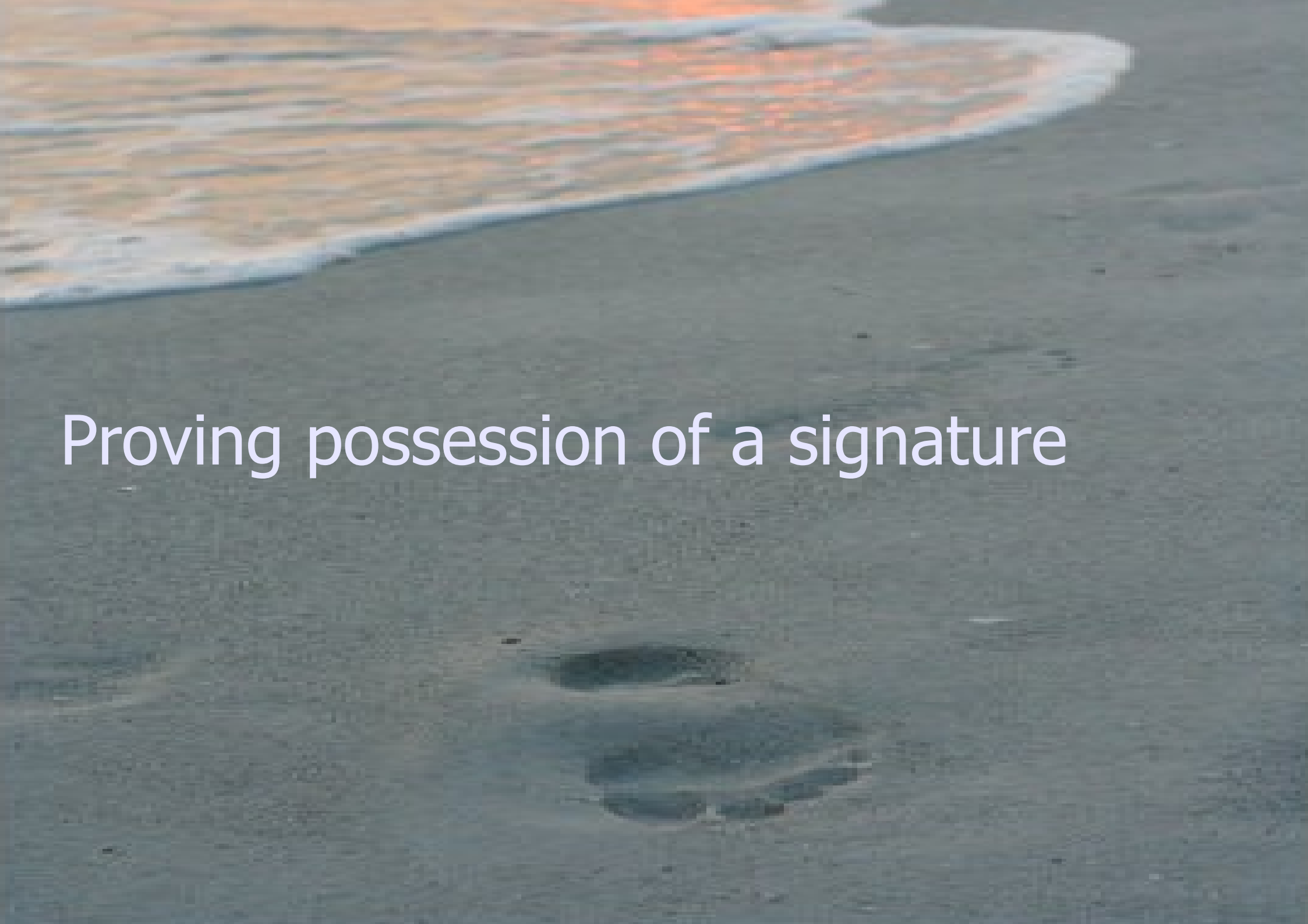
- SRSA-based (just presented)
- Bi-linear maps based: LRSW-Assumption
- Bi-linear maps based: q-SDH-Assumption

.... Brands DL-based scheme (uProve)

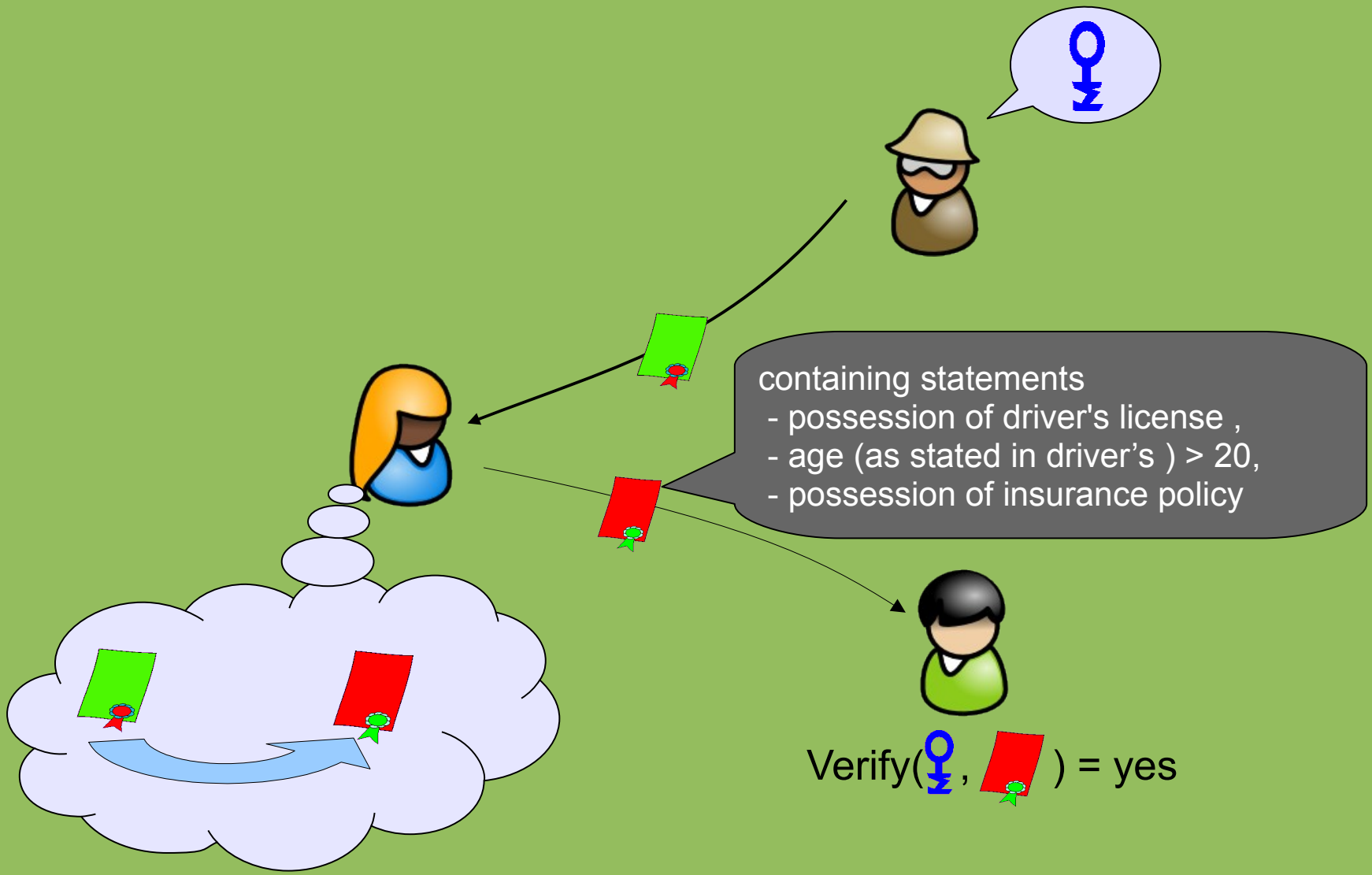
- Blind signatures (use once)
- No security proofs

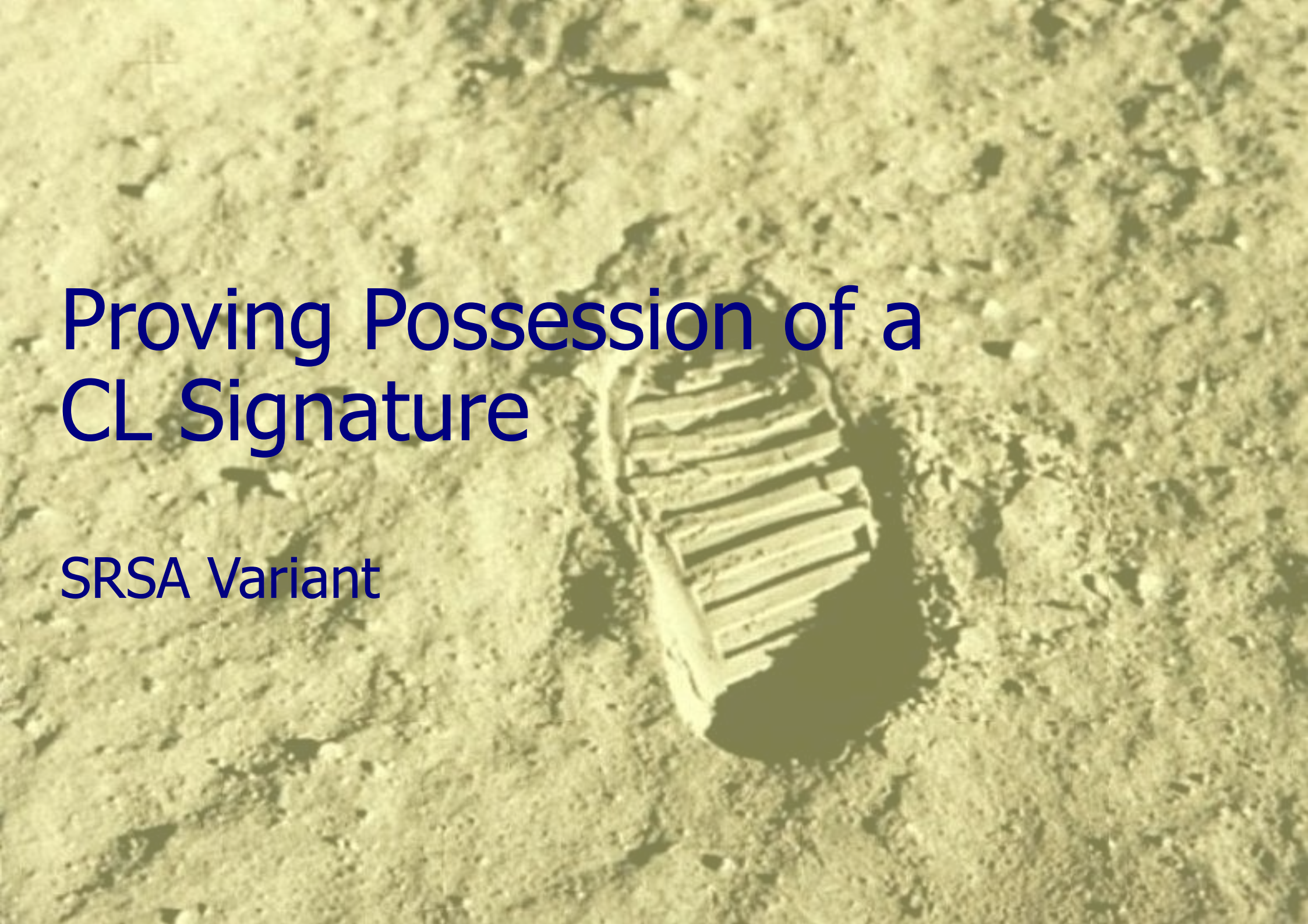
… a number of schemes geared towards GS-proofs

- not efficient enough yet


Proving possession of a signature

Recall Goal...


containing statements
- possession of driver's license ,
- age (as stated in driver's) > 20,
- possession of insurance policy
Verify(,) = yes



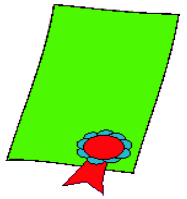
Proving Possession of a CL Signature

SRSA Variant

Recall Verification of Signature

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

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



Thus to prove knowledge of values

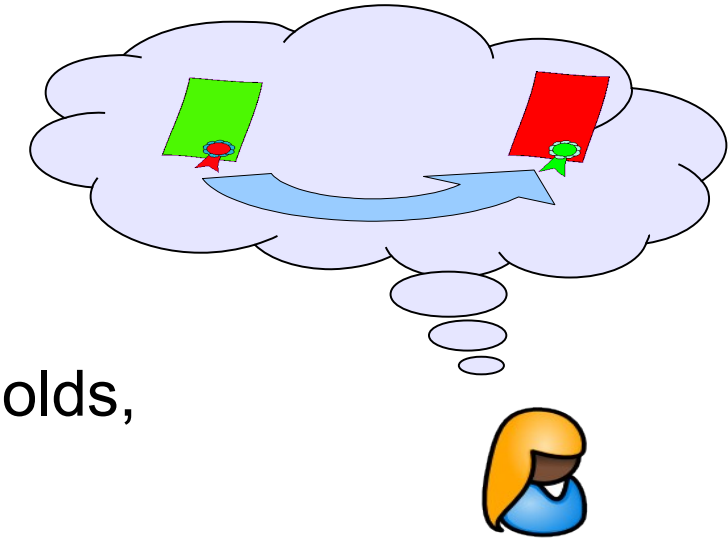
$$m1, \ldots, mk, e, s, c$$

such that the above equations hold.

Problem: c is not an exponent...

Proof of Knowledge of a CL Signature

Solution randomize c :



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

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

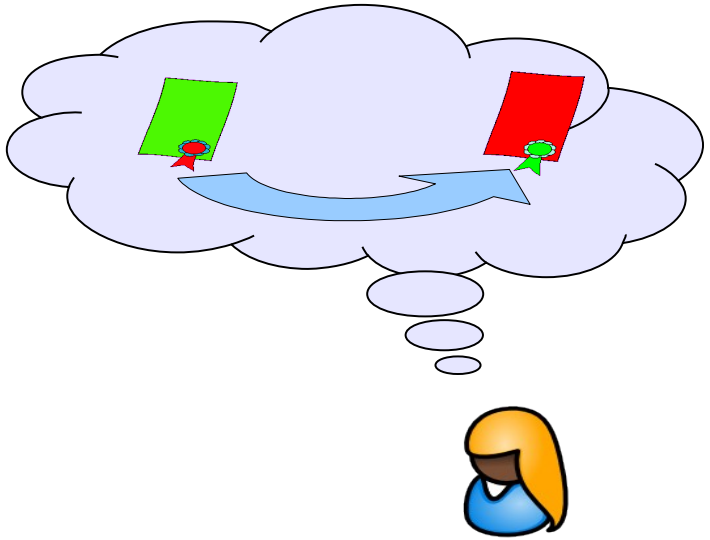
Therefore, to prove knowledge of signature on hidden msgs:

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

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

Proof of Knowledge of a CL Signature

.....revealing some of the messages



- Randomise, $c' = c\, b^{s'} \bmod n$
- provide c'
- PK{$(e, m3, \ldots, mk, s)$:

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

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



Proving Possession of a U-Prove Signature

Presentation of U-Prove Tokens

Recall: To verify signature $(a, h, z, (c, s))$ on k messages $m1, \ldots, mk \in Z_q$:

- Check $h = (g_0 \, g_1^{m1} \cdot \ldots \cdot g_k^{mk} \, g_{k+1}^{ctype})^a$
- verify $(c, s) = SPK\{(x): y = g^x \ \wedge \ z = h^x\}(z,h)$

To prove possession of signature $(a, h, z, (c, s))$ on messages mi :

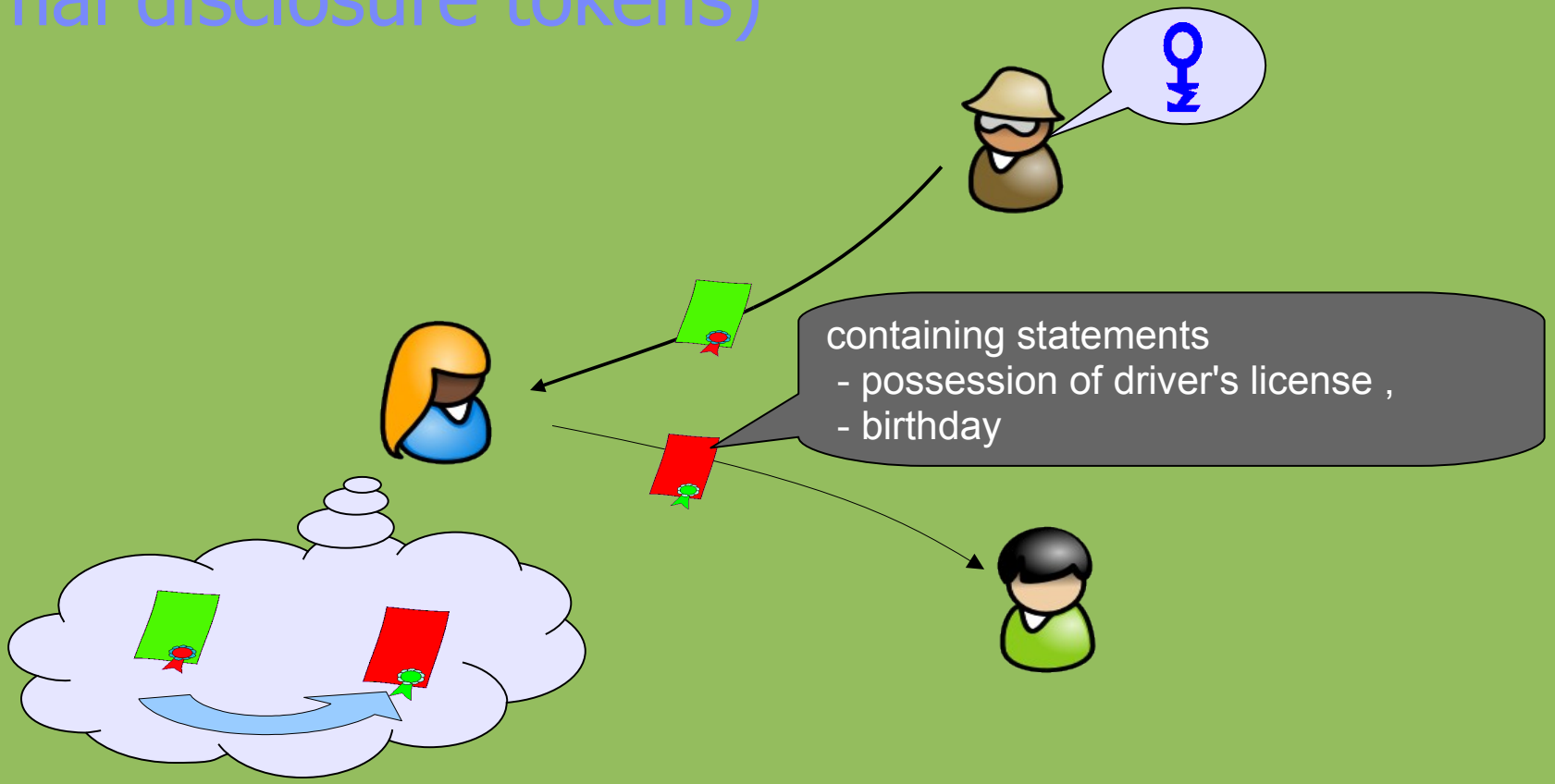
- $(c', s_i) = SPK\{(mi, a): \ 1/(g_0 \, g_{k+1}^{ctype}) = h^{-1/a} \ g_1^{m1} \cdot \ldots \cdot g_k^{mk}\}$
- $(c, s) = SPK\{(x): y = g^x \ \wedge \ z = h^x\}(z,h)$

New signature: $(h, z, (c', s_i), (c, s), ctype)$

Note: reveals h, z, (c, s) of the original signature but hides messages!


Issuing Credentials
& doing minimal disclosure tokens

So we can already do anonymous tokens (or minimal disclosure tokens)



Includes:

- On-line e-cash (merchant checks validity of cash w/bank)

But no

- predicates of attribute, revocation, tracing, etc,....

… Issuing U-Prove Signatures/Credentials

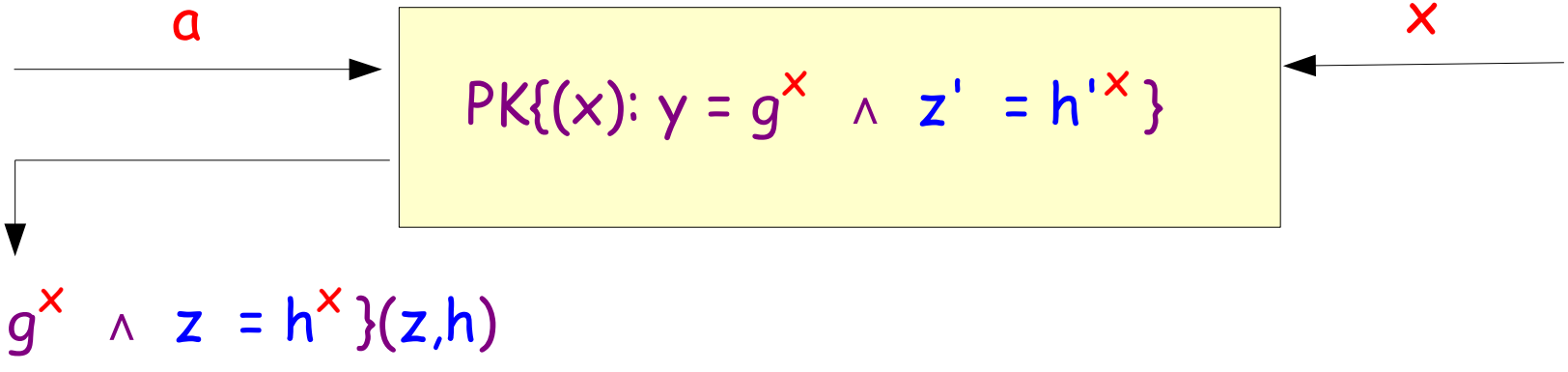




random $a \in Z_q$

$h = h'^{a}$

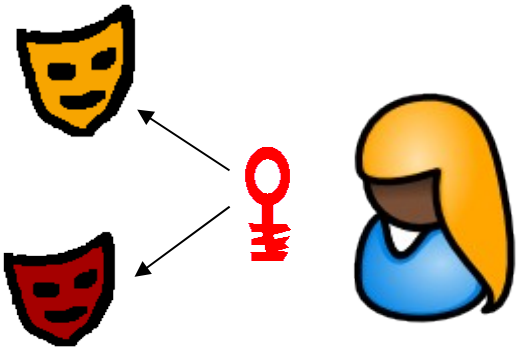
$h' = g_0 g_1^{m1} \cdot \ldots \cdot g_k^{mk} g_{k+1}^{ctype}$



- A kind of blind signature scheme to ensure privacy
- Security
  - Based on blind signatures, but not quite
  - Does *not* reduce to assuming U-Prove Signatures are secure


Coming Back to Pseudonyms

(Cryptographic) Pseudonyms



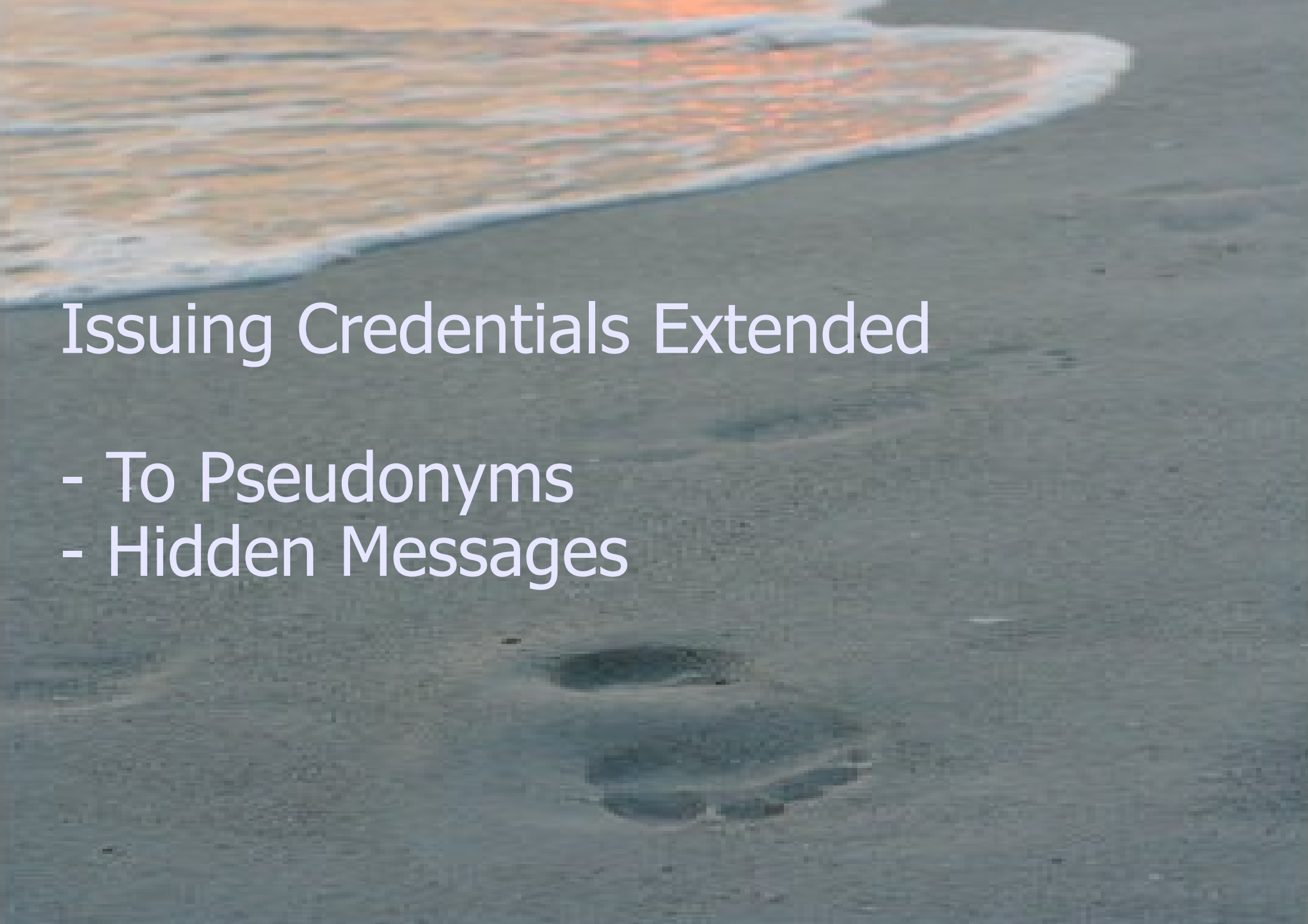
Algebraic Setting: Group $G = \langle g \rangle$ of order q.

Pseudonyms:

- Secret identity: $sk \in Z_q$.
- Pseudonym: pick random $r \in Z_q$ and compute $P = g^{sk} h^r$.
- Domain pseudonym: let $g_d = H(\text{domain})$. Then compute $P = g_d^{sk}$.
  Thus domain pseudonym as unique (per secret identity)

Security:

- Pseudonyms are perfectly unlinkeable.
- Domain pseudonyms are unlinkeable provided
  - Discrete logarithm assumption holds and
  - $H(\text{domain})$ is a random function.

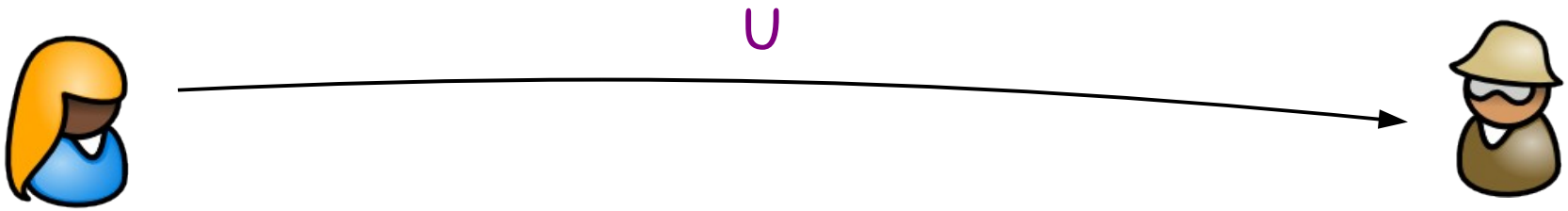

Issuing Credentials Extended
- To Pseudonyms
- Hidden Messages



CL Signature Scheme

Issuing aCredential to Hidden Messages (idemix)

$U := a_1^{m1} a_2^{m2} b^{s'}$



$PK\{(m1, m2, s') : U = a_1^{m1} a_2^{m2} b^{s'} \wedge mi \in \{0,1\}^{\ell} \}$

Issuing a Credential to Hidden Messages (idemix)

$U := a_1^{m1} a_2^{m2} b^{s'}$



U



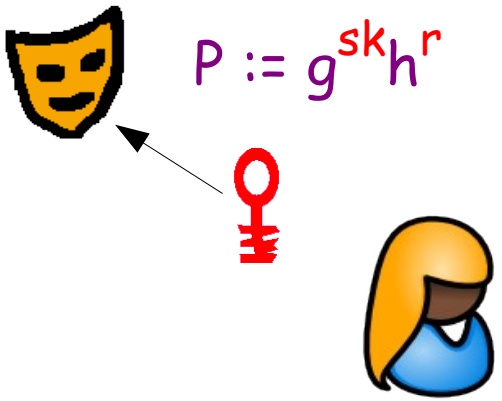
Choose e, s''

$c = (d/(U a_3^{m3} b^{s''}))^{1/e} \bmod n$

(c, e, s'')

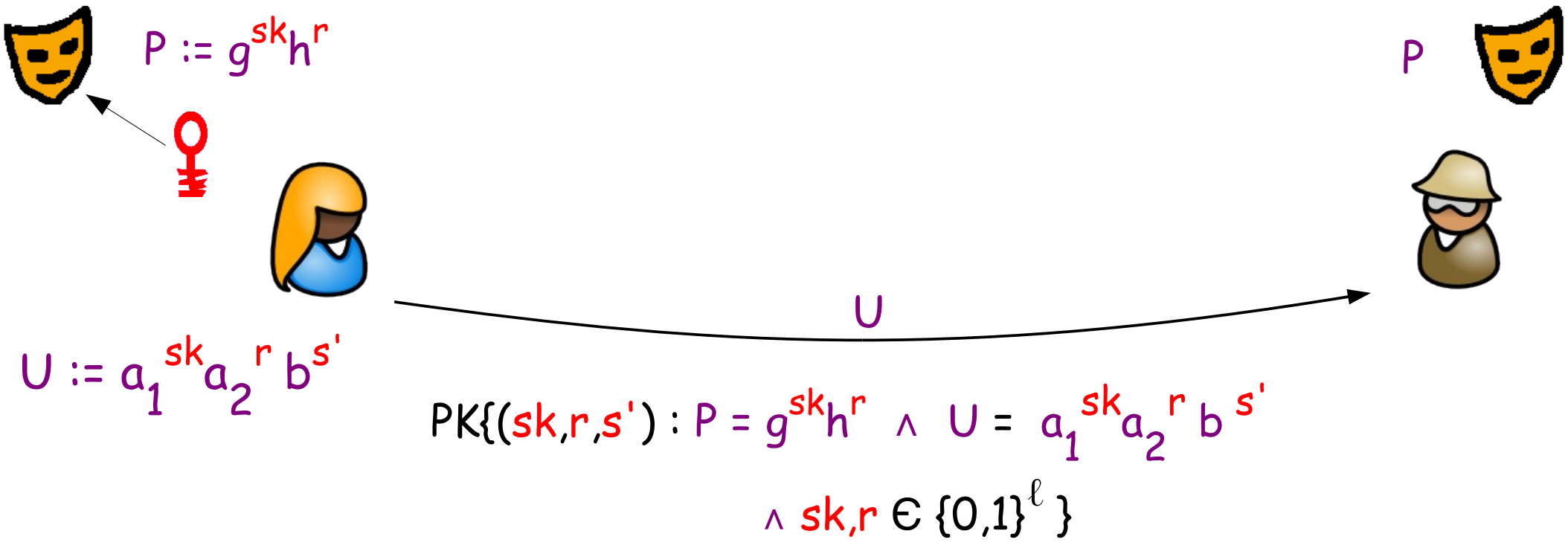
$$d = a_1^{m1} a_2^{m2} a_3^{m3} b^{s'' + s'} c^{e} \pmod{n}$$

Issuing a Credential to a Pseudonym (idemix)





Issuing a Credential to a Pseudonym (idemix)



.... and then issue credential just as before

Issuing on Hidden & Committed Attributes

$Com := g_1^{m1} g_2^{m2} h^r$



$U := a_1^{sk} a_2^{m1} a_3^{m2} b^{s'}$

Com



U

$PK\{(sk,m1,m2,r,s') : Com = g_1^{m1} g_2^{m2} h^r \wedge$

$U = a_1^{sk} a_2^{m1} a_3^{m2} b^{s'} \wedge$

$sk,m1,m2 \in \{0,1\}^{\ell}\}$

Example use case: issue credential on last name

- Commit to last name
- Prove correctness using Government credential
- Get new credential issued



U-Prove Signature Scheme

Recall: issuing of signatures

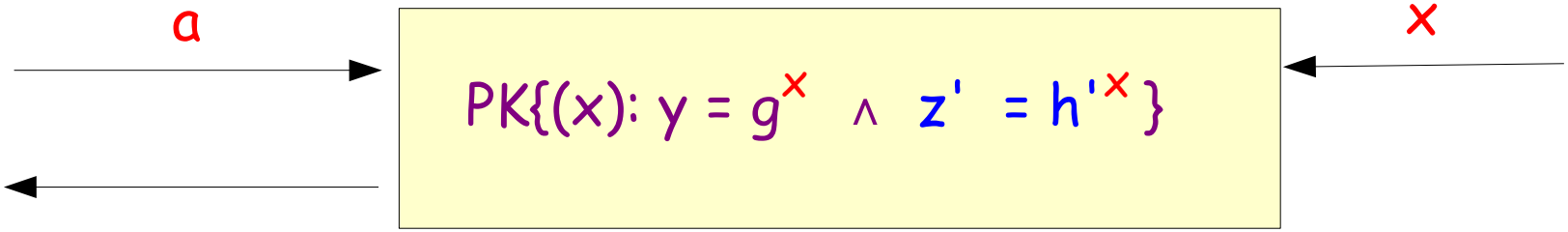




random $a \in Z_q$

$h = h'^{a}$

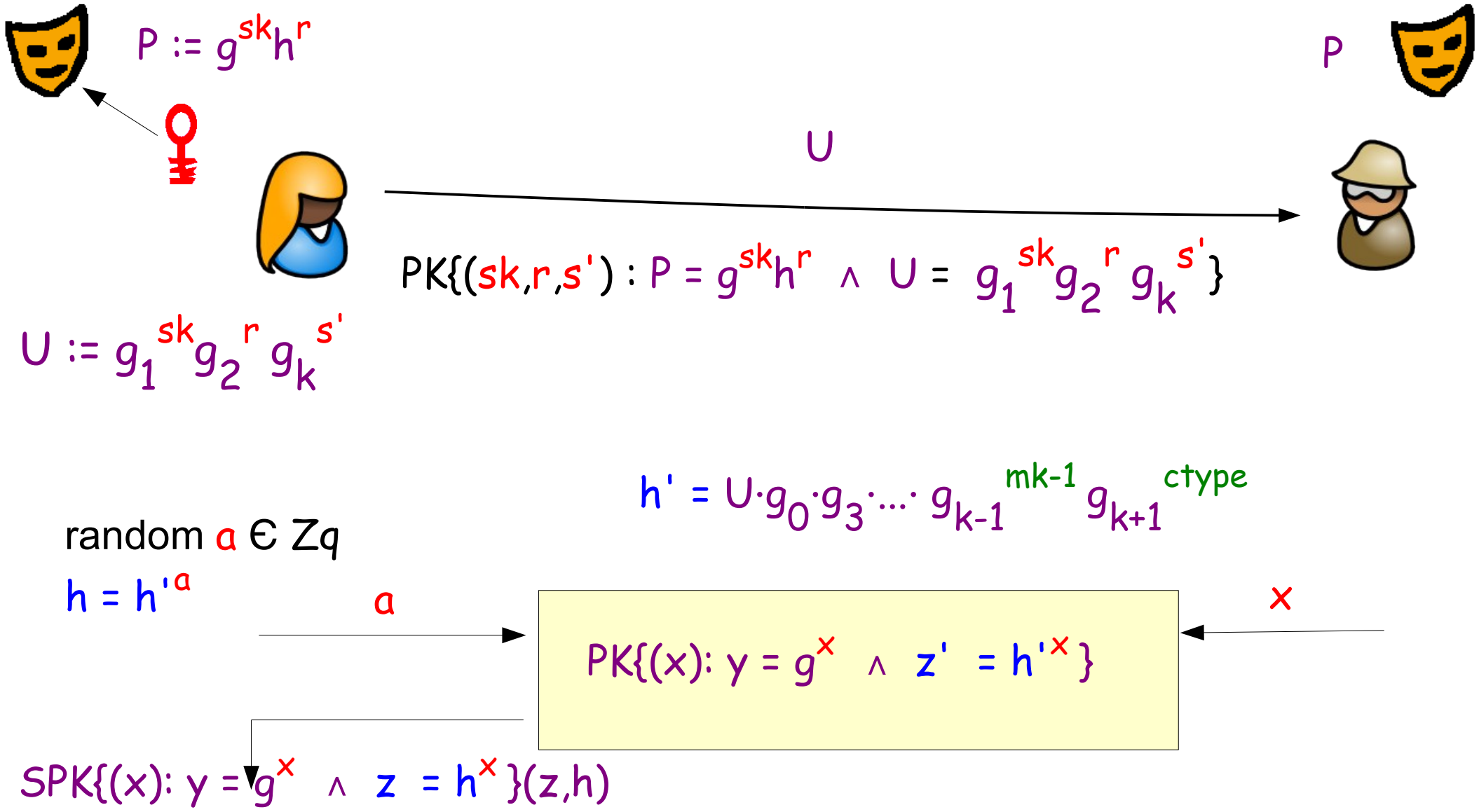
$h' = g_0 g_1^{m1} \cdot \ldots \cdot g_k^{mk} g_{k+1}^{ctype}$



$SPK\{(x): y = g^x \wedge z = h^x\}(z,h)$

- A kind of blind signature scheme to ensure privacy
- Security
  - Based on blind signatures, but not quite
  - Does *not* reduce to assuming U-Prove Signatures are secure

Issuing on Pseudonyms: essentially the same....



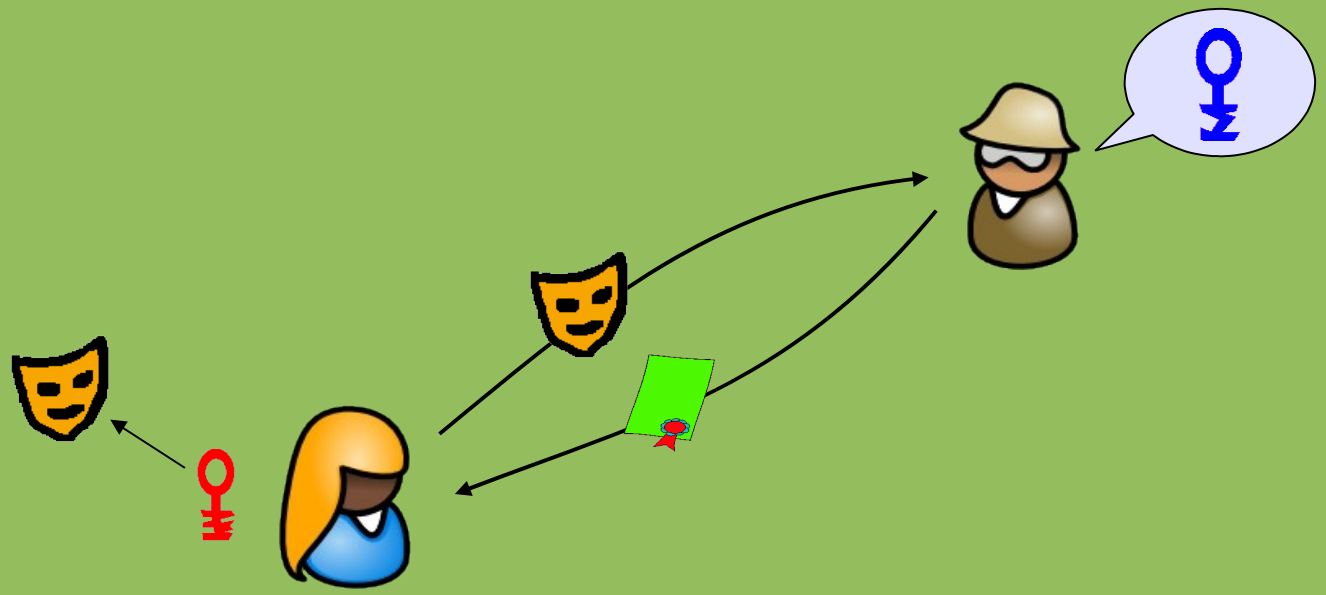


Example: Polling

Polling: Scenario and Requirements

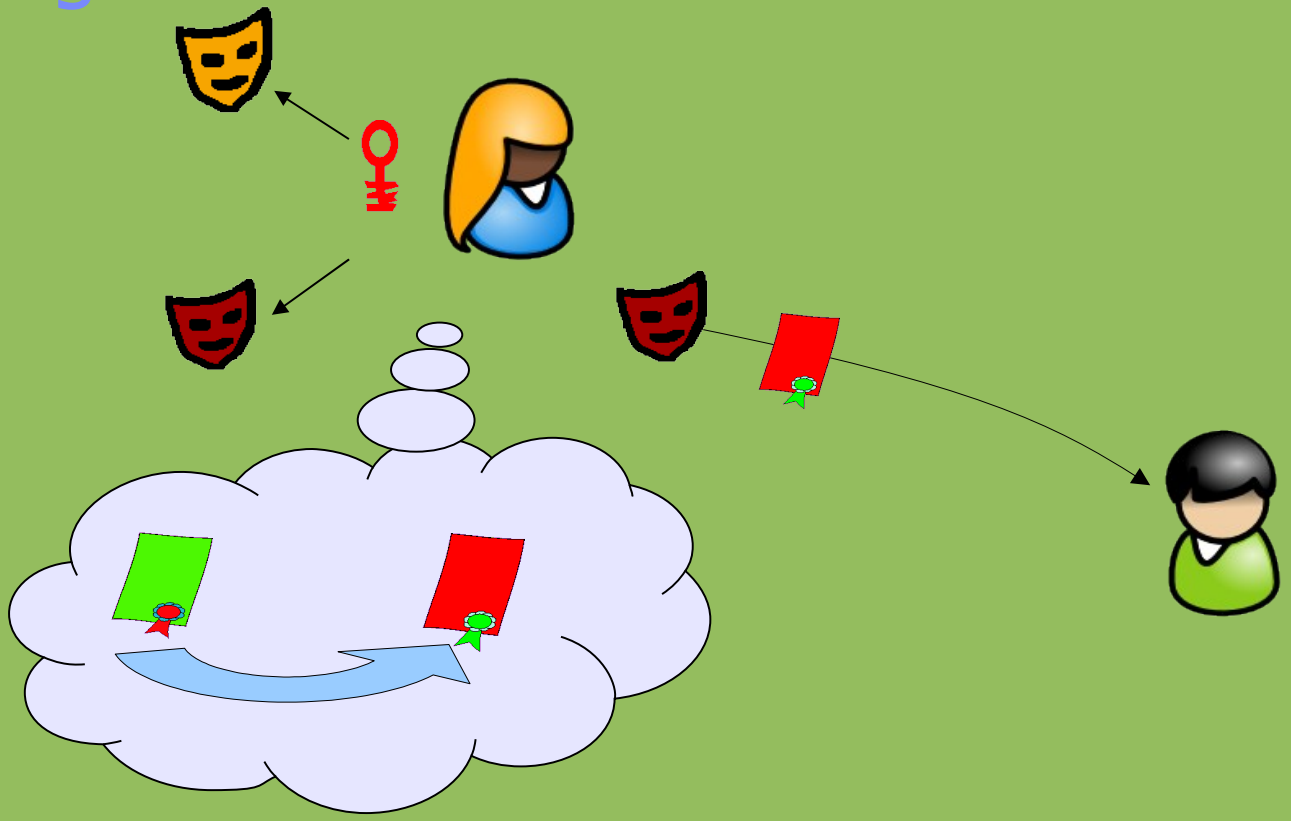
- User obtain credential that they are eligible under ID
- User can voice opinion anonymously
  - Different polls must not be linkable
- User can do so only once per poll
  - or if they do multiple times, only one voice counts – first, last?

Polling: Solution



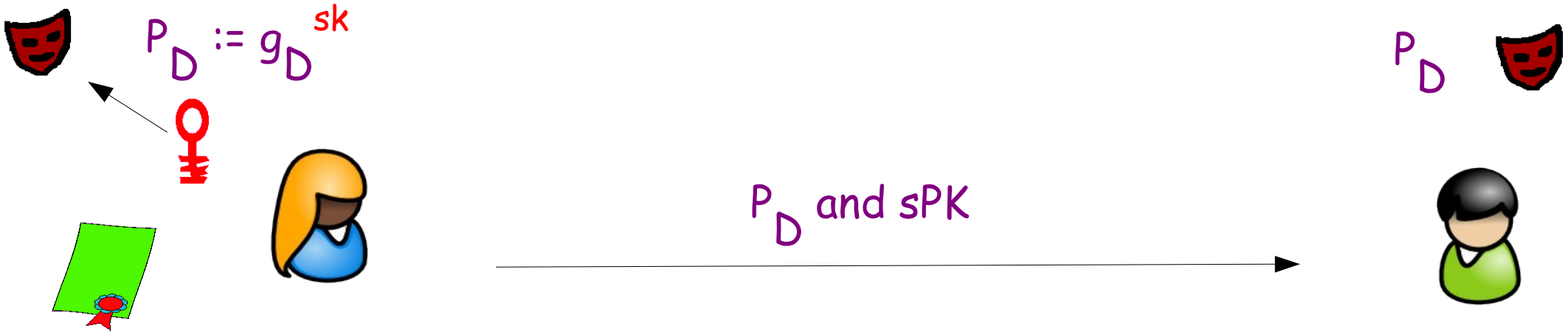
- User generates pseudonym (Id for registration)
- User obtains credential on pseudonym stating that she is eligible for polls
- Credential can contain attributes about her

Polling: Solution



- User generates *domain pseudonym*
  - Pseudonym is unique per domain, but not linkeable otherwise
- User transforms credential
  - aka proves that she possess the necessary credential

Proving a credential w.r.t. a Pseudonym



SPK{(sk,r,m2, .., mk) :

$P_D = g_D^{sk} \wedge$

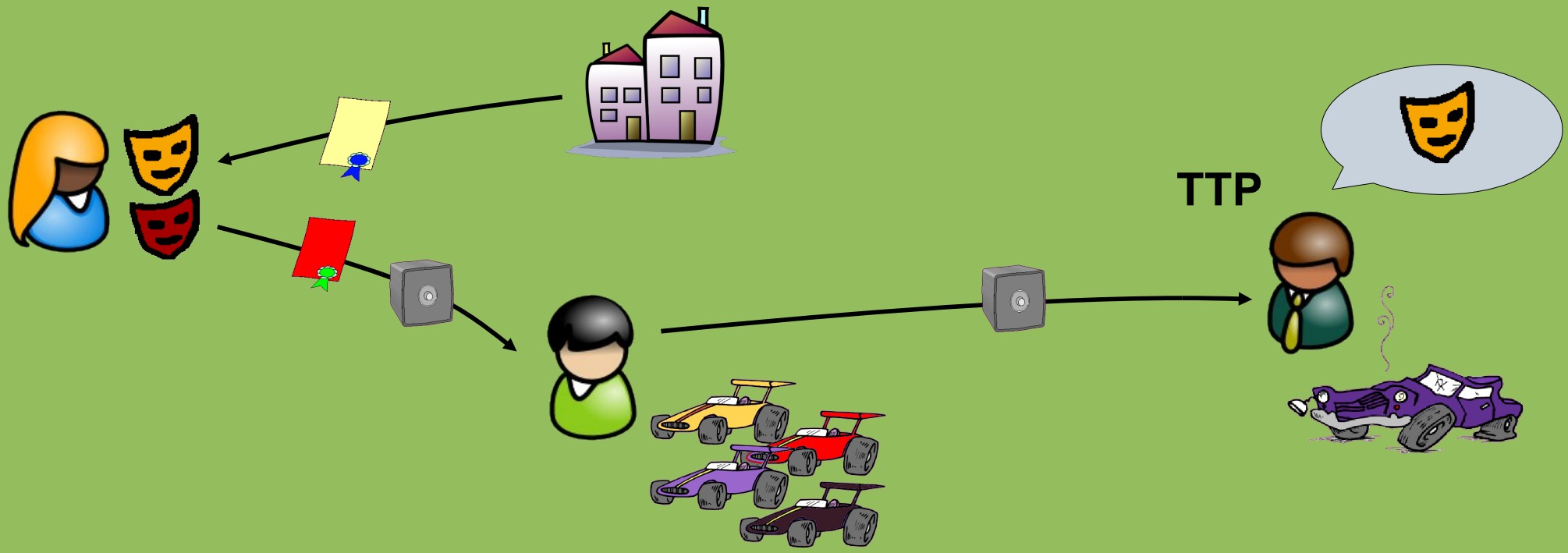
$d = c'^{e} a_1^{sk} \cdot a_2^{r} \cdot a_3^{m3} \cdot \ldots \cdot a_k^{mk} b^{s} \wedge$

$sk,r,mi \in \{0,1\}^{\ell} \wedge e \in 2^{\ell+1} \pm \{0,1\}^{\ell}$ }(opinion)

- Domain pseudonym different for each poll, e.g., $g_D = f(poll)$

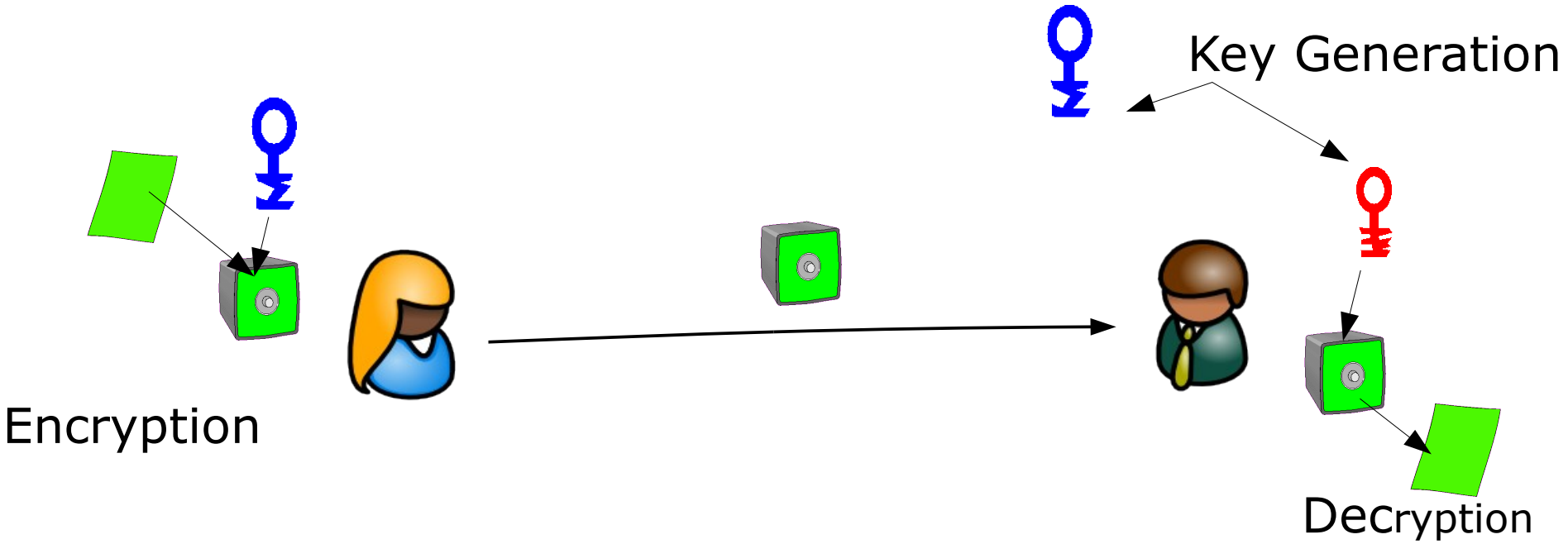

Verifiable Encryption

Motivation: Tracing & 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

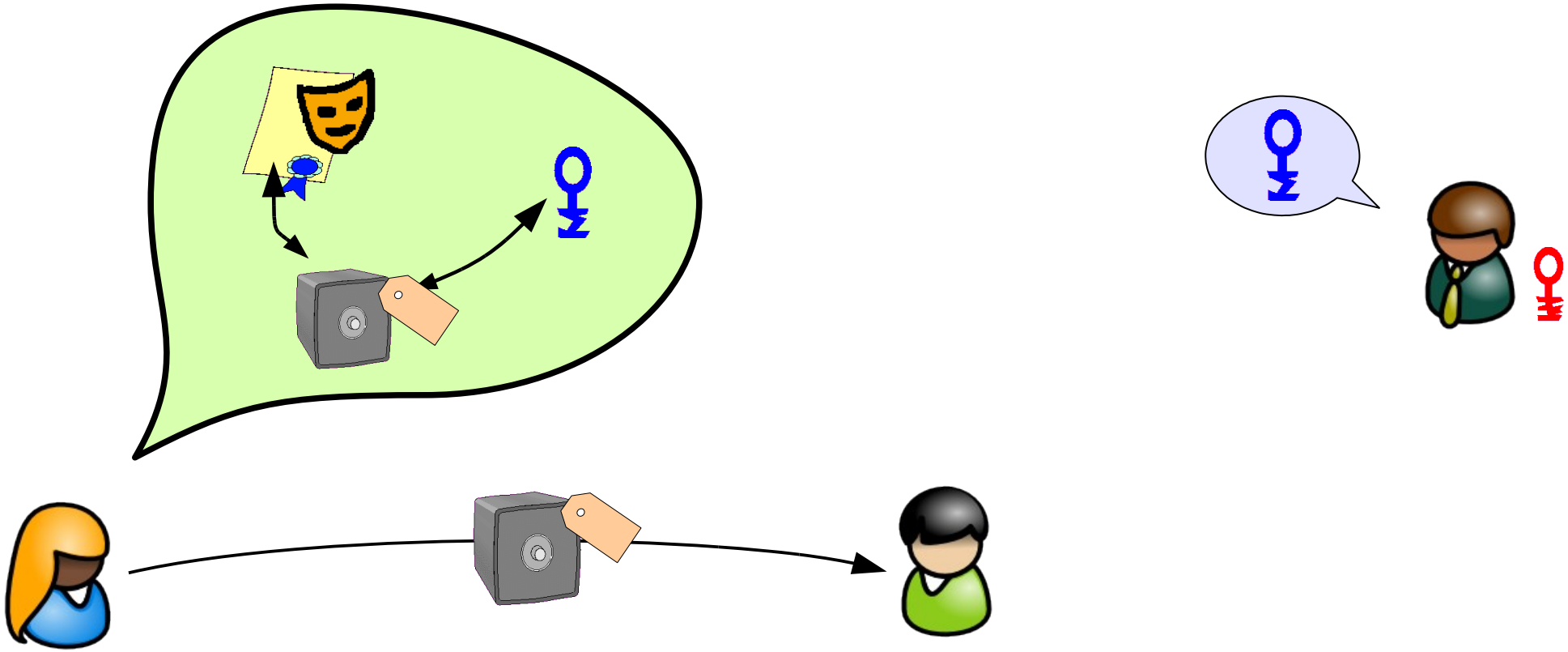
Public Key Encryption: algorithms



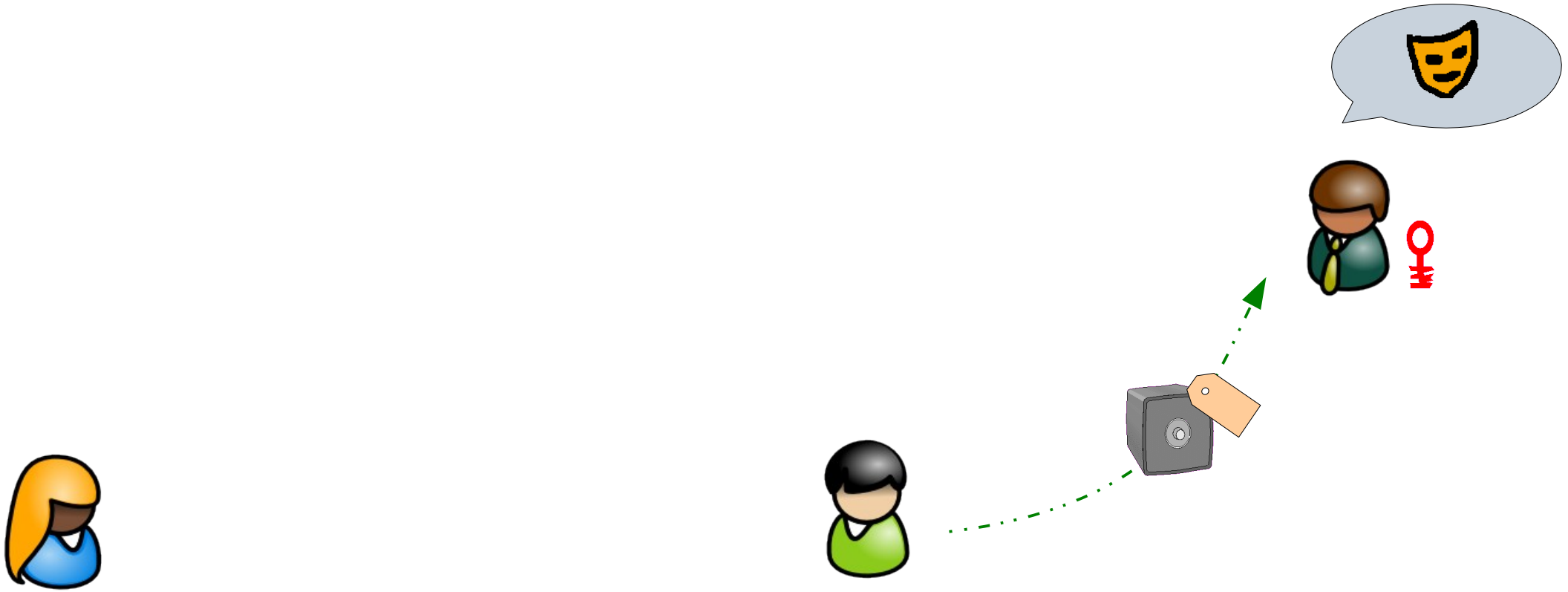
Security Definitions (far from trivial...)

- Semantic security: ciphertext does not leak if scheme is used once only.
- Adaptive security: .... if used continuously.

Verifiable Encryption with Label



Verifiable Encryption with Label



Verifiable Encryption

- Of attributes (discrete logarithm)
  - Camenisch-Shoup (SRSA) – based on Paillier Encryption
- Of pseudonyms (group elements)
  - Cramer-Shoup (DL) or rarely ElGamal (DL)
- Otherwise
  - Camenisch-Damgaard, works for any scheme, but much less efficient
- ....Open Problem to find new ones!



ElGamal Encryption
(for Pseudonyms & Tracing)

ElGamal Encryption scheme

Public Key: Group $G = \langle g \rangle$ of order q $y := g^x$

Secret Key: $x \in Z_q$

Encryption of a message $m \in G$:

- Choose $r \in Z_q$
- Compute ciphertext $(c1,c2)$ as: $c1 := g^r$, $c2 := y^r m$

Decryption of a ciphertext $(c1,c2)$:

- Compute $m' = c2\ c1^{-x}$ $(= y^r m g^{-rx} = y^r m y^{-r} = m)$

Semantically secure under Discrete Log assumption. *Cramer-Shoup encryption scheme is adaptive secure extension that should be used.*

Tracing: Encryption of a Certified Pseudonym

Public Key Of Tracer: Group $G = \langle g \rangle$ of order q $y := g^x$

Pseudonym with issuer: $P := g^{sk} h^r$

…by definition of credential by issuer $d = c^e a_1^{sk} \cdot a_2^r \cdot a_3^{m3} \cdot \ldots \cdot a_k^{mk} b^s$

To make a traceable presentation of credential, user

- Choses rand. r and computes $c1 := g^{r'}$, $c2 := y^{r'} P \ (= y^{r'} g^{sk} h^r)$
- Computes $c' = c\, b^{s'} \bmod n$ with random s'
- Sends $(c', (c1, c2))$ to verifier
- Computes SPK{$(sk, r, m2, .., mk)$:

$$c1 := g^{r'},\ c2 := y^{r'}\ g^{sk} h^r \ \wedge$$

$$d = c'^e a_1^{sk} \cdot a_2^r \cdot a_3^{m3} \cdot \ldots \cdot a_k^{mk} b^s \}(c1, c2, tr_policy)$$



Excursus: Accountability

Making the User Accountable: Discussion

Scenario:

- User registers pseudonym with issuer
- Get credential from the issuer
- Presents credential to verifier with tracing (encryption of registered pseudonym)
- TTP traces a presentation proof and points to a user

Problems/attacks:

- How can one prove that it really was the user?
- Could the issuer just generate another pseudonym and credential and then blame the user?

Making the User Accountable: Solution

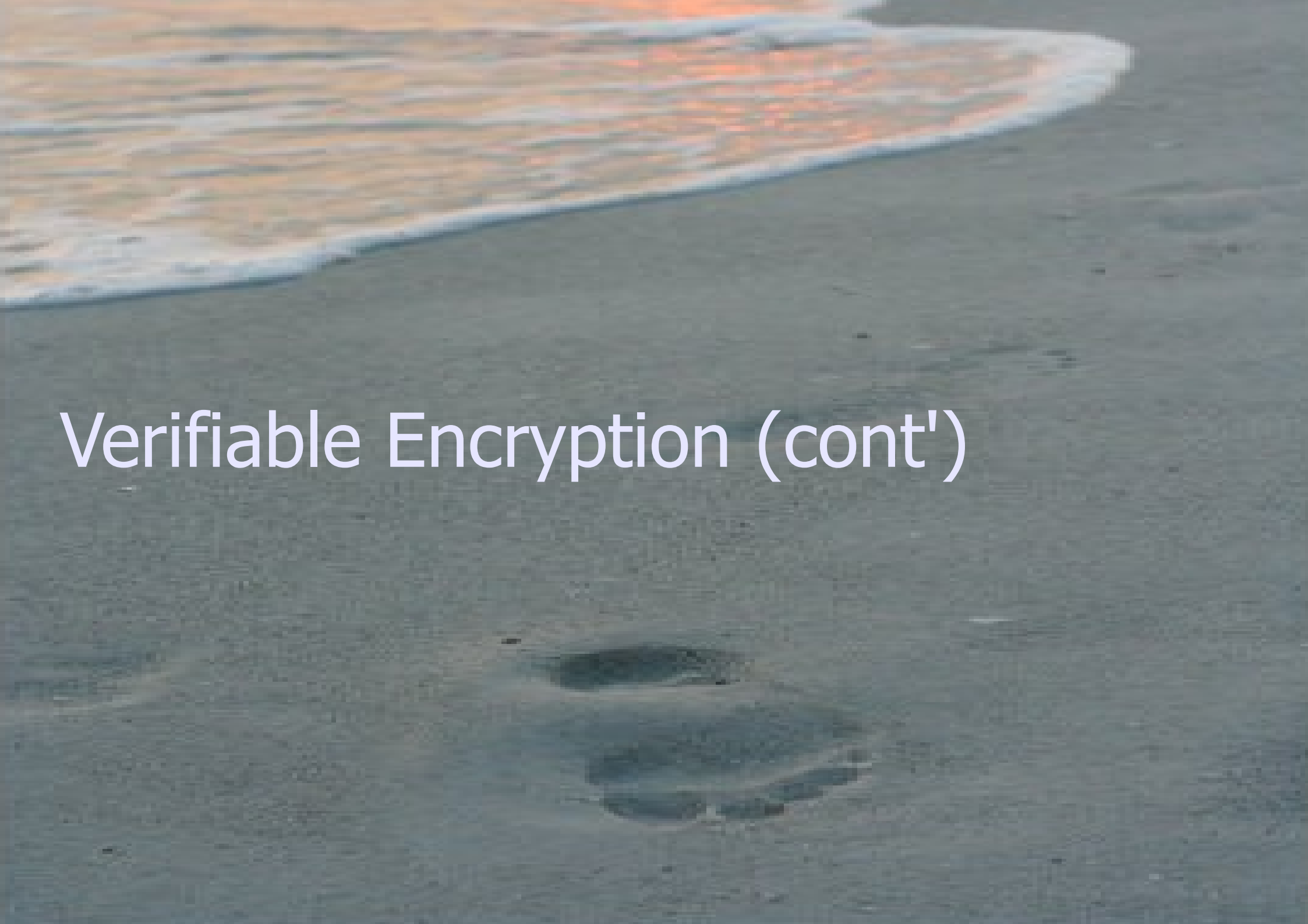
Assume: user has traditional government issued signing key and certificate (aka eID).

Then:

- User generates pseudonym P and *signs it* with gov't issued signing key, registers pseudonym and sign with issuer.
- Gets credential issued on pseudonym.
- Presents credential with tracing enabled (encryption of pseudonym) – *needs to be non-interactive presentation/signature.*
- Tracers claims this was the user holding pseudonym P????

Convincing a judge:

- Verify users' non-interactive presentation proof
- Tracer needs to prove that encryption in presentation proof indeed contains P
- Issuer needs provide user's signature with gov't issued key on P



Verifiable Encryption (cont')



Camenisch-Shoup Encryption
(for encryption of attributes)

The Decision Composite Residuosity Ass.

- Introduced by Paillier '99.
- If $n = (2p'+1)(2q'+1)$ then $Z^*_{n^2} = Z_2 \times Z_2 \times Z_n \times Z_{p'q'}$
- $(1+n)^u = (1+un) \bmod n^2$.
- Order of $(1+n)$ is n : $(1+n)^u = (1+nn) = 1 \bmod n^2$.

The DCR Problem: Given n and x, decide whether or not

$$x \in (Z^*_{n^2})^n$$

i.e., whether or not is has a component in the subgroup of order $p'q'$.

Camenisch-Shoup Encryption Scheme

Public Key: n and $g, y_1, y_2, y_3 \in \langle (g')^{2n}$, where $g' \in Z^*_{n^2}$,

Secret Key: $x_i = \log_g y_i$

Encryption of message $m \in [0,n]$ under label L:

- choose $r \in [0,n/4]$
- $u := g^r$, $w := y_1^r (1+n)^m$, $v := \text{abs}(y_2 y_3^{H(u,e,L)})^r$
- output (u,w,v).

where abs() maps (a mod n^2) to (n^2 - a mod n^2) if $a > n^2/2$, and (a mod n^2) otherwise, where $0 < a < n^2$.

Camenisch-Shoup Encryption Scheme

Decryption of ciphertext (u,e,v) under label L:

- verify $v = \text{abs}(v)$ and $u^{2(x_2 + H(u,w,L)x_3)} = v^2$.
- $\hat{c} := (w/u^{x_1})^{2t}$ where $t = 2^{-1} \bmod n$,
- if $n \mid (\hat{c}-1)$ output $m := (\hat{c}-1)/n$, otherwise output $\perp$

Intuition: remember $(1+n)^a = 1+an \pmod{n^2}$

so $(w/u^{x_1}) = y_1^r (1+n)^m / (g^r)^{x_1} = (1+n)^m = 1+mn$)

Theorem: *Encryption scheme is secure against adaptively chosen ciphertext attacks under DCR assumption.*

Verifiable Encryption of a Discrete Log

Let (c,e,s) be a credential, i.e., $d = a_1^{sk}\, a_2^{name}\, b^s\, c^e \pmod n$ and (u,v,w) be an encryption of name.

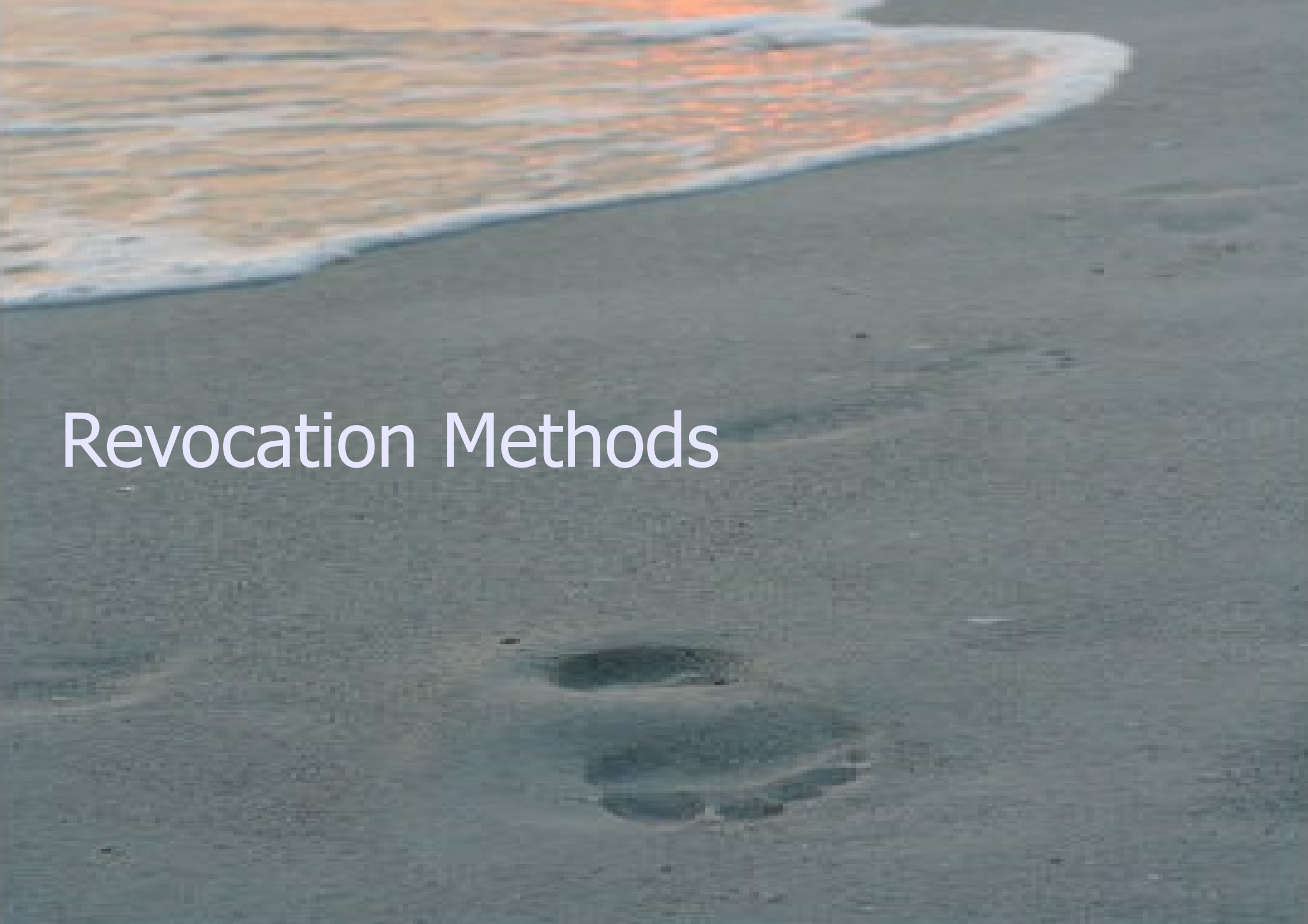
To prove that (u,v,w) indeed encrypts name:

$$SPK\{(\varepsilon, \mu1, \mu2, \rho, \sigma) :$$

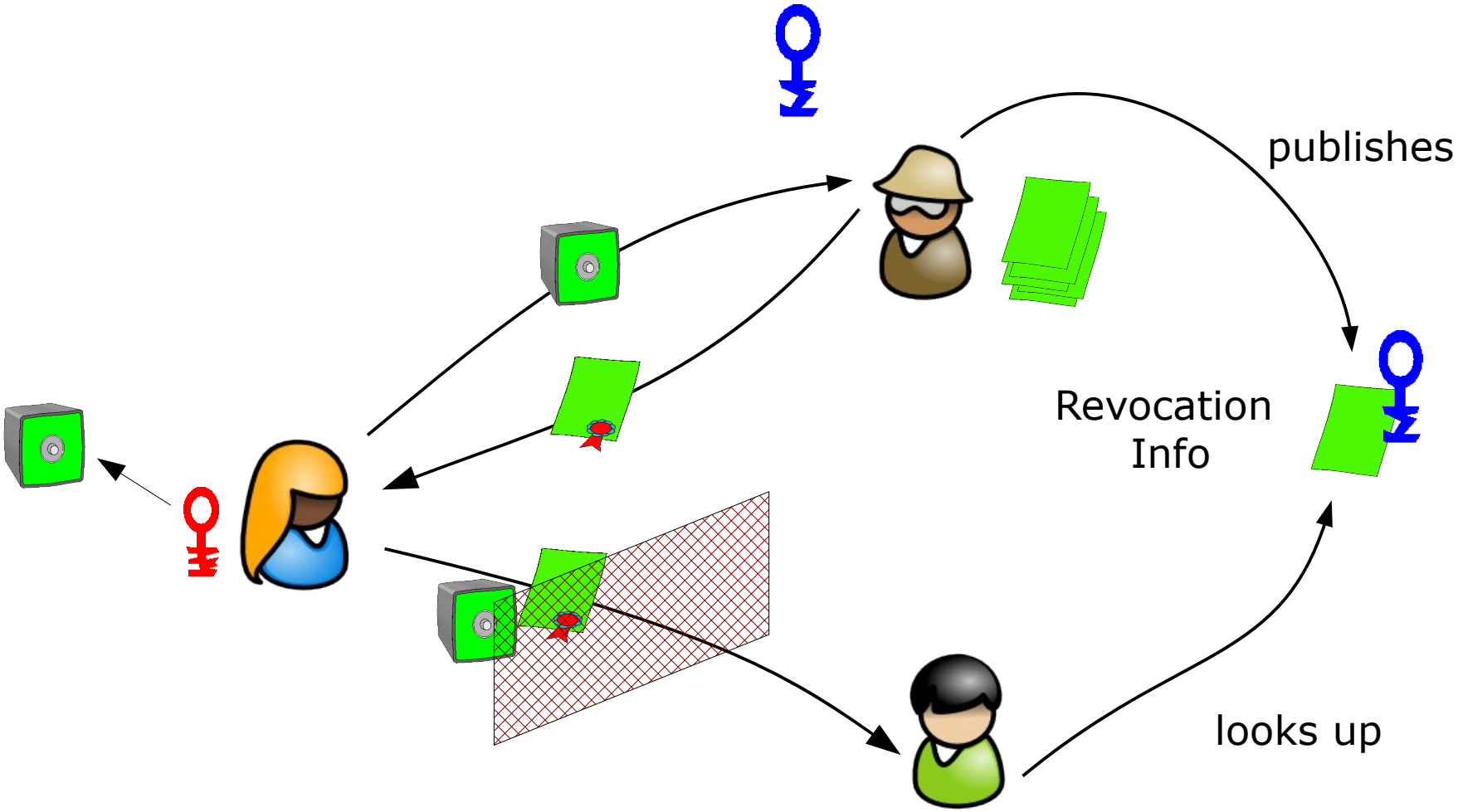
$$d := c'^{\varepsilon}\, a_1^{\mu1}\, a_2^{\mu2}\, b^{\sigma} \;\wedge\; \mu1, \mu2 \in \{0,1\}^{\ell} \;\wedge$$

$$u^2 = g^{2\rho} \;\wedge\; w^2 = y_1^{2\rho}\,(1+n)^{2\mu2} \;\wedge\; v^2 = (y_2 y_3^{H(u,e,L)})^{2\rho} \;\}$$

Note: does not mean that the person "name" indeed produced this proof. E.g., issuer could also have done this.


Revocation Methods

Revocable Credentials: Scenario



Alice should be able to convince verifier that her credential is not revoked (yet)!

Different Methods

CRL based methods

- traditional serial number & proof of non-membership
  - Best method uses signatures on pairs of succeeding revoked serial numbers
- Accumulator based solution
  - Presentation proof is efficient, but users need to update for each revoked credential

Short lived credentials

- Re-issuing (can always be done done, but requires interaction....)
- Publishing updates of credentials (define validity epochs), compute updates off-line

Verifier-Local “revocation”

- Essentially uses domain pseudonym and

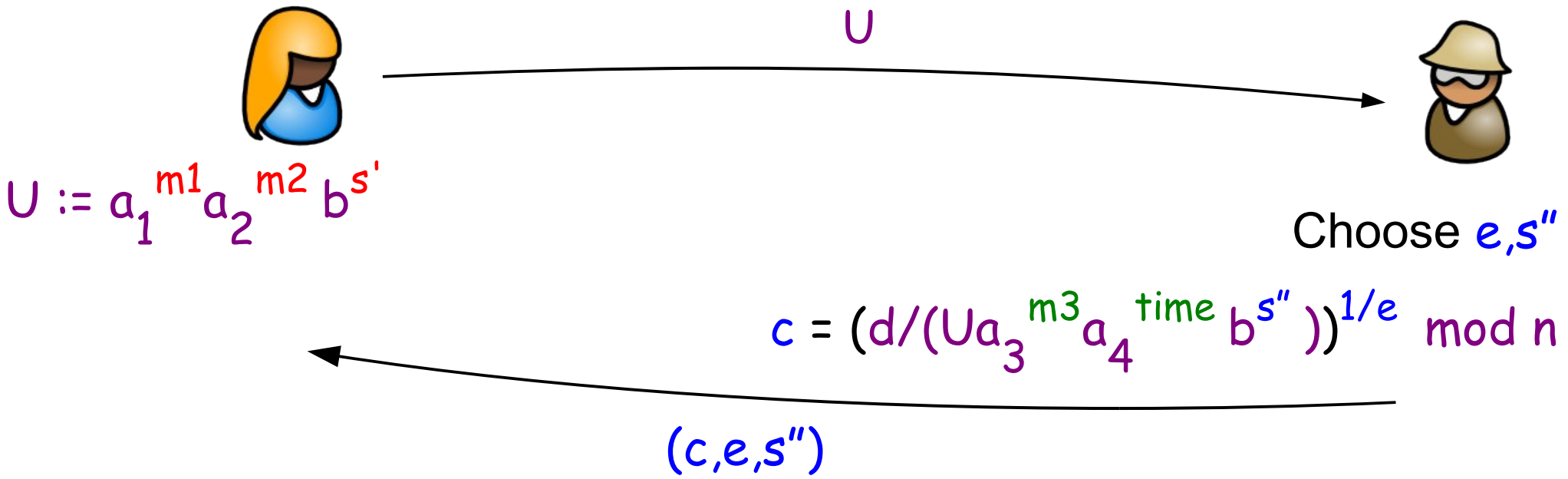
Notes:

- Choice on method depend on use case and can also be combined
- All methods work for both U-Prove and idemix signatures - excepts off-line credential update does not work for U-Prove signature scheme (needs interaction)

Revocation: Zeroth Solution

Re-issue certificates (off-line – interaction might be too expensive)

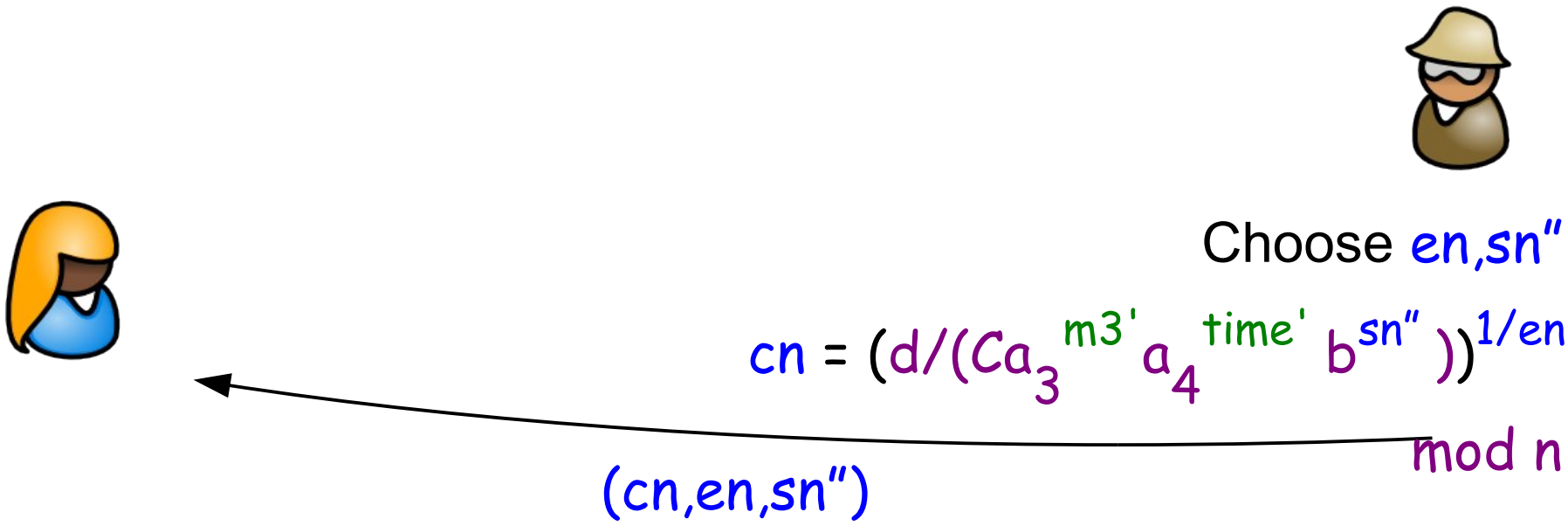
- Recall issuing for identity mixer:



Revocation: Zeroth Solution

Re-issue certificates (off-line – interaction might be too expensive)

- Idea: just repeat last step for each new time time':



- Update information (cn,en,sn'') can be pushed to user by many different means

Revocable Credentials: First Solution

- Include into credential some credential ID # as message, e.g.,

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

- Publish list of all valid (or invalid) #'s.

(#1,..., #k)

- Alice proves that her # is (or is not) on the list.
  - Compute $U_j = g^{\#_j}$ for #j in (#1,..., #k)
  - Prove $PK\{(\varepsilon, \mu, \rho, \sigma) : d = c'^{\varepsilon} a_1^{\rho} a_2^{\mu} b^{\sigma} \pmod{n} \wedge$

$$(U_1 = g^{\mu} \vee \cdots \vee U_k = g^{\mu})\}$$

- Not very efficient, i.e., linear in size k of list :-(

A better implementation of this idea where the issuer signs pairs of serial numbers (i.e., sig(#i,#i+1)) and have the user prove knowledge of sig(#i,#i+1) such that #i < # < #i+1 (c.f. Anja's presentation).

Revocable Credentials: Second Solution

- Include into credential some credential ID #i as message, e.g.,

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

- Publish list of all *invalid* #i's.

  (#1,..., #k)

- Alice proves that her ui is on the list.
  - Choose random h and compute $U = h^{\#i}$
  - Prove $PK\{(\varepsilon, \mu, \rho, \sigma) : d = c'^{\varepsilon} a_1^{\rho} a_2^{\mu} b^{\sigma} \pmod n \wedge U = h^{\mu}\}$
  - Verifier checks whether $U = h^{\#j}$ for all $\#j$ on the list.

- Better, as *only verifier* needs to do linear work (and it can be improved using so-call batch-verification...)
- What happens if we make the list of all valid #i's public?

Revocable Credentials: Second Solution

Variation: verifier could choose h and keep it fixed for a while

- Can pre-compute list $U_i = h^{x_i}$
- -> single table lookup

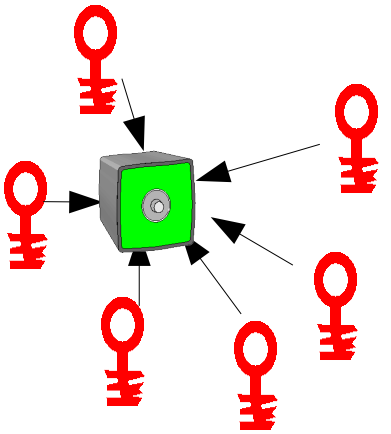
- BUT: if user comes again, verifier can link!!!
- ALSO: verifier could not change h at all! or use the same as other verifiers!
  - one way out h = H(verifier, date), so user can check correctness.
  - date could be the time up to seconds and the verifier could just store all the lists, i.e., pre-compute it.

Note: This is the method implemented in TPM's Direct Anonymous Attestation, where #i is a secret only known to the user. Thus credentials can only be revoked if TPM is broken and secrets are extracted – which is exactly why we want to revoke in this case.

Revocable Credentials: Third Solution

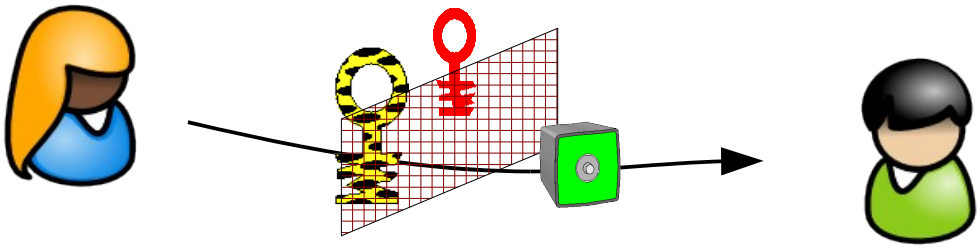
Using so-called cryptographic accumulators:

- Accumulate:

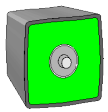


- Prove that your key is in accumulator: requires witness

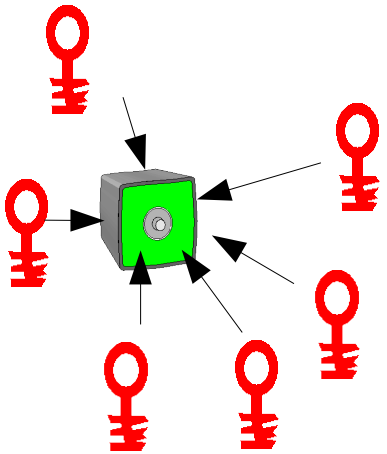




- Security: must work only for values contained in acc.



Revocable Credentials: Third Solution



Using so-called cryptographic accumulators:

- Key setup: RSA modulus n, seed v
- Accumulate:
  - values are primes e_i
  - accumulator value: $z = v^{\prod e_i} \bmod n$
  - publish z and n
  - witness value x for e_j : s.t. $z = x^{e_j} \bmod n$
    can be computed as $x = v^{e_1 \cdot \ldots \cdot e_{j-1} \cdot e_{j+1} \cdot \ldots \cdot e_k} \bmod n$

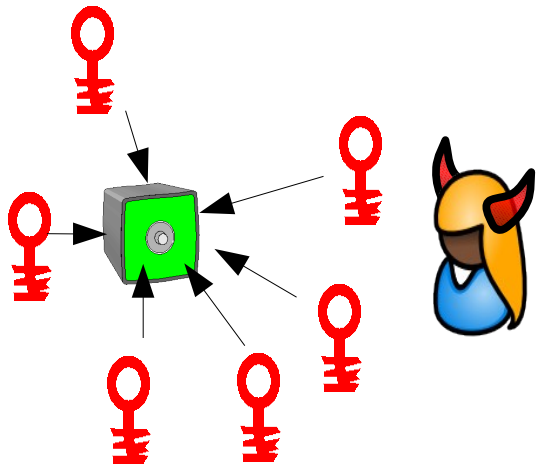


- Show that your value e is contained in accumulator:
  - provide x for e
  - verifier checks $z = x^{e} \bmod n$

Revocable Credentials: Third Solution

Security of accumulator: show that e s.t. $z = x^e \bmod n$ for e that is not contained in accumulator:

- For fixed e: Equivalent to RSA assumption
- Any e: Equivalent to Strong RSA assumption



Revocation: Each cert is associated with an e and each user gets witness x with certificate. But we still need:

- Efficient protocol to prove that committed value is contained in accumulator.
- Dynamic accumulator, i.e., ability to remove and add values to accumulator as certificates come and go.

Revocable Credentials: Third Solution

- Prove that your key is in accumulator:
  - choose random s and g and compute $U1 = x\, h^s$ (where h is a publicly known value such that it is assured that x lies in $\langle h \rangle$) and

    compute $U2 = g^s$ and reveal $U1$,$U2$, g

  - Run proof-protocol with verifier

$$PK\{(\varepsilon, \mu, \rho, \sigma, \xi, \delta) :$$

$$d = c'^{\varepsilon}\, a_1^{\rho} a_2^{\mu}\, b^{\sigma} \pmod n \;\wedge\; z = U1^{\mu}(1/h)^{\xi} \pmod n$$

$$\wedge\; 1 = U2^{\mu}(1/g)^{\xi} \pmod n \;\wedge\; U2 = g^{\delta} \pmod n\}$$

Revocable Credentials: Third Solution

- Analysis
  - No information about x and e is revealed:
    - $(U1, U2)$ is a secure commitment to x
    - proof-protocol is zero-knowledge
  - Proof is indeed proving that e contained in the certificate is also contained in the accumulator:

    a) $1 = U2^{\mu}(1/g)^{\xi} = (g^{\delta})^{\mu}(1/g)^{\xi} \pmod{n}$

    $\Rightarrow \xi = \delta\mu$

    b) $z = U1^{\mu}(1/v)^{\xi} = U1^{\mu}(1/v)^{\delta\mu} = (U1/v^{\delta})^{\mu} \pmod{n}$

    c) $d = c'^{\varepsilon} a_1^{\rho} a_2^{\mu} b^{\sigma} \pmod{n}$

Revocation: Third Solution

Dynamic Accumulator

- When a new user gets a certificate containing e_{new}
  - Recall: $z = v^{\prod e_i} \bmod n$
  - Thus: $z' = z^{e_{new}} \bmod n$
  - But: then all witnesses are no longer valid, i.e., need to be updated $x' = x^{e_{new}} \bmod n$

Revocation: Third Solution

Dynamic Accumulator

- When a certificate containing e_{rev} revoked
  - Now $z' = v^{\prod e_i} = z^{1/e_{rev}} \bmod n$
  - Witness:
    - Use Ext. Euclid to compute a and b
      s.t. $a\, e_{own} + b\, e_{rev} = 1$
    - Now $x' = x^{b} z'^{a} \bmod n$
    - Why: $x'^{e_{own}} = ((x^{b} z'^{a})^{e_{own}})^{e_{rev}\, 1/e_{rev}} \bmod n$

      $= ((x^{b} z'^{a})^{e_{own}\, e_{rev}})^{1/e_{rev}} \bmod n$

      $= ((x^{e_{own}})^{b\, e_{rev}} (z'^{e_{rev}})^{a\, e_{won}})^{1/e_{rev}} \bmod n$

      $= (z^{b\, e_{rev}} z^{a\, e_{own}})^{1/e_{rev}} \bmod n$

      $= z^{1/e_{rev}} \bmod n$

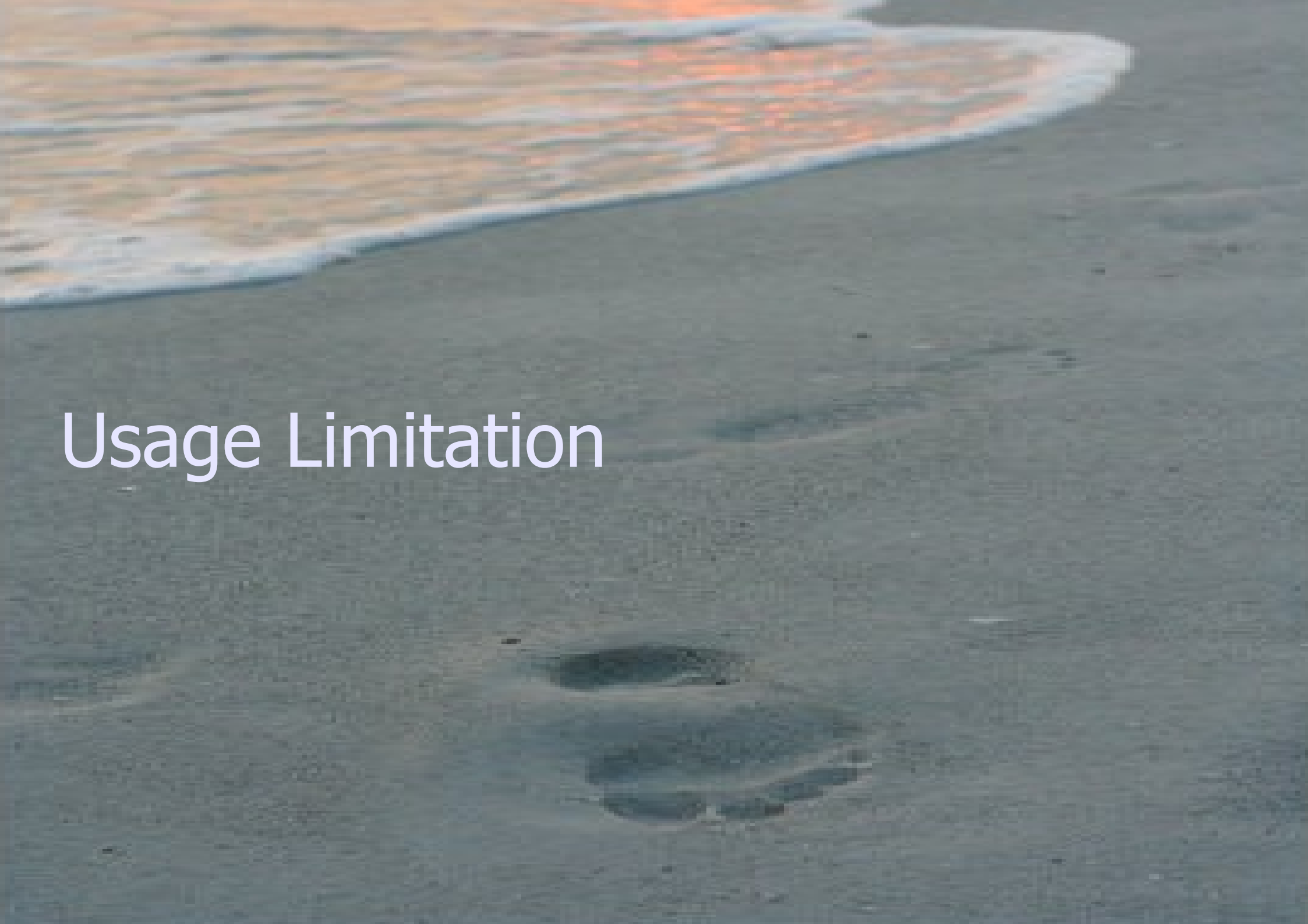
Revocation: Third Solution (improved)

Dynamic Accumulator: in case the issuer knows the factorization of n

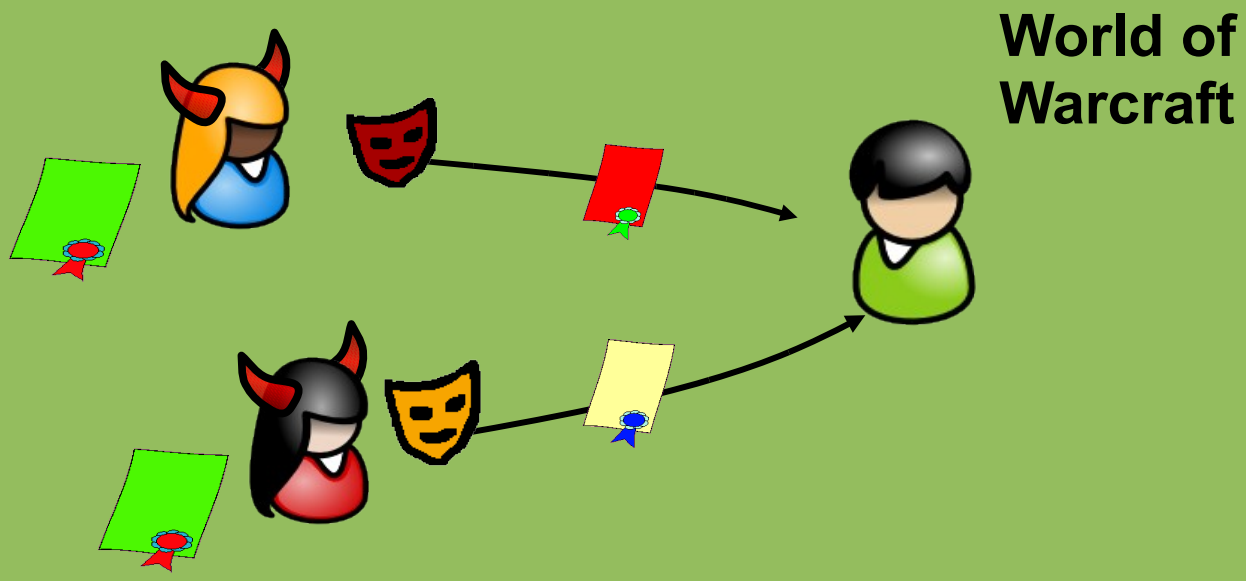
- When a new user gets a certificate containing e_{new}
  - Recall: $z = v^{\prod e_i} \bmod n$
  - Actually v never occurs anywhere...

    so: $v' = v^{1/e_{new}} \bmod n$ and $x = z^{1/e_{new}} \bmod n$
  - Thus z needs not to be changed in case new member joins!

- Witnesses need to be recomputed upon revocation only!


Usage Limitation

Other Properties: Cheating Prevention



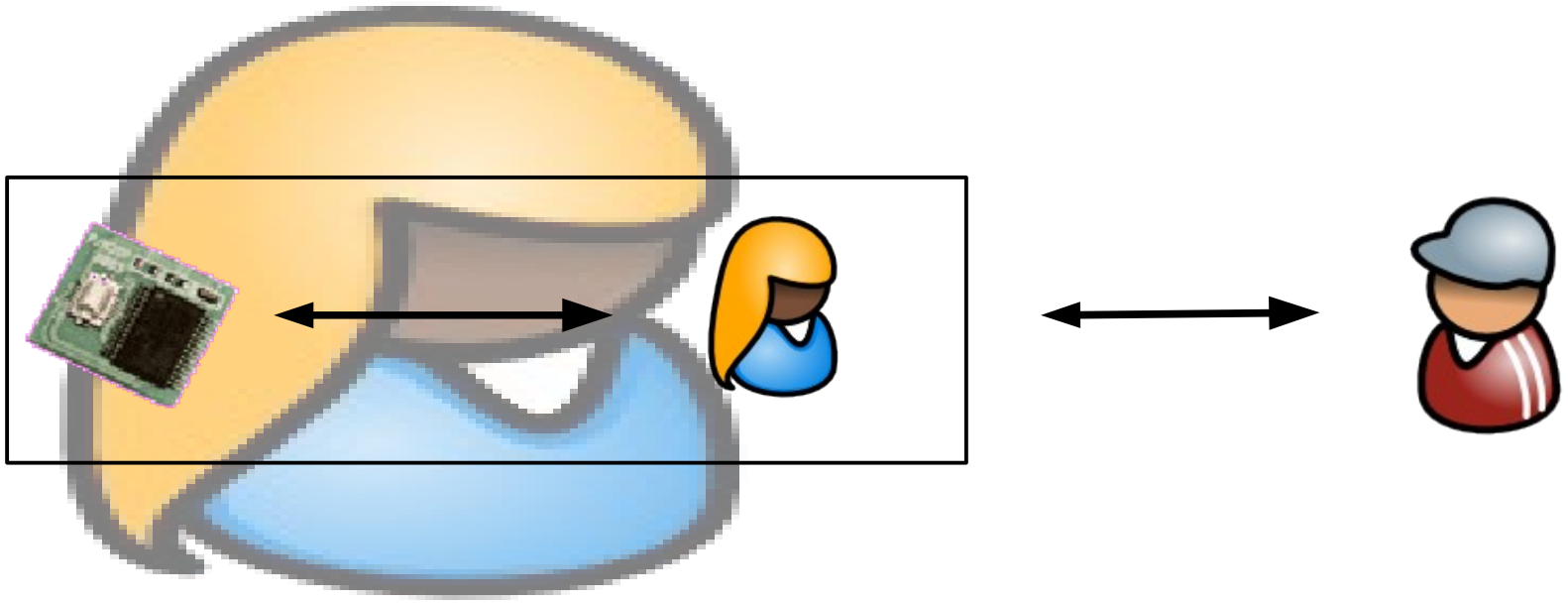
Limits of anonymity possible *(optional)*:

- 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)

Binding Credential to Hardware Token

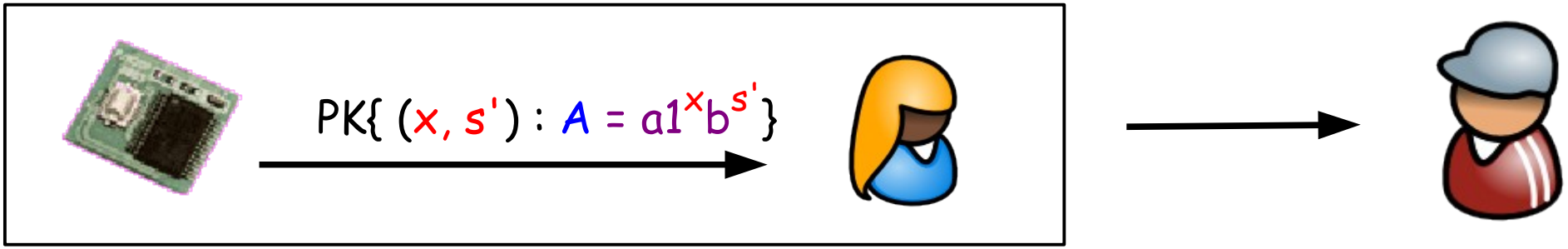
Idea:

- Store secret key (secret identity) of user on Hardware token
- Have hardware do all parts computation with secret key



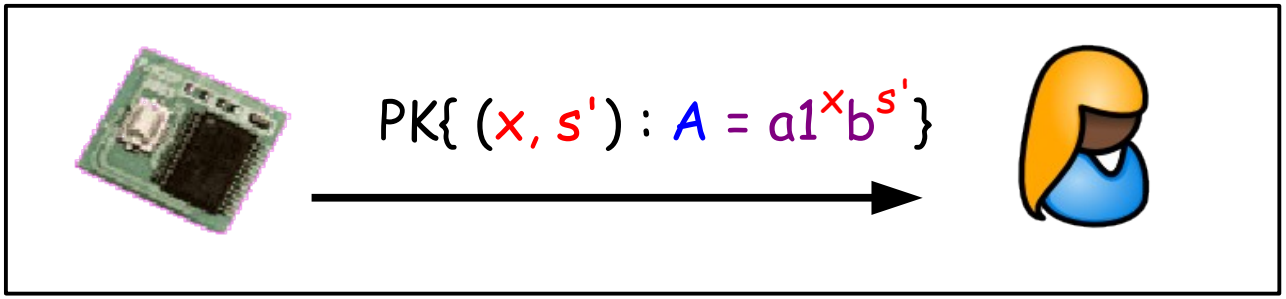
Binding Credential to Hardware Token

$$PK\{ (e, x, m, s^*) : \quad d = c'^{e} a1^{x} a2^{m} b^{s^*} \pmod{n} \}$$



Binding Credential to Hardware Token

$PK\{ (e, x, m, s^*) : \quad d = c'^{e} a1^{x} a2^{m} b^{s^*} \pmod{n} \}$



random r2, r3'

$t' = a1^{r2} b^{r3'}$

t'

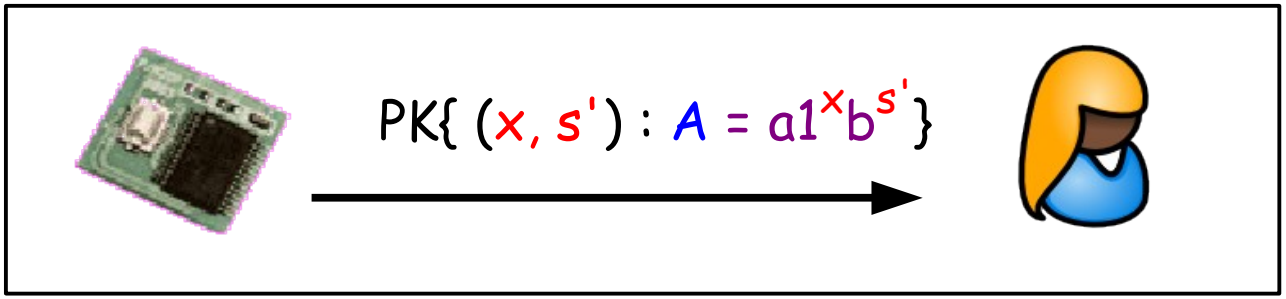
random r1, r3, r4

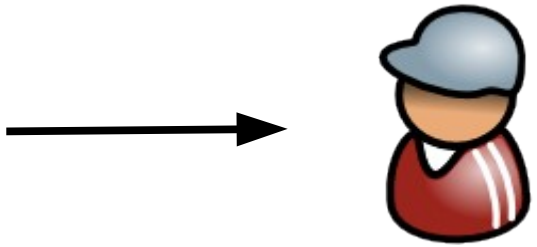
$t = t' c'^{r1} a2^{r3} b^{r4}$

t

Binding Credential to Hardware Token

$PK\{ (e, x, m, s^*) : \quad d = c'^{e} a1^{x} a2^{m} b^{s^*} \pmod{n} \}$





random $r2, r4'$

$t' = a1^{r2} b^{r4'}$

t'

random $r1, r3, r4$

$t = t' c'^{r1} a2^{r3} b^{r4}$

t

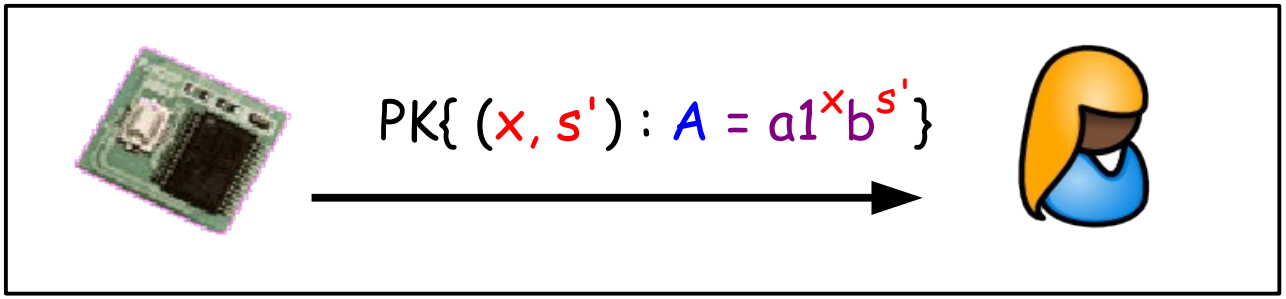
random c

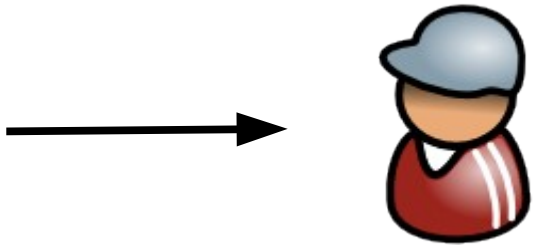
c

c

Binding Credential to Hardware Token

PK{ (e, x, m, s*) : $d = c'^{e} a1^{x} a2^{m} b^{s*} \pmod n$ }





random r2, r4'

t' = $a1^{r2} b^{r4'}$

t'

random r1, r3, r4

t = t' $c'^{r1} a2^{r3} b^{r4}$

t

random c

c

c

s2= r2 - c x

s4 = r4' - c s'

s2,s3

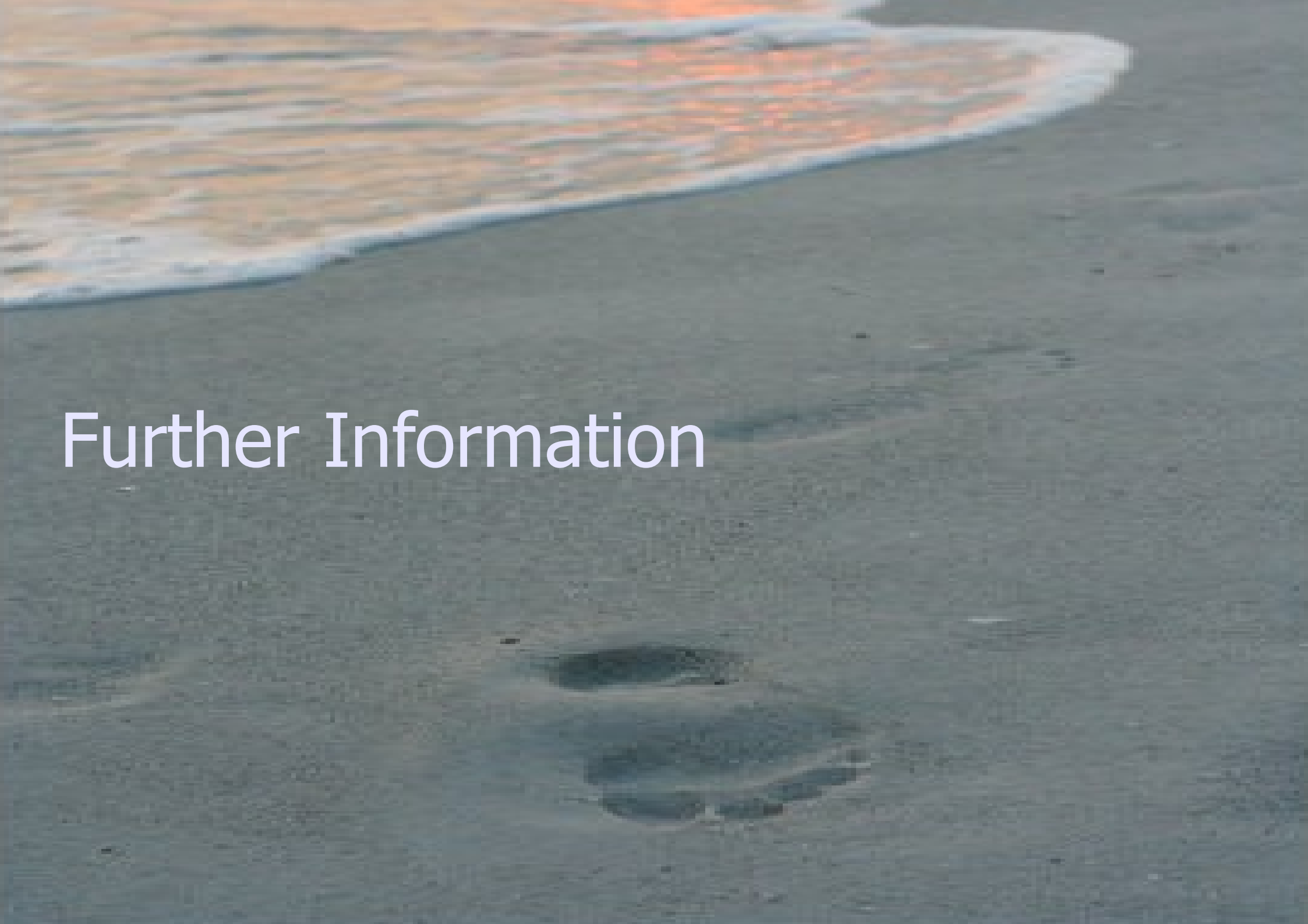
s1= r1 - c e

s3= r3 - c ê

s4 = - s4 + r4 - c ŝ

s1, -s2,s3

f ≟ $d^{c} c'^{s1} a1^{s2} a3^{s3} b^{s4}$?



Further Information

Further Information

- Video Version of Tutorial
  - https://abc4trust.eu/index.php/events/67-credential-tutorial
- Links
  - idemix.wordpress.com
  - www.zurich.ibm.com/idemix
  - www.abc4trust.eu
  - www.primelife.eu
  - www.prime-project.eu
- Code
  - http://www.primelife.eu/results/opensource/55-identity-mixer
  - http://www.primelife.eu/results/opensource/140-abcauth
- Contact
  - jca@zurich.ibm.com