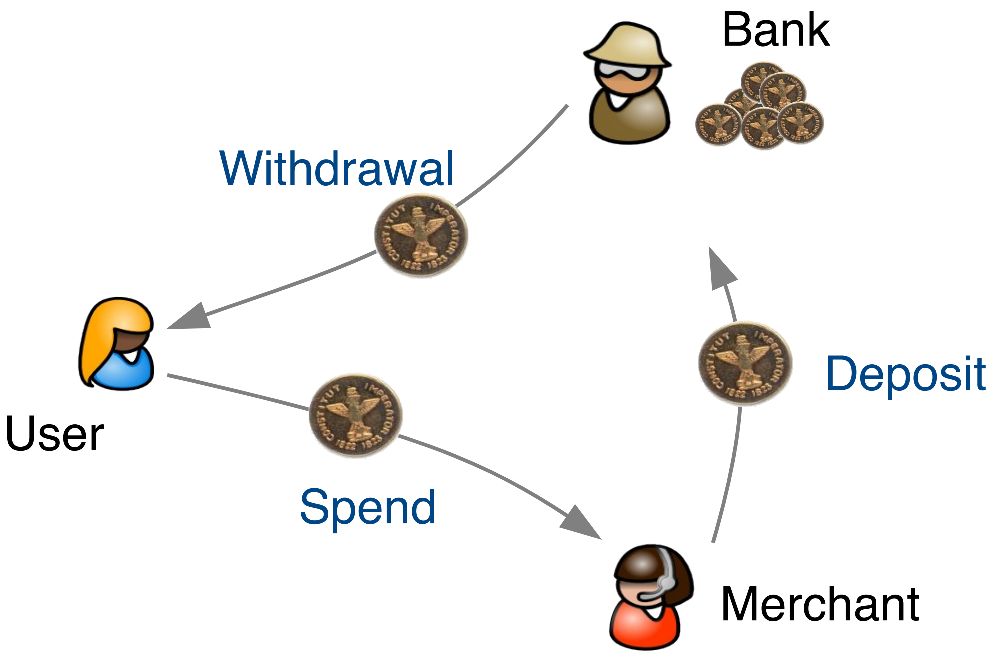
Cryptographic e-Cash

Jan Camenisch

IBM Research – Zurich
@JanCamenisch
ibm.biz/jancamenisch

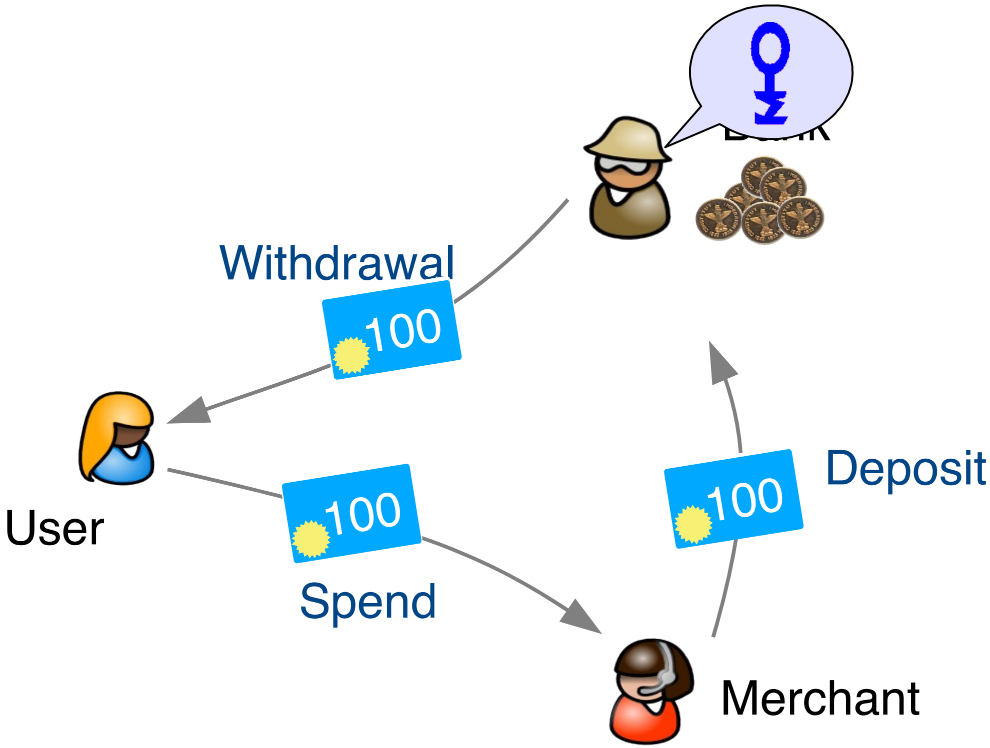
eCash scenario & requirements



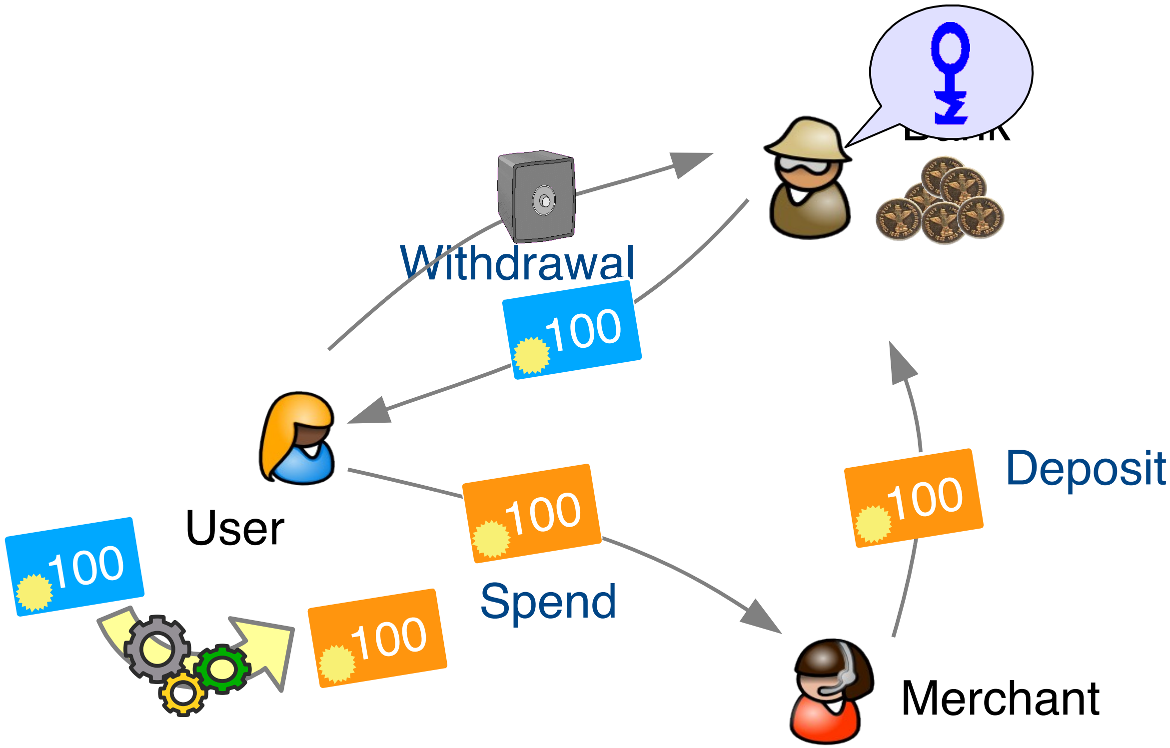
Requirements

- Anonymity: Withdrawal and Deposit must be unlinkable
- No Double Spending: Coin is bit-strings, can be spend twice

Towards a Solution: do it like paper money

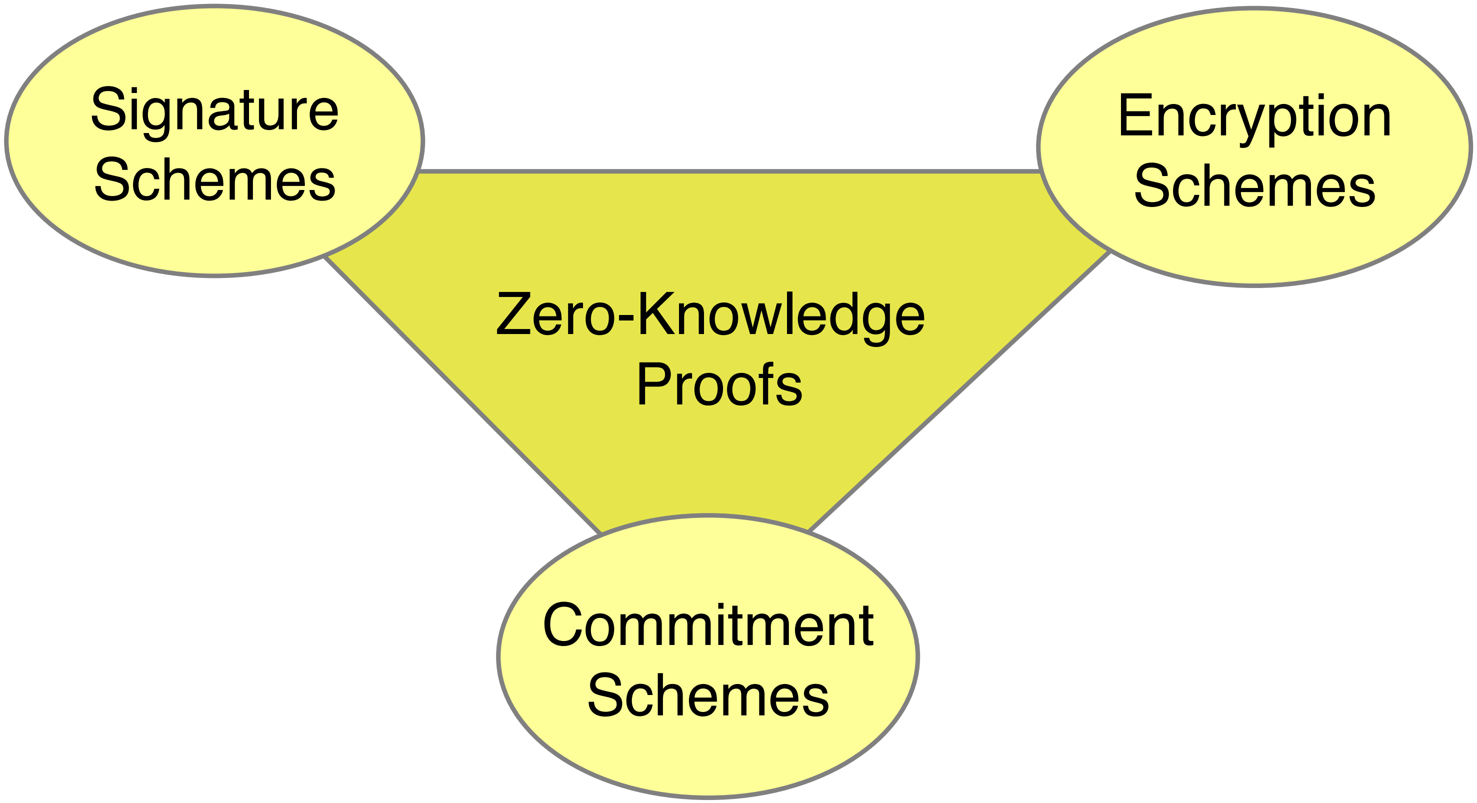


- Sign notes with digital signature scheme
  - Note = (serial number #, value)
  - Secure because
    - signature scheme can not be forged
    - bank will accepts some serial number only once → on-line e-cash
  - *Not* anonymous because (cf. paper solution)
    - bit-string of signature is unique
    - serial number is unique



- Use (more) cryptography
  - Hide serial number from bank when issuing
    - e.g., sign commitment of serial number
  - Reveal serial number and proof
    - knowledge of signature on
    - commitment to serial number
  - Anonymous because of commitments scheme and zero-knowledge proof

How to implement this?



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

mathematical setting

A set G with operation ▫ is called a group if:

– closure

for all a,b, in G → a ▫ b in G

– commutativity

for all a,b, in G → a ▫ b = b ▫ a

– associativity

for all a,b,c, in G → (a ▫ b) ▫ c = a ▫ (b ▫ c)

– identity

there exist some e in G, s.t. for all a: a ▫ e = a

– invertibility

for all a in G, there exist a^{-1} in G: a ▫ a^{-1} = e

- *Example:*

  integers under addition (Z,+)={...,-2,-1,0,1,2,...} or (Zn,+) = {0,1,2,...,n-1}
  identity: e = 0
  inverse: a^{-1} = - a

Cyclic Groups

IBM

- exponentiation = repeated application of · , e.g., $a^3 = a \cdot a \cdot a$

- a group is cyclic if every element is power of some fixed element:
  - i.e., for each a in G, there is unique i such that $g^i = a$
  - g = generator of the group
  - define $g^0 = 1$ = identity element

$$G = \langle g \rangle = \{1 = g^0, g^1, g^2, \ldots, ., g^{q-1}\}$$

  - q = |G|= order of group
    if q is a prime number then G is cyclic
    → computation in exponents can be done modulo q:

$$g^i = g^{i \bmod q}$$

- computing with exponents:

$$g^{i+j} = g^i \cdot g^j \qquad g^{i-j} = g^i / g^j = g^i \cdot (g^j)^{-1}$$

$$g^{ij} = (g^i)^j \qquad g^{-i} = (g^{-1})^i = (g^i)^{-1}$$

given g and x it is easy to compute g^x, $g^{1/x}$, ...

given g^x and g^y it is easy to compute $g^x g^y = g^{x+y}$

Discrete Log Assumption

given g^x it is hard to *compute* x

Diffie-Hellman Assumption

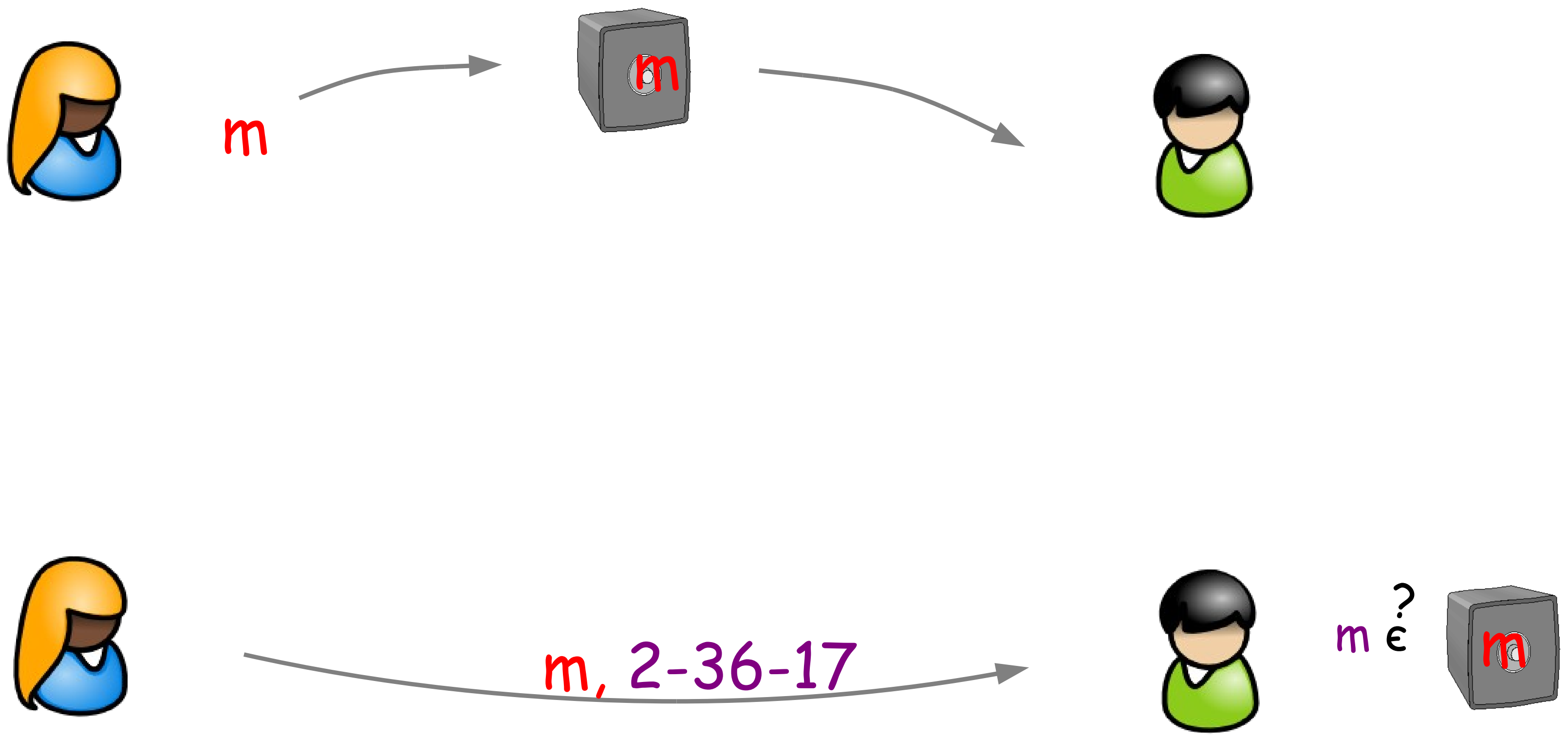
given g^x and g^y it is hard to *compute* g^{xy}

Decisional Diffie-Hellman Assumption

given g^x, g^y, and g^z it is hard to *decide* if $g^z = g^{xy}$

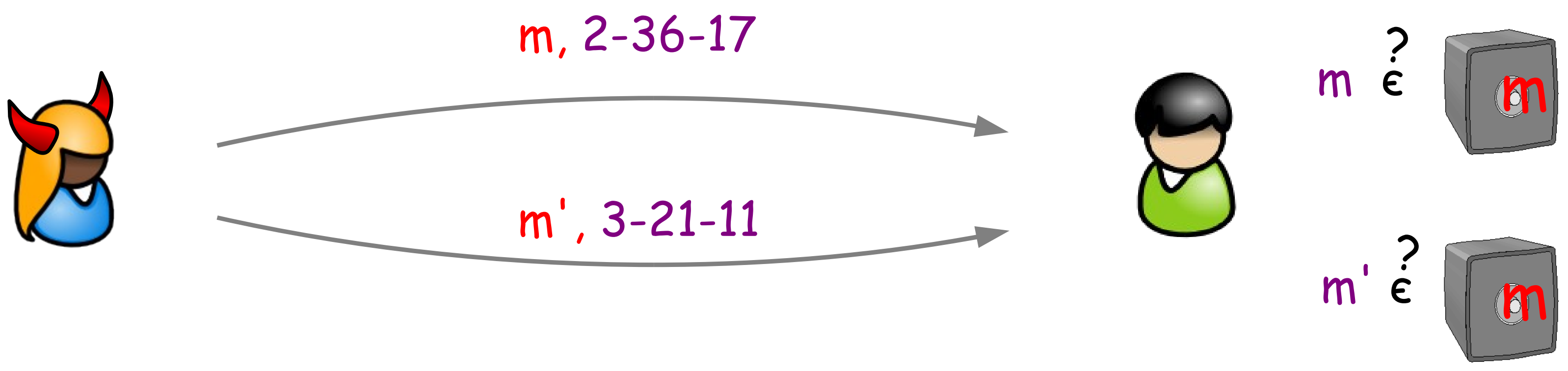

commitment scheme

Commitment Scheme: Functionality



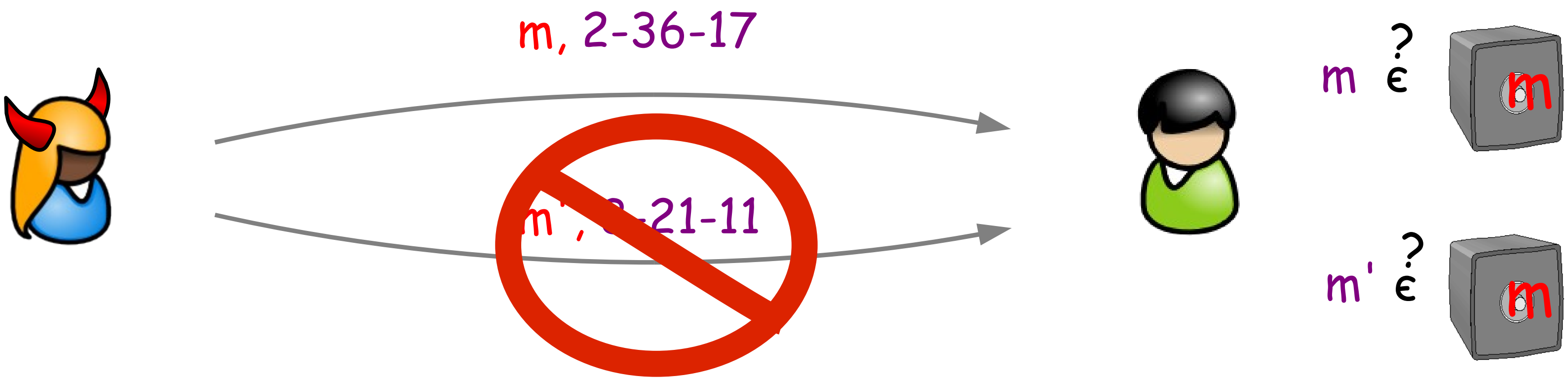
Commitment Scheme: Security

Binding



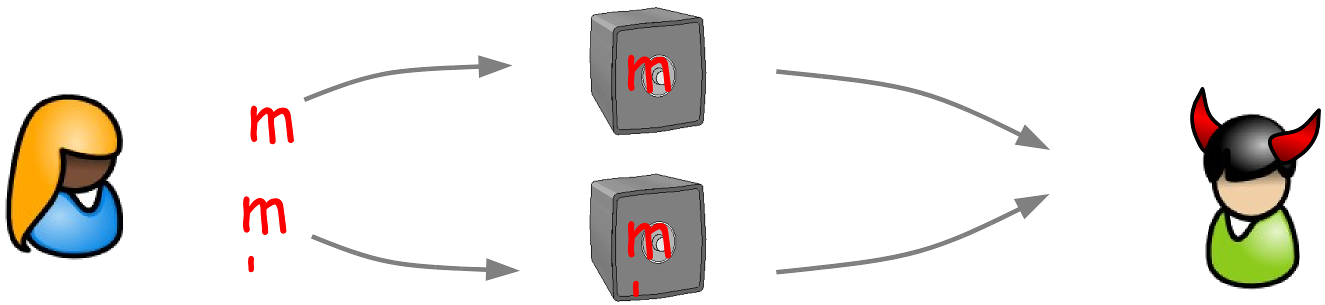
Commitment Scheme: Security

Binding



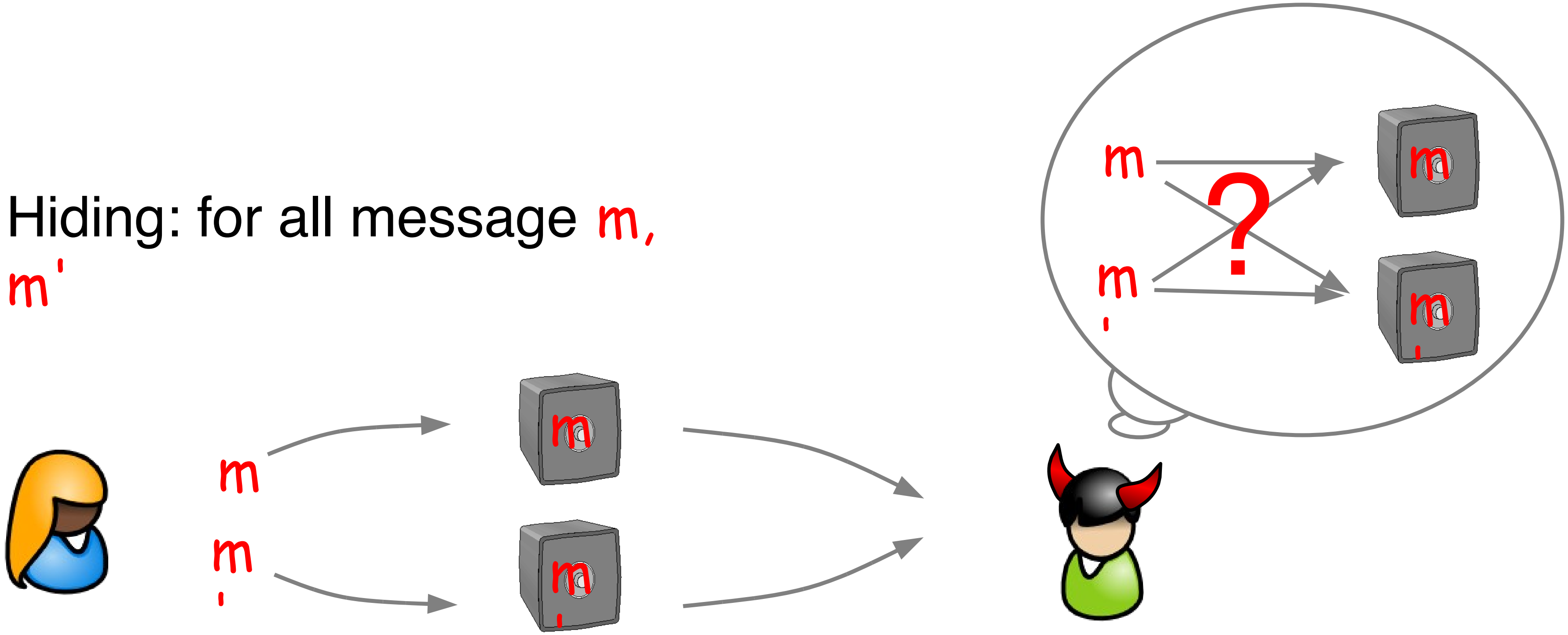
Commitment Scheme: Security

Hiding: for all message m, m'



Commitment Scheme: Security

Hiding: for all message m, m'



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 $c = g^x h^r$

Pedersen's Scheme:

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

Perfectly hiding:

Let c be a commitment and $u = \log_g h$

Thus $c = g^x h^r = g^{x+ur} = g^{(x+ur')+u(r-r')}$

$= g^{x+ur'} h^{r-r'}$ for *any* r'!

I.e., given c and x' here exist r' such that $c = g^{x'} h^{r'}$

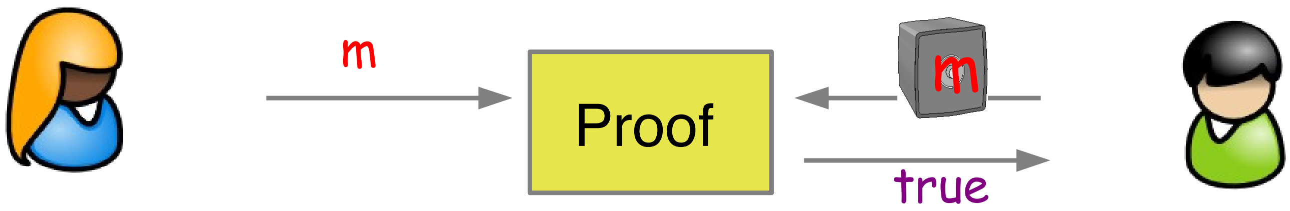
Computationally binding:

Let c, (x', r') and (x, r) s.t. $c = g^{x'} h^{r'} = g^x h^r$

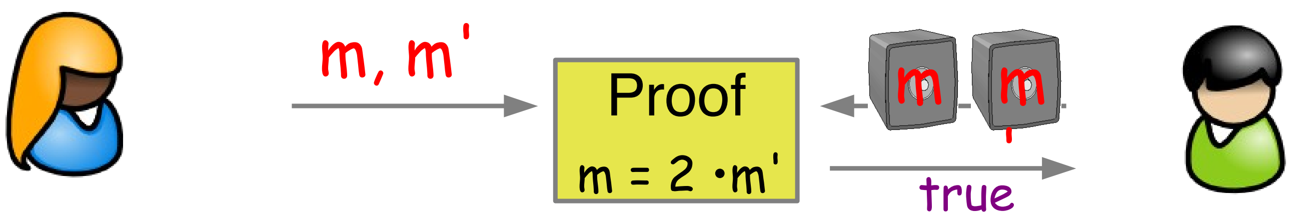
Then $g^{x'-x} = h^{r-r'}$ and $u = \log_g h = (x'-x)/(r-r') \bmod q$

Commitment Scheme: Extended Features

Proof of Knowledge of Contents



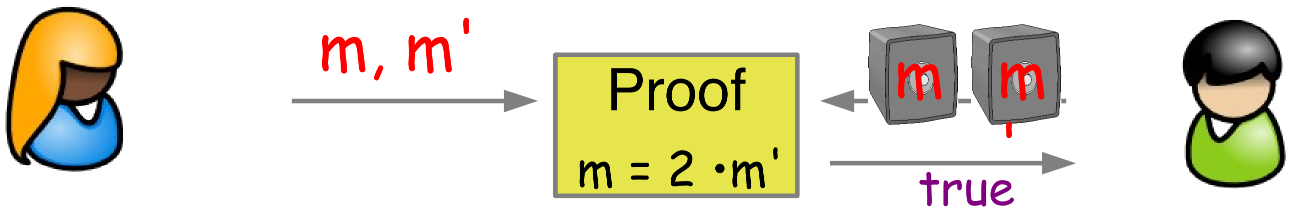
Proof of Relations among Contents



Commitment Scheme: Extended Features

Let $C1 = g^{m}h^{r}$ and $C' = g^{m'}h^{r}$ then:

$$PK\{(\alpha,\beta): \quad C = g^{\beta}h^{\alpha}\}$$



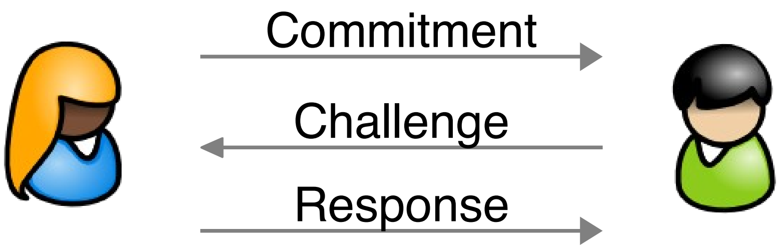
$$PK\{(\alpha,\beta,\gamma): \quad C' = g^{\beta}h^{\alpha} \wedge C = (g^{2})^{\beta}h^{\gamma}\}$$

zero-knowledge proofs

Zero-Knowledge Proofs

IBM

- interactive proof between a prover and a verifier about the prover's knowledge



- properties:

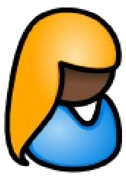
  zero-knowledge
    verifier learns nothing about the prover's secret

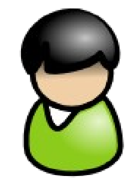
  proof of knowledge (soundness)
    prover can convince verifier only if she knows the secret

  completeness
    if prover knows the secret she can always convince the verifier

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

Prover wants to convince verifier that she *knows* x s.t. $y = g^x$
such that verifier only learns y and g.

x  Prover:

Verifier:  y, g

Prover		Verifier
random r		
$t := g^r$	t →	
	← c	random c
$s := r - cx$	s →	$t = g^s y^c$?

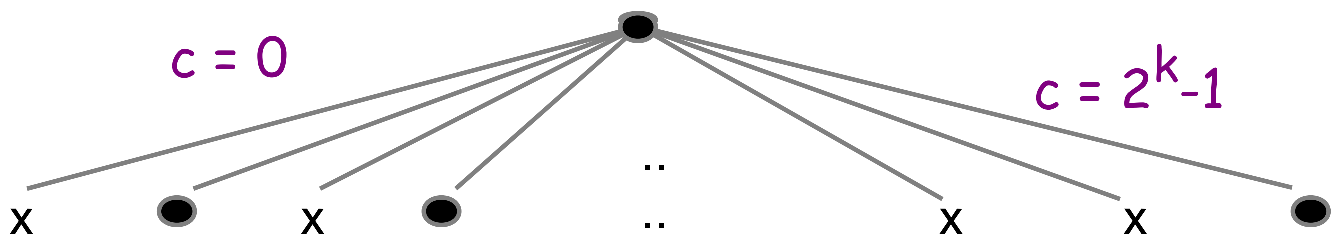
notation: $PK\{(\alpha): y = g^{\alpha}\}$

Zero Knowledge Proofs: Security

Proof of Knowledge Property:

If prover is successful with non-negl. probability, then she “knows” $x = \log_g y$,
i.e., ones can extract x from her.

Assume $c \in \{0,1\}^k$ and consider execution tree:



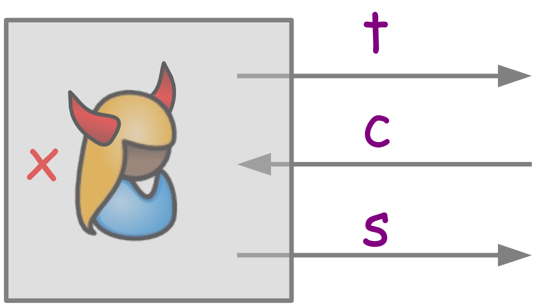
If success probability for any prover (including malicious ones) is $>2^{-k}$ then there are two *accepting* tuples (t,c1,s1) and (t,c2,s2) for the same t.

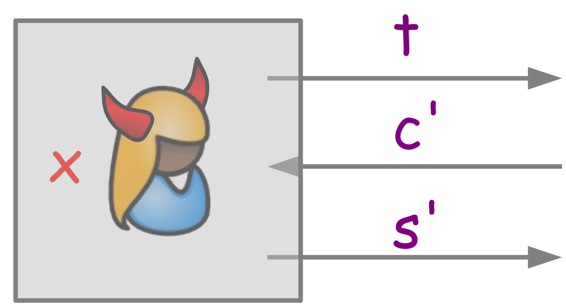
Zero Knowledge Proofs: Security

Prover might do protocol computation in any way it wants & we cannot analyse code.

Thought experiment:

- Assume we have prover as a black box → we can reset and rerun prover
- Need to show how secret can be extracted via protocol interface





$$t = g^{s} y^{c} = g^{s'} y^{c'} \quad \rightarrow \quad y^{c'-c} = g^{s-s'}$$

$$\rightarrow \quad y = g^{(s-s')/(c'-c)}$$

$$\rightarrow \quad x = (s-s')/(c'-c) \mod q$$

Zero Knowledge Proofs: Security

Zero-knowledge property:

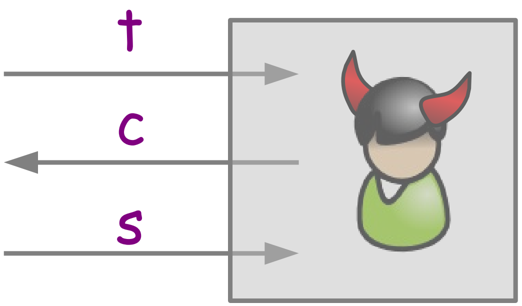
If verifier does not learn anything (except the fact that Alice knows $x = \log_g y$)

Idea: One can simulate whatever Bob "sees".

Choose random c', s'

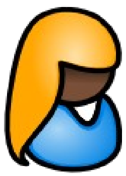
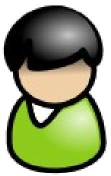
compute $t := g^{s'} y^{c'}$

if $c = c'$ send $s' = s$,
otherwise restart



Problem: if domain of c too large, success probability becomes too small

One way to modify protocol to get large domain c:

x  **Prover:** ——— **Verifier:**  y, g

Prover		Verifier
		random c, v
random r	$\xleftarrow{\;h\;}$	$h := H(c,v)$
$t := g^r$	$\xrightarrow{\;t\;}$	
$h := H(c,v)$?	$\xleftarrow{\;c,v\;}$	
$s := r - cx$	$\xrightarrow{\;s\;}$	$t = g^s y^c$?

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

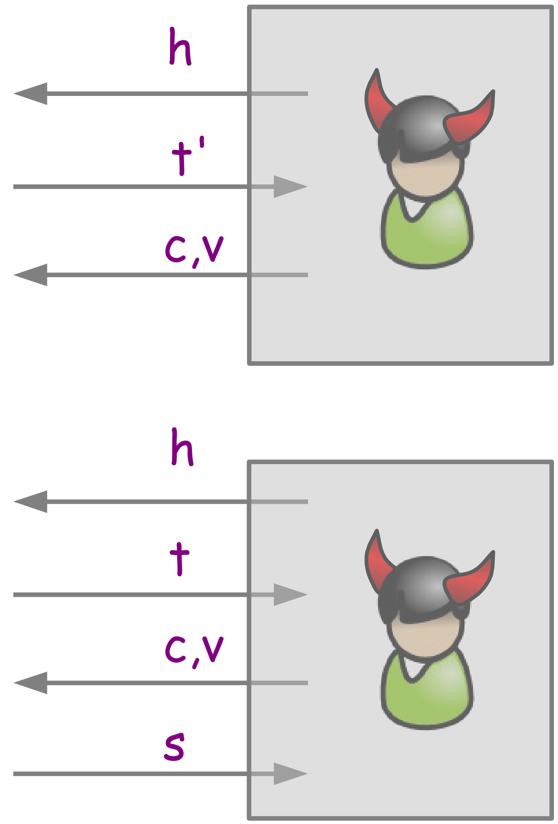
Zero Knowledge Proofs: Security

One way to modify protocol to get large domain c:

Choose random c', s'
compute $t' := g^{s'} y^{c'}$

after having received c "reboot" verifier

Choose random s
compute $t := g^{s} y^{c}$
send s



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

Signing a message m:

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

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

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

Security:

- underlying protocol is zero-knowledge proof of knowledge
- hash function $H(.)$ behaves as a "random oracle."

Many Exponents:

$$PK\{(\alpha,\beta,\gamma,\delta): \quad y = g^{\alpha} h^{\beta} z^{\gamma} k^{\delta} u^{\beta} \}$$

Logical combinations:

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

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

Intervals and groups of different order (under SRSA):

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

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

Non-interactive (Fiat-Shamir heuristic, Schnorr Signatures):

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

Let $g, h, C1, C2, C3$ be group elements.

Now, what does

$PK\{(\alpha1,\beta1,\alpha2,\beta2, \alpha3, \beta3): \quad C1 = g^{\alpha1}h^{\beta1} \wedge C2 = g^{\alpha2}h^{\beta2} \wedge C3 = g^{\alpha3}h^{\beta3} \wedge C3 = g^{\alpha1}g^{\alpha2}h^{\beta3}\}$

mean?

→ Prover knows values $\alpha1, \beta1, \alpha2, \beta2, \beta3$ such that

$C1 = g^{\alpha1}h^{\beta1}$, $C2 = g^{\alpha2}h^{\beta2}$ and

$C3 = g^{\alpha1}g^{\alpha2}h^{\beta3} = g^{\alpha1 + \alpha2}h^{\beta3} = g^{\alpha3}h^{\beta3}$

$\alpha3 = \alpha1 + \alpha2 \pmod q$

And what about:

$PK\{(\alpha1,\ldots,\beta3): \quad C1 = g^{\alpha1}h^{\beta1} \wedge C2 = g^{\alpha2}h^{\beta2} \wedge C3 = g^{\alpha3}h^{\beta3} \wedge C3 = g^{\alpha1}(g^{5})^{\alpha2}h^{\beta3}\}$

→ $C3 = g^{\alpha1}g^{\alpha2}h^{\beta3} = g^{\alpha1 + 5\alpha2}h^{\beta3}$

$\alpha3 = \alpha1 + 5\alpha2 \pmod q$

Some Example Proofs and Their Analysis

Let $g, h, C1, C2, C3$ be group elements.

Now, what does

$$PK\{(\alpha 1,..,\beta 3): \quad C1 = g^{\alpha 1}h^{\beta 1} \wedge C2 = g^{\alpha 2}h^{\beta 2} \wedge C3 = g^{\alpha 3}h^{\beta 3} \wedge C3 = C2^{\alpha 1}h^{\beta 3}\}$$ mean?

→ Prover knows values $\alpha 1, \beta 1, \alpha 2, \beta 2, \beta 3$ such that

$C1 = g^{\alpha 1}h^{\beta 1}$, $C2 = g^{\alpha 2}h^{\beta 2}$ and

$$C3 = C2^{\alpha 1}h^{\beta 3} = (g^{\alpha 2}h^{\beta 2})^{\alpha 1}h^{\beta 3} = g^{\alpha 2 \cdot \alpha 1}h^{\beta 3 + \beta 2 \cdot \alpha 1}$$

$$C3 = g^{\alpha 2 \cdot \alpha 1}h^{\beta 3 + \beta 2 \cdot \alpha 1} = g^{\alpha 3}h^{\beta 3'}$$

$$\alpha 3 = \alpha 1 \cdot \alpha 2 \pmod q$$

And what about

$$PK\{(\alpha 1,\beta 1\ \beta 2): \quad C1 = g^{\alpha 1}h^{\beta 1} \wedge C2 = g^{\alpha 2}h^{\beta 2} \wedge C2 = C1^{\alpha 1}h^{\beta 2}\}$$

→ $\alpha 2 = \alpha 1^2 \pmod q$

Let $g, h, C1, C2, C3$ be group elements.

Now, what does

$$PK\{(\alpha 1,..,\beta 2): \quad C1 = g^{\alpha 1} h^{\beta 1} \wedge C2 = g^{\alpha 2} h^{\beta 2} \wedge g = (C2/C1)^{\alpha 1} h^{\beta 2}\}$$ mean?

→ Prover knows values $\alpha, \beta 1, \beta 2$ such that

$$C1 = g^{\alpha 1} h^{\beta 1}$$

$$g = (C2/C1)^{\alpha 1} h^{\beta 2} = (C2\, g^{-\alpha 1} h^{-\beta 1})^{\alpha 1}\, h^{\beta 2}$$

→ $$g^{1/\alpha 1} = C2\, g^{-\alpha 1} h^{-\beta 1}\, h^{\beta 2/\alpha 1}$$

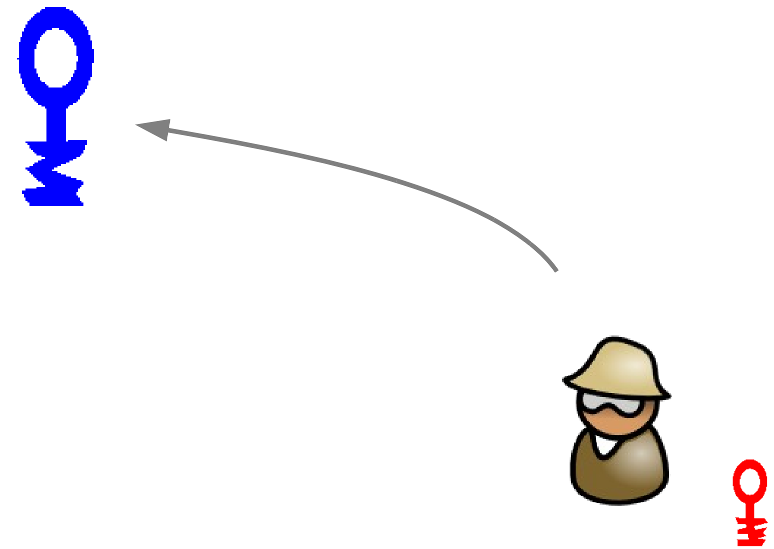
$$C2 = g^{\alpha 1}\, h^{\beta 1}\, h^{-\beta 2/\alpha 1}\, g^{1/\alpha 1} = g^{\alpha 1 + 1/\alpha 1}\, h^{\beta 1 - \beta 2/\alpha 1}$$

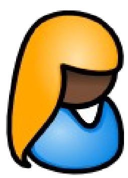
$$C2 = g^{\alpha 2}\, h^{\beta 2}$$

$$\alpha 2 = \alpha 1 + \alpha 1^{-1} \pmod q$$

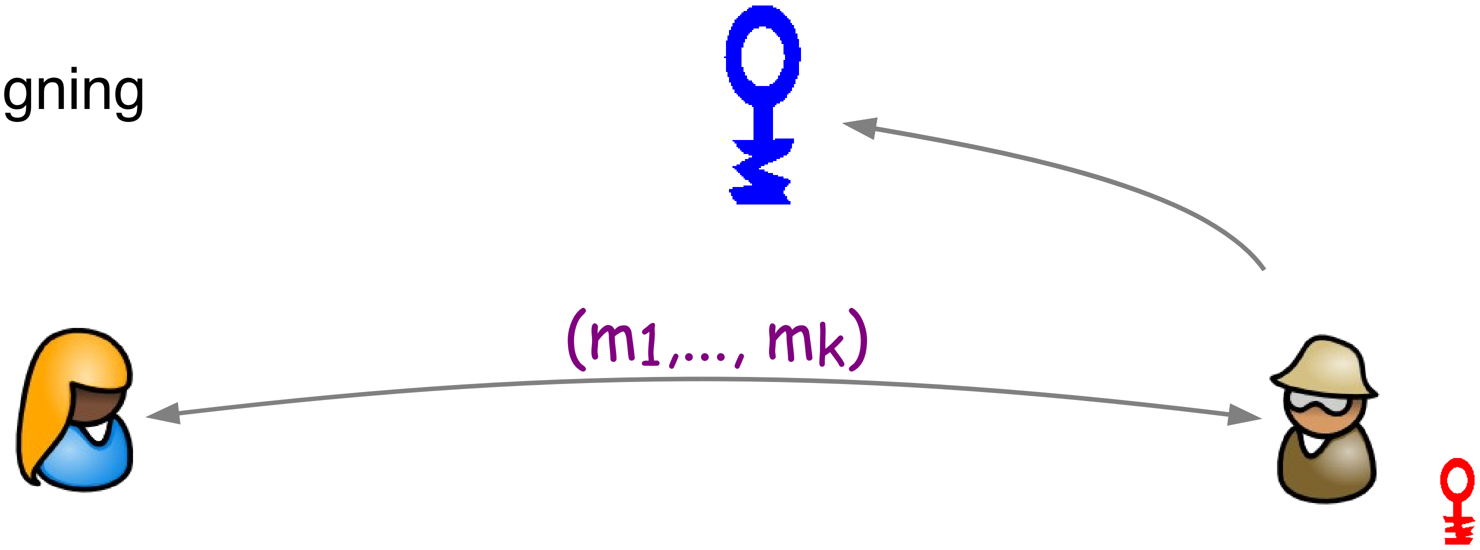
signature schemes

Key Generation





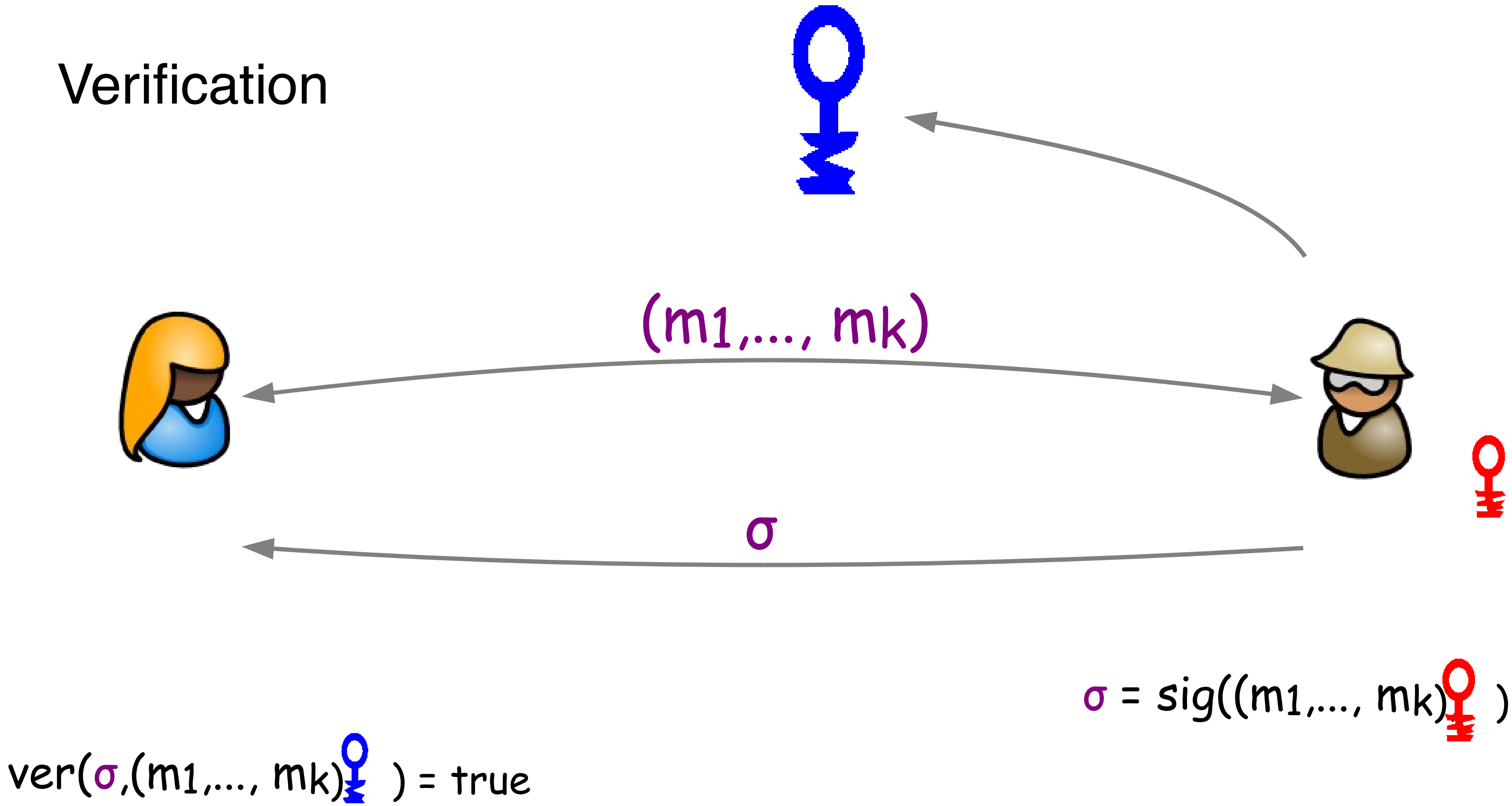
Signing



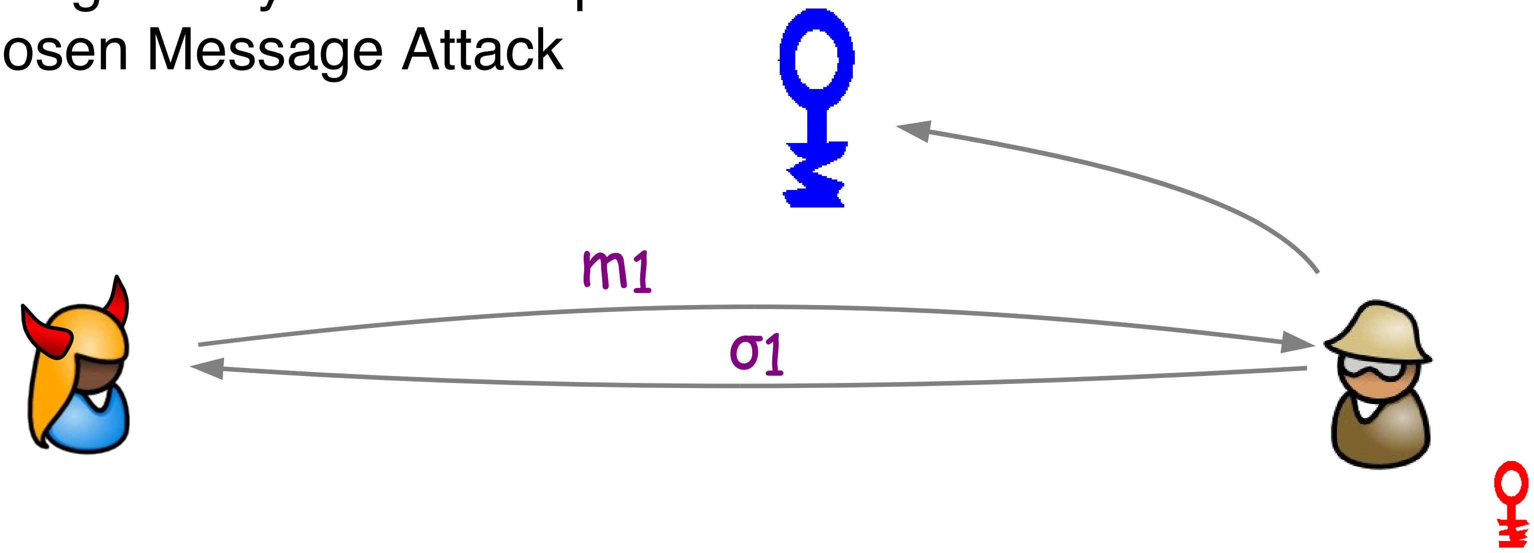
$\sigma = sig((m_1, \ldots, m_k)\ )$

Signature Scheme: Functionality

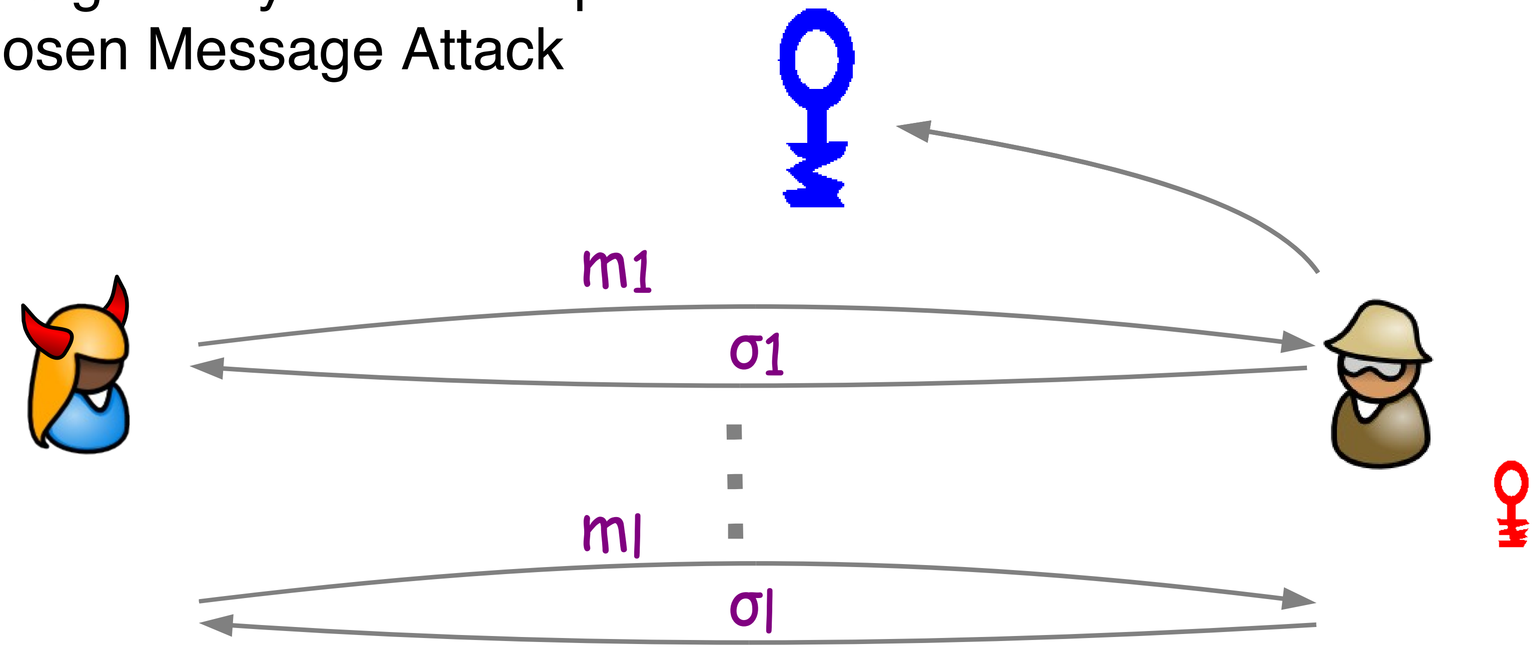
Verification



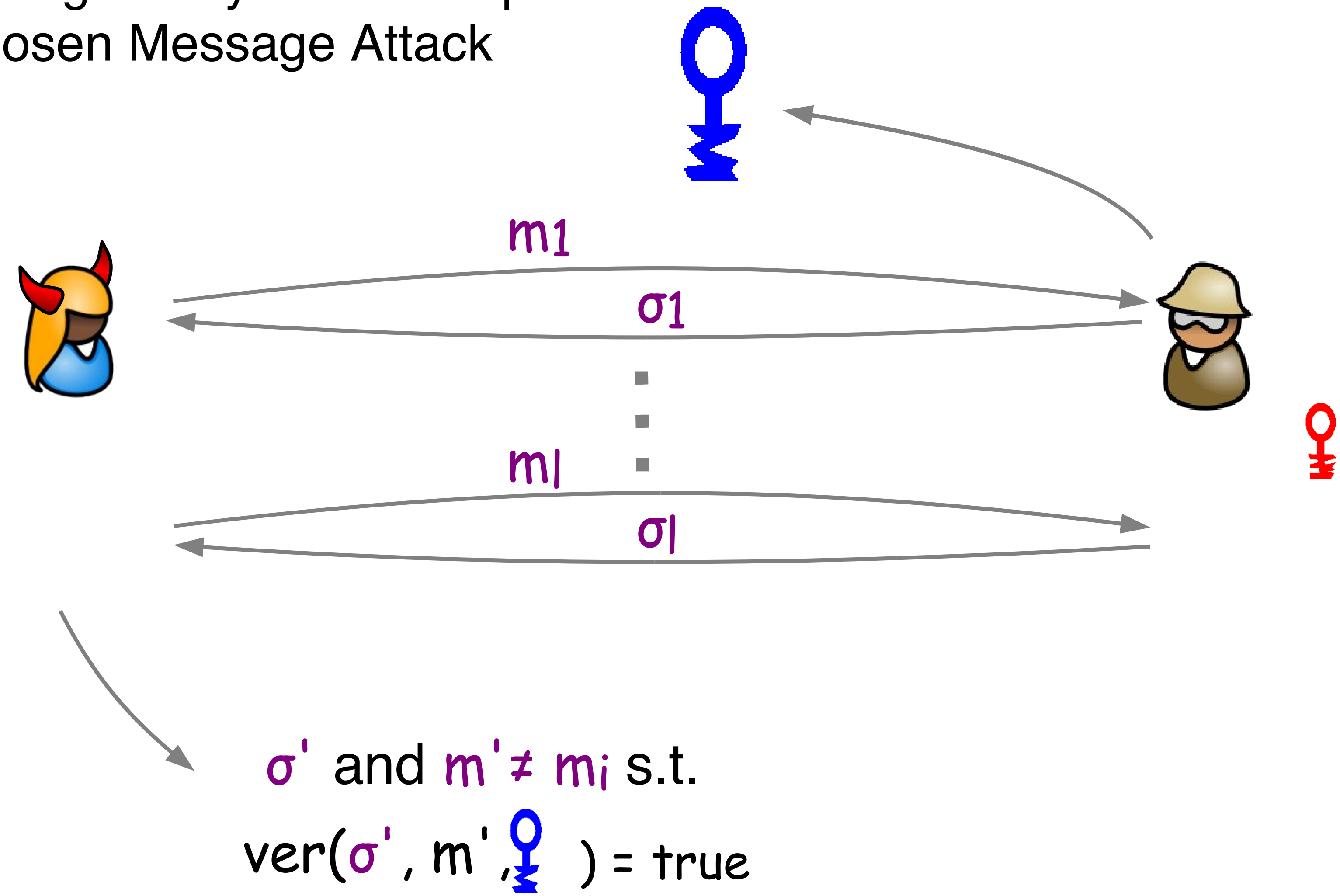
Unforgeability under Adaptive Chosen Message Attack



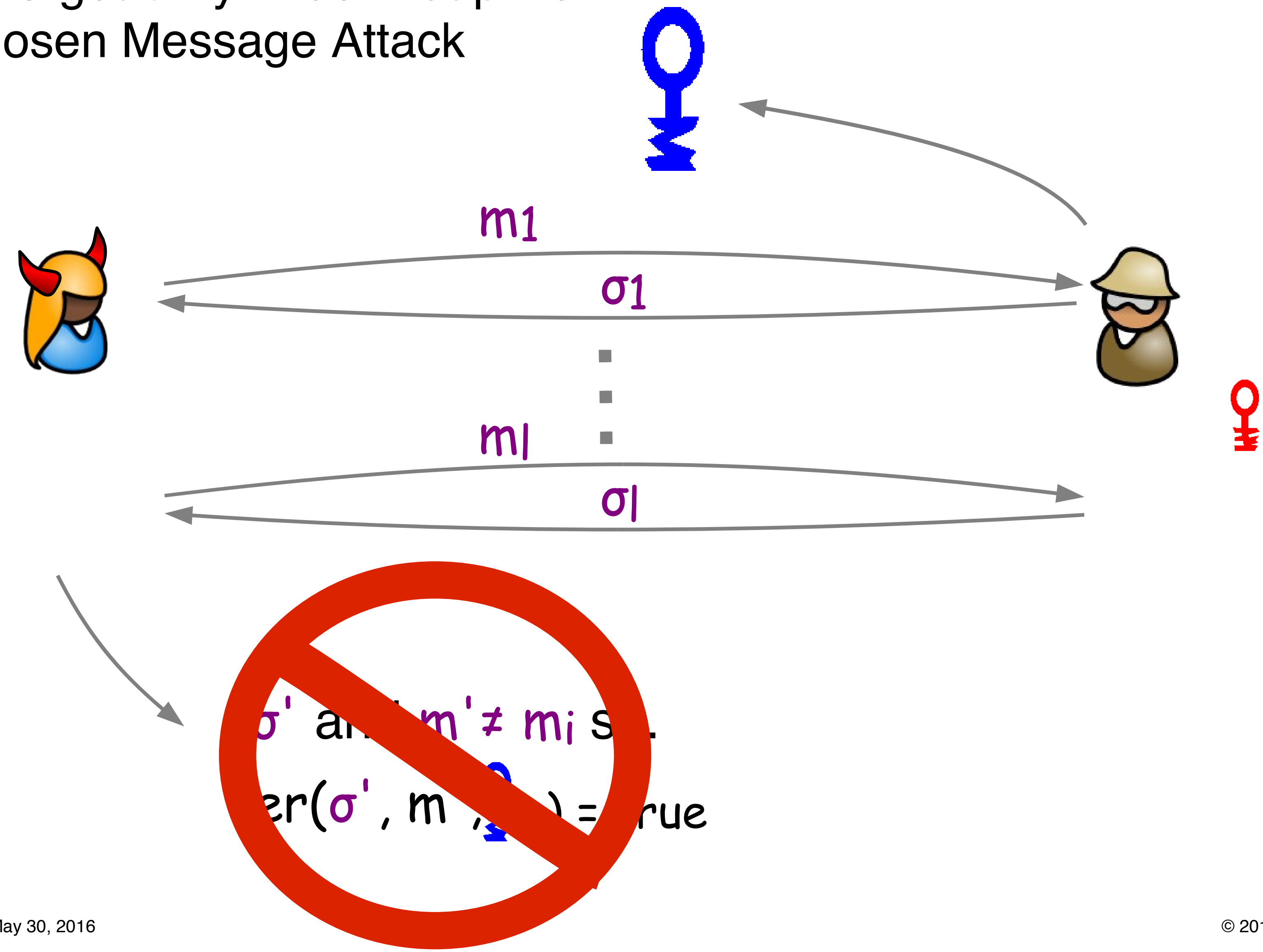
Unforgeability under Adaptive Chosen Message Attack



Unforgeability under Adaptive Chosen Message Attack

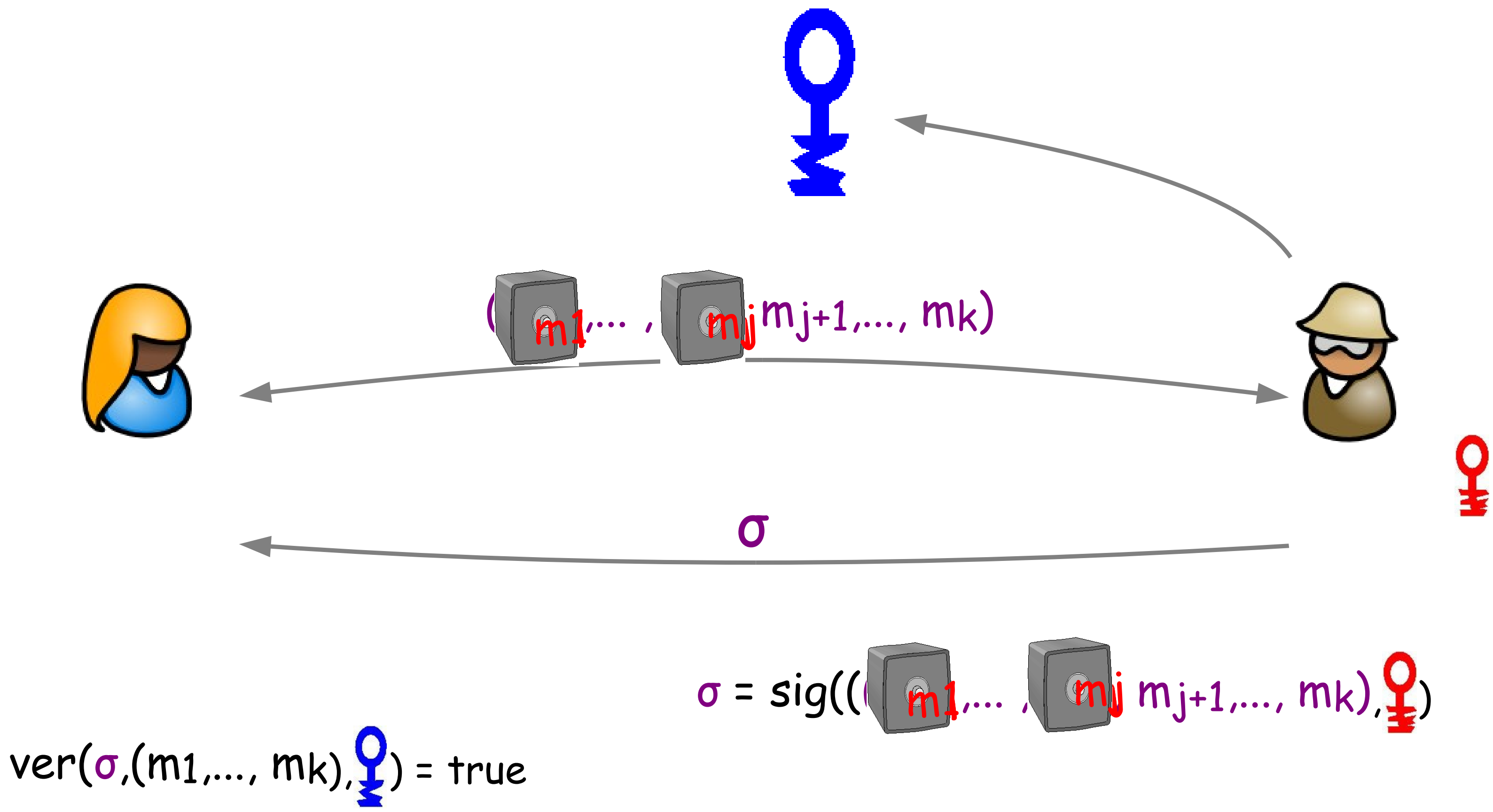


Unforgeability under Adaptive Chosen Message Attack



signature schemes with protocols

Signature Scheme: Signing Hidden Messages

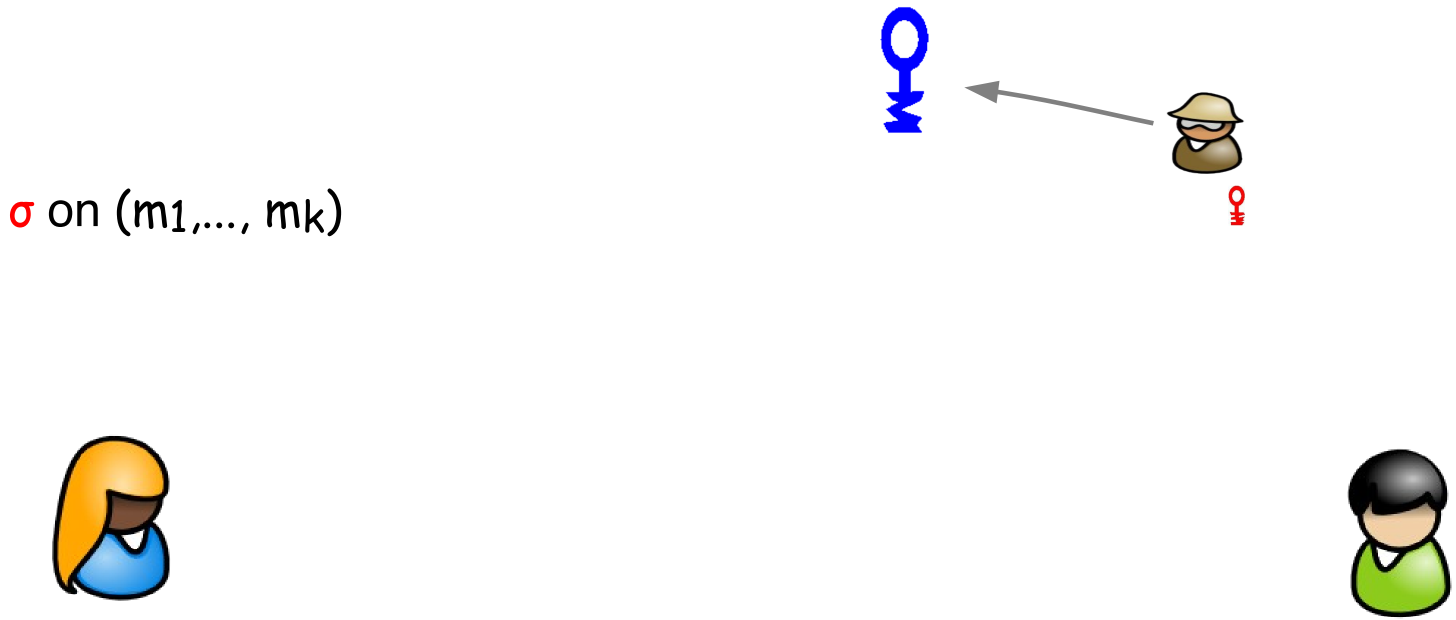


Verification remains unchanged!

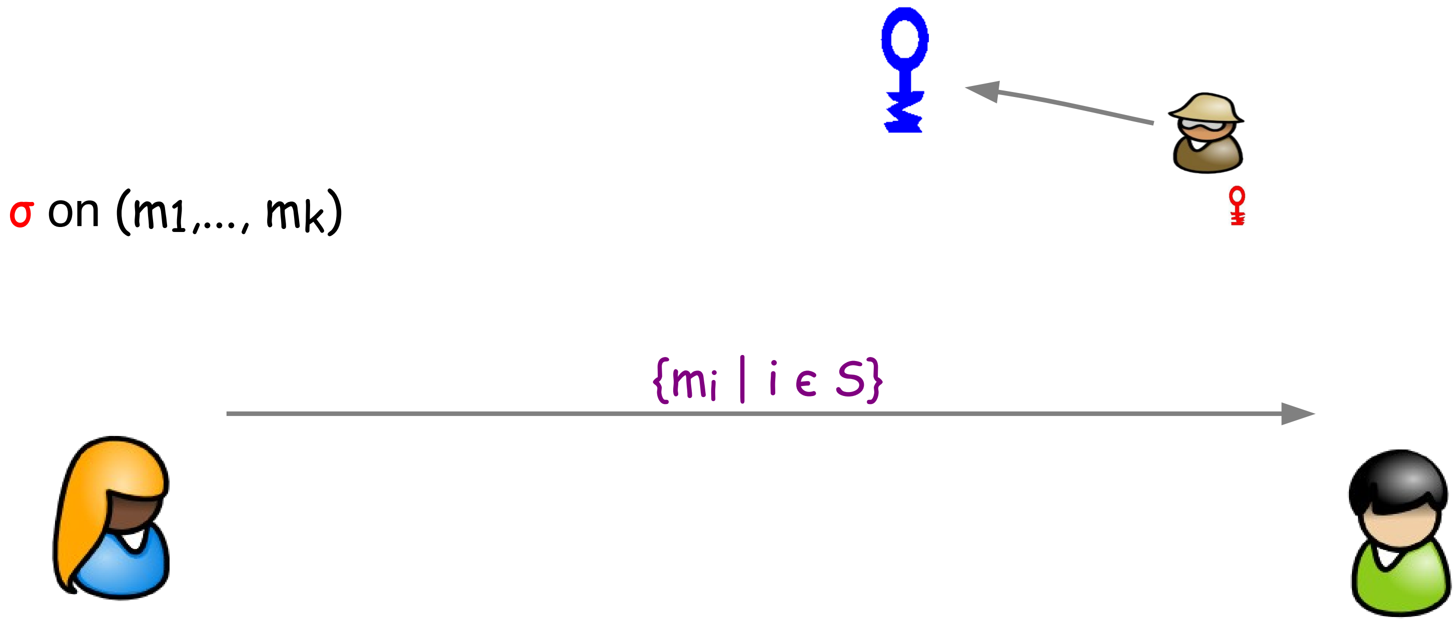
Security requirements basically the same, but

- Signer should not learn any information about m1, ..., mj
- Forgery w.r.t. message clear parts and opening of commitments

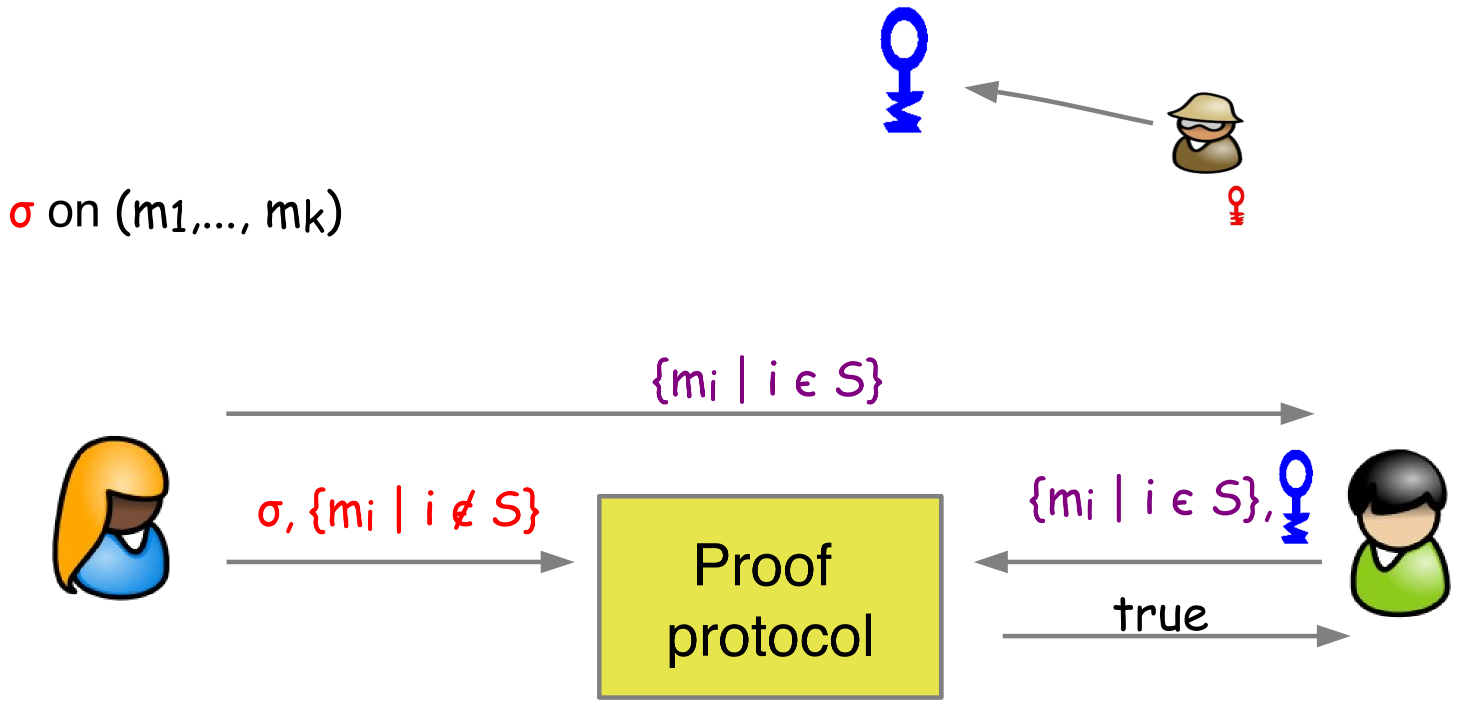
Proving Possession of a Signature



Proving Possession of a Signature



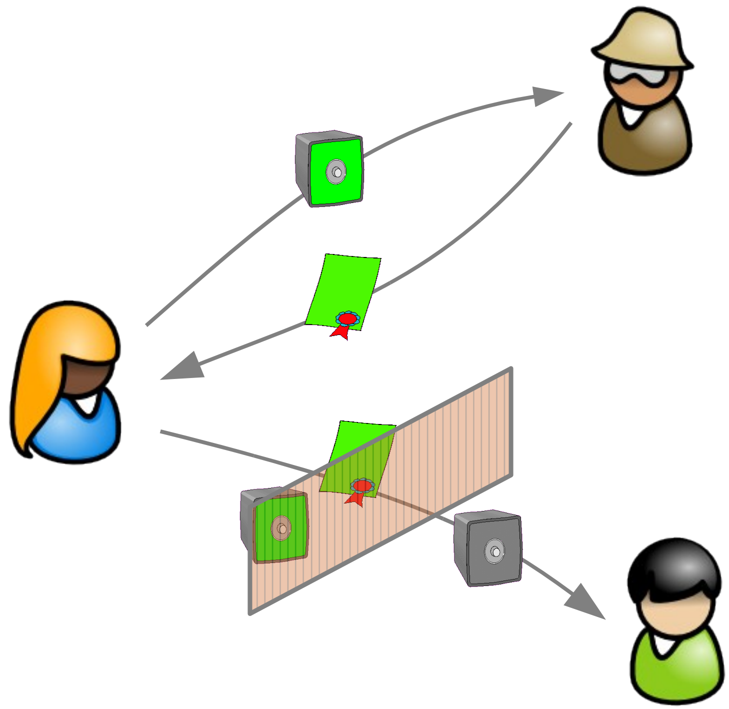
Proving Possession of a Signature



Variation:

- Send also m_i to verifier and
- Prove that committed messages are signed
- Prove properties about hidden/committed m_i

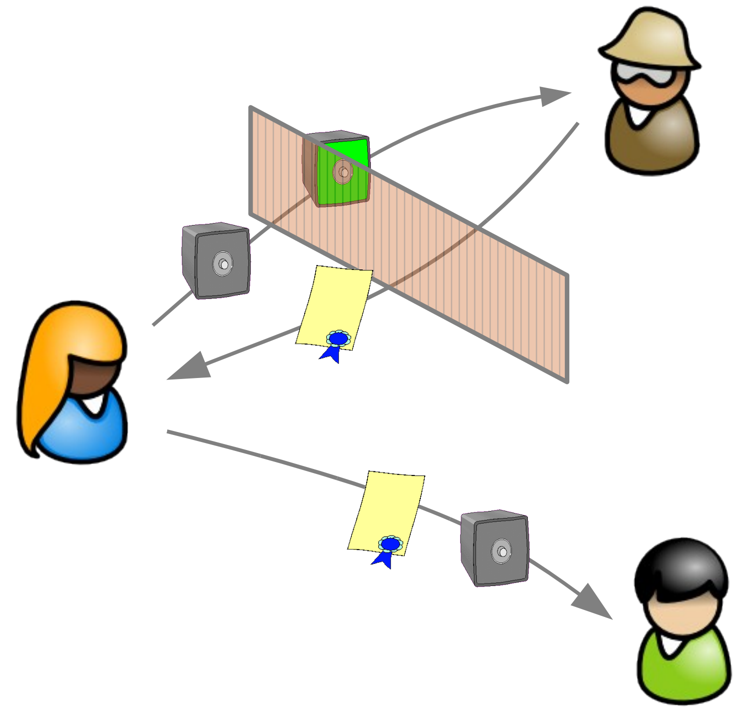
Blind Signatures vs Signatures with Protocols



can be used multiple times

Damgaard,Camenisch&Lysyanskaya

Strong RSA, DL-ECC,..



can be used only once

Chaum, Brands, et al.

Discrete Logs, RSA,..

Some signature schemes

RSA Signature Scheme – For Reference

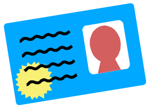
Rivest, Shamir, and Adlemann 1978

Secret Key: two random primes p and q

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

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

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



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

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

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

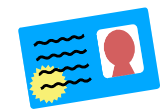
$$s^e = H(m) \pmod{n}$$

Correctness: $s^e = (H(m)^d)^e = H(m)^{d \cdot e} = H(m) \pmod{n}$

RSA Signature Scheme – for reference

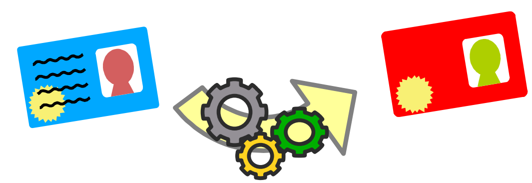
Verification signature on a message $m \in \{0,1\}^*$

$$s^e := H(m) \pmod{n}$$



Wanna do proof of knowledge of signature on a message, e.g.,

$$PK\{ (m,s):\ s^e = H(m) \pmod{n} \}$$



But this is not a valid proof expression!!!! :-(

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

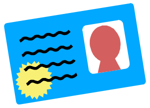
Secret key: factors of n

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

- choose random *prime* $2^{\ell+2} > e > 2^{\ell+1}$ and *integer* $s \approx n$
- compute c :

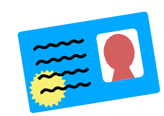
$$c = (d \,/\, (a_1^{m1} \cdot \ldots \cdot a_k^{mk}\; b^s\,))^{1/e} \bmod n$$

- signature is (c,e,s)



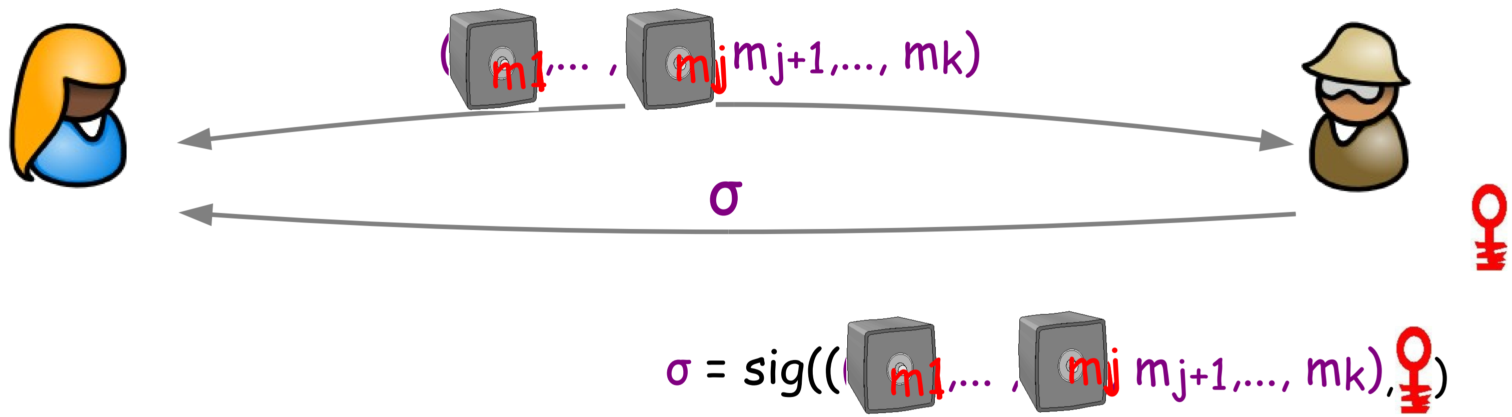
To verify a signature (c,e,s) on messages $m1, \ldots, mk$:

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



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

Sign blindly with CL signatures



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

$C + PK\{(m1, m2, s'): C = a_1^{m1} a_2^{m2} b^{s'}\}$

Choose e, s''

$c = (d/(C\, a_3^{m3}\, b^{s''}))^{1/e}$ mod n

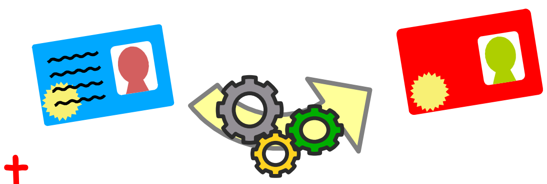
(c, e, s'')

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

Proving Knowledge of a CL-signature

Recall: $d = c^{e}\, a1^{m1} a2^{m2}\, b^{s} \bmod n$

Observe:



- Let $c' = c\, b^{t} \bmod n$ with randomly chosen t
- Then $d = c'^{e}\, a1^{m1} a2^{m2}\, b^{s-et} \pmod{n}$, i.e.,
  $(c', e, s^* = s-et)$ is also signature on $m1$ and $m2$

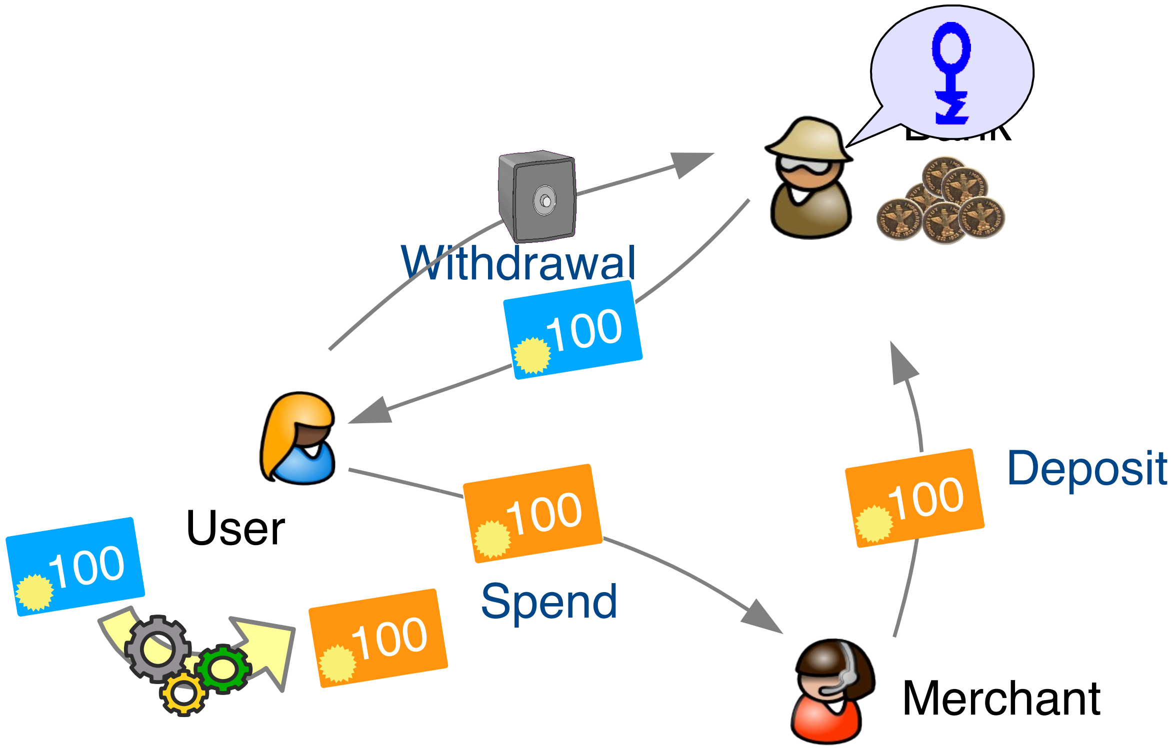
To prove knowledge of signature (c', e, s^*) on $m2$ and some $m1$

- provide c'
- $PK\{(\varepsilon, \mu 1, \sigma) : d/a2^{m2} := c'^{\varepsilon}\, a1^{\mu 1}\, b^{\sigma} \wedge \mu \in \{0,1\}^{\ell} \wedge \varepsilon > 2^{\ell+1}\}$

$\rightarrow$ proves $d := c'^{\varepsilon}\, a1^{\mu 1}\, a2^{m2} b^{\sigma}$

Realizing On-Line eCash

Recall basic idea



- Issue coin: Hide serial number from bank when issuing
  - sign commitment of random serial number
- Spend coin: reveal serial number and proof
  - knowledge of signature on
  - commitment to serial number

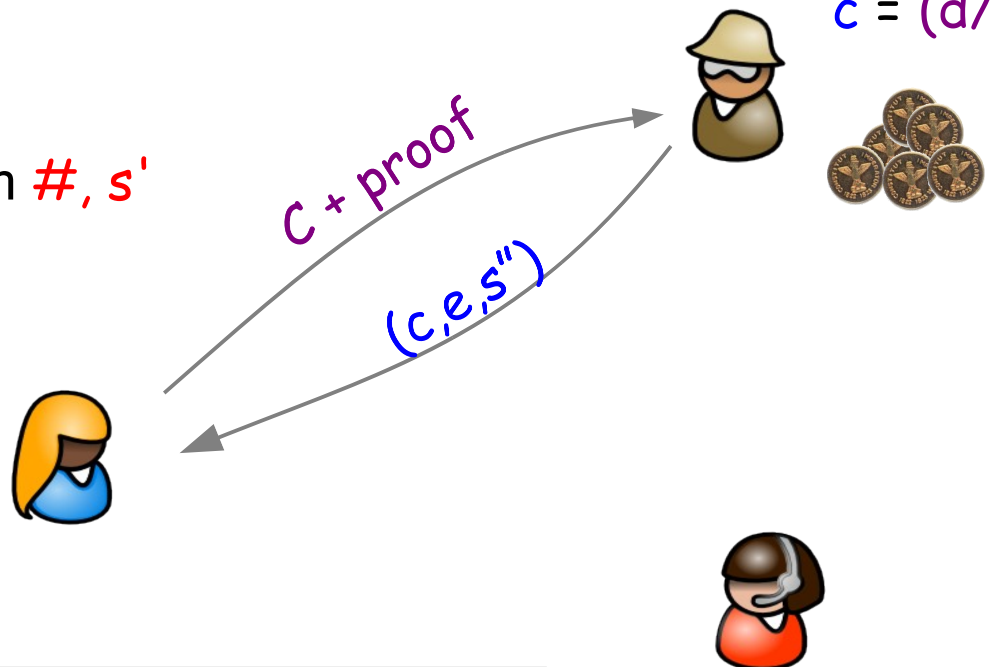
Choose e, s''

$c = (d/(C\, b^{s''}))^{1/e} \mod n$

choose random $\#, s'$

and compute

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

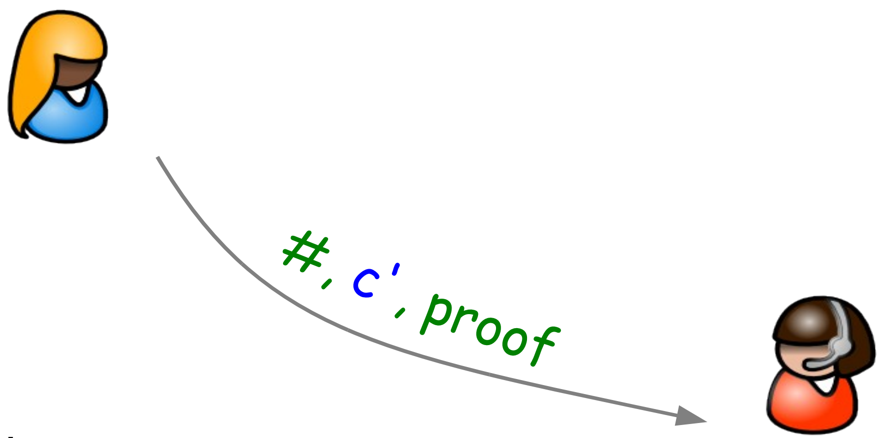


$(c, e, s''+s')$ s.t.

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

$(c, e, s'' + s')$ s.t.

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



compute

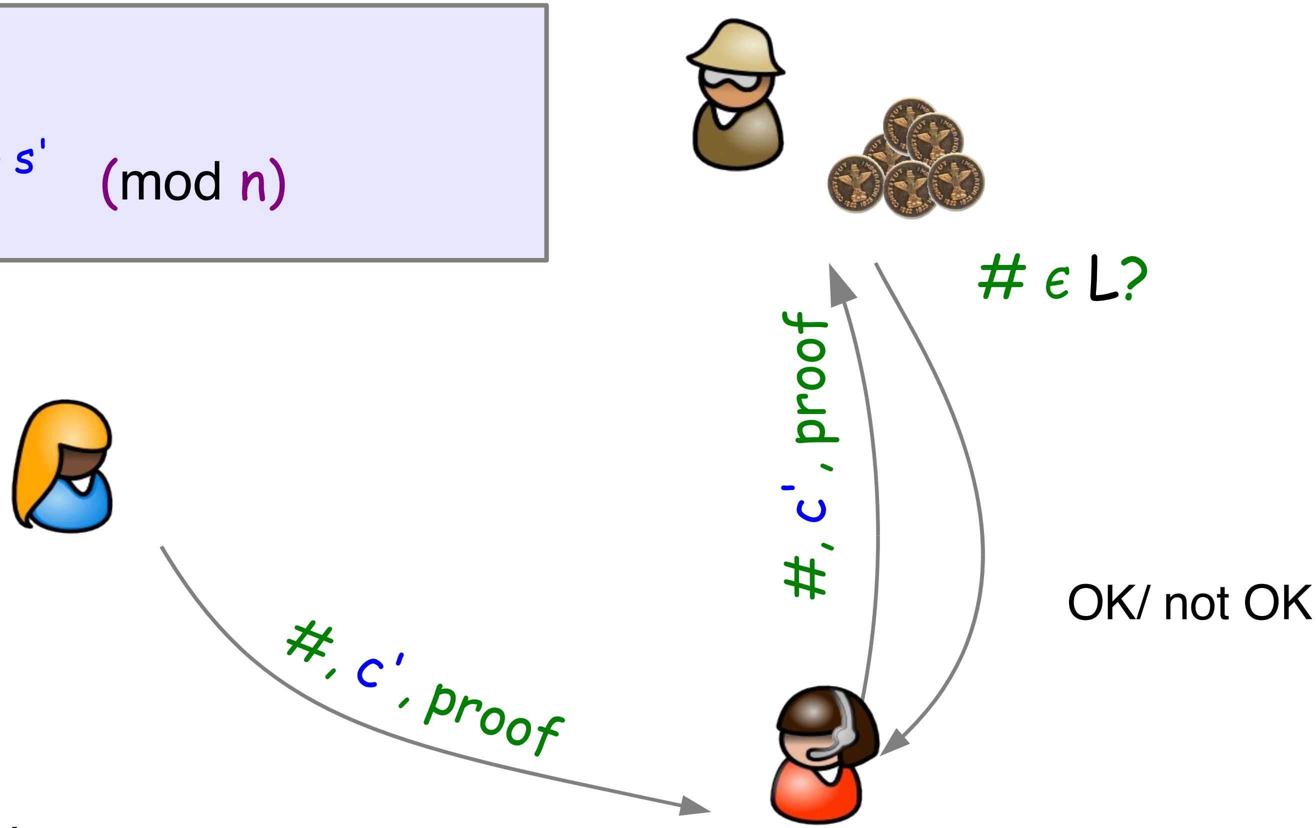
$c' = c\, b^{s'} \bmod n$

$\text{proof} = PK\{(\varepsilon, \mu, \rho, \sigma) : \quad d / a_1^{\#} = c'^{\varepsilon} b^{\sigma} \pmod{n} \}$

On-line E-cash: Payment

$(c, e, s'' + s')$ s.t.

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

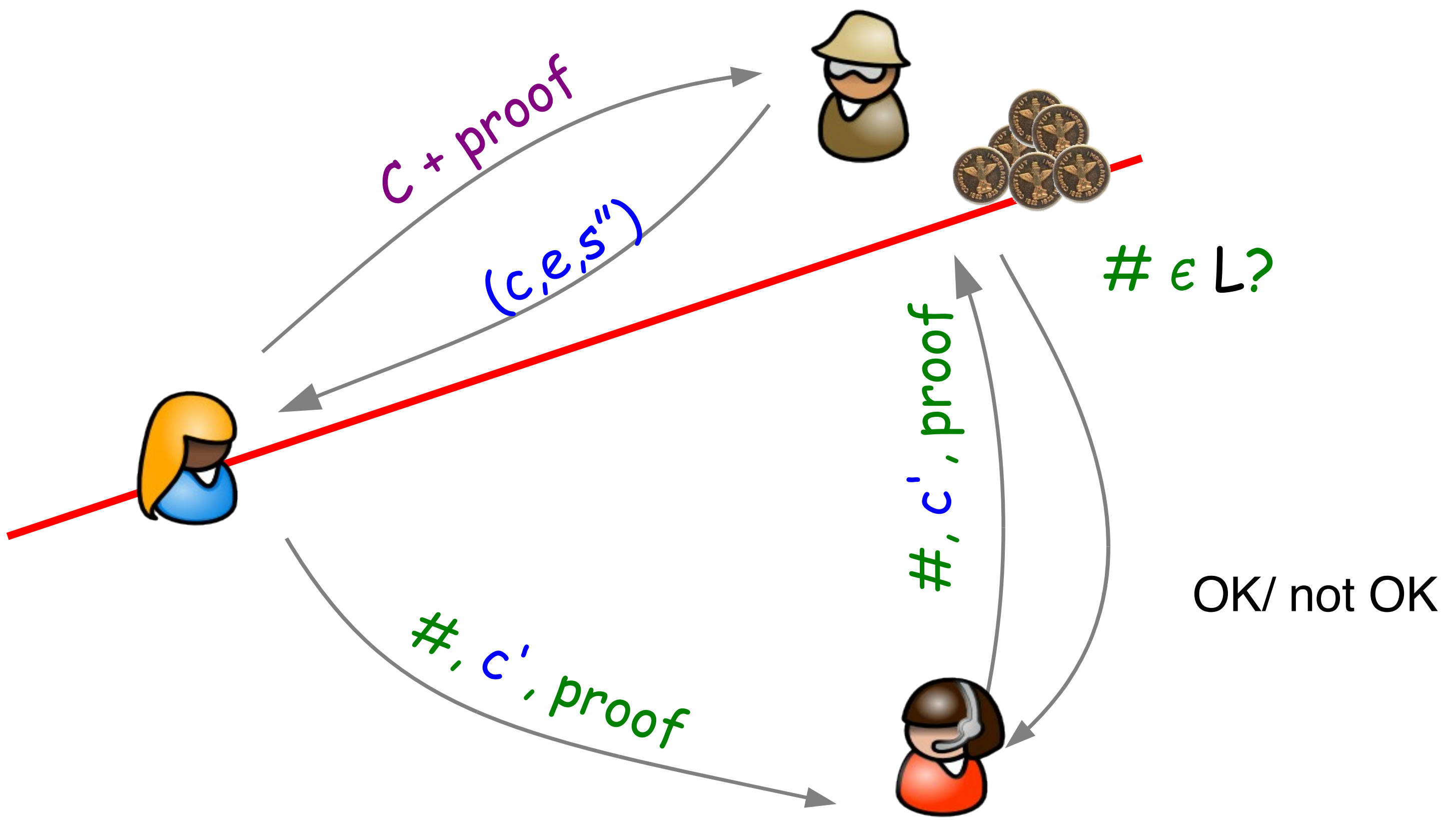


compute

$$c' = c\, b^{s'} \bmod n$$

$$\text{proof} = PK\{(\varepsilon, \mu, \rho, \sigma) : \quad d / a_1^{\#} = c'^{\varepsilon} b^{\sigma} \pmod{n} \}$$

- Anonymity
  - Bank does not learn # during withdrawal
  - Bank & Shop do not learn c, e when spending

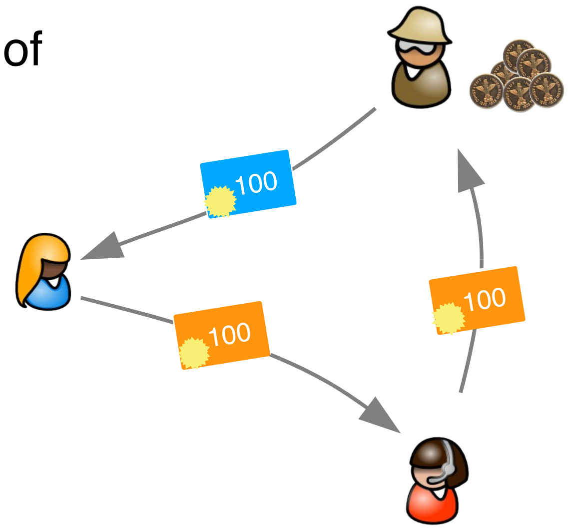


Double Spending:

- Spending = Compute
  - $c' = c\, b^{s'} \bmod n$
  - proof = $PK\{(\varepsilon, \mu, \rho, \sigma) : \quad d / a_1^{\#} = c'^{\varepsilon}\, b^{\sigma} \pmod{n}\}$
- Can use the same # only once....
  - If more #'s are presented than withdrawals:
    - Proofs would not sound
    - Signature scheme would not secure

Realizing Off-Line eCash

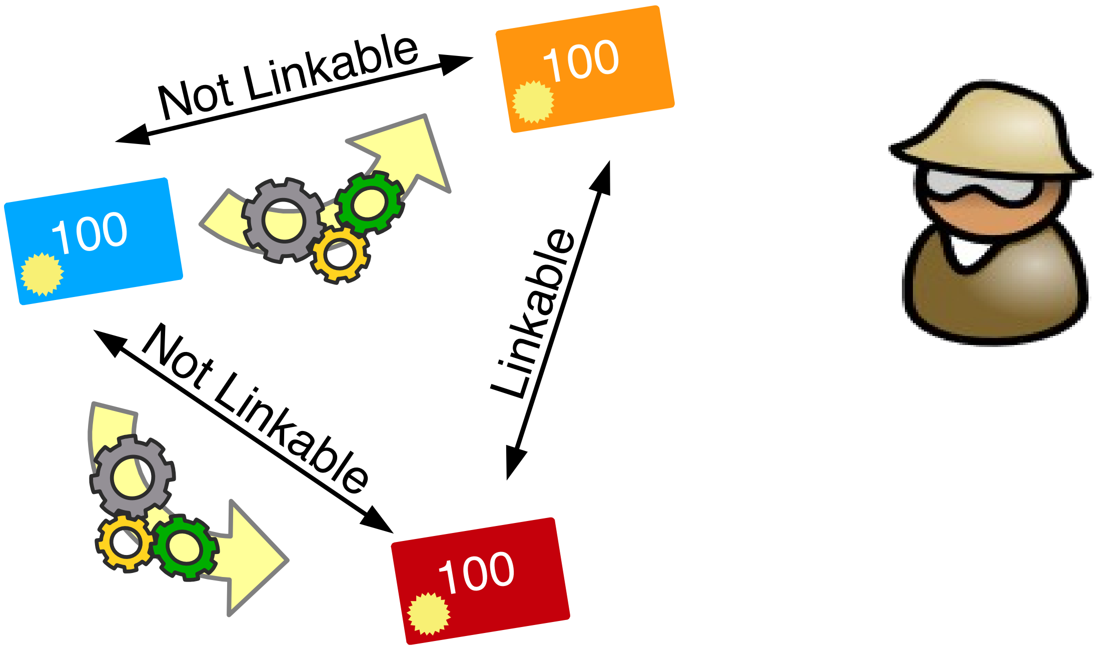
On-Line Solution:
1. Coin = random serial # (chosen by user) signed by Bank
2. Banks signs blindly
3. Spending by sending # and prove knowledge of signature to Merchant
4. Merchant checks validy w/ Bank
5. Bank accepts each serial # only once.



Off-Line:
- Can check serial # only after the fact
- … but at that point user will have been disappeared...

Goal:

- spending coin once: OK
- spending coin twice: anonymity revoked



Seems like a paradox, but crypto is all about solving paradoxical problems :-)

Main Idea:

- Include #, id, r
- Upon spending:

  reveal #, and $t = id + r\,u$,

  with c randomly chosen by merchant
- t won't reveal anything about id!
- However, given two equations (for the same #, id, r)

  $t1 = id + r\,u1$

  $t2 = id + r\,u2$

  one can solve for id.

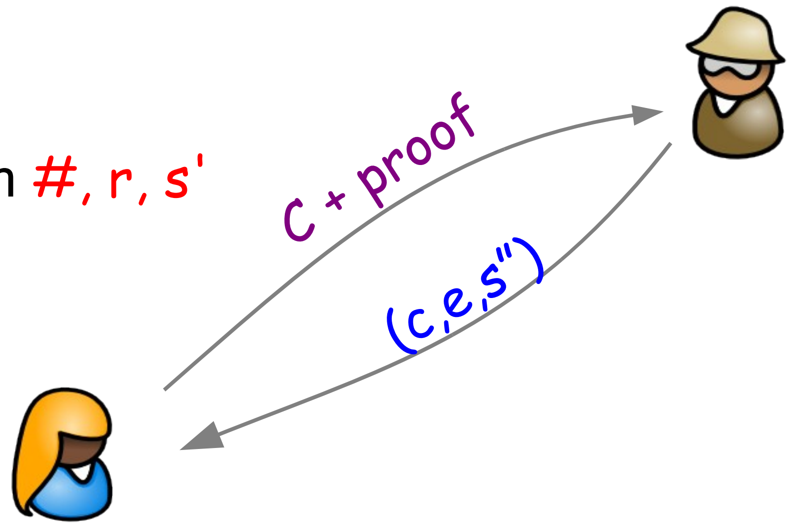
Off-line E-cash: Withdrawal

$d = c^{e}\, C\, a_3^{nym}\, b^{s''} \mod n$

choose random #, r, s'

and compute

$C = a_1^{\#} a_2^{r} b^{s'}$



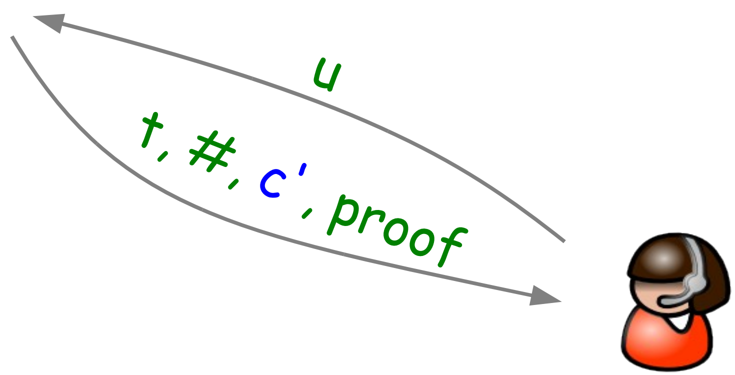
$(c,e,s''+s')$ s.t.

$d = c^{e}\, a_1^{\#} a_2^{r} a_3^{nym}\, b^{s''+s'} \pmod{n}$

Let $G=\langle g\rangle$ be a group of order q

$(c,e,s''+s')$ s.t.

$$d = c^e a_1^{\#} a_2^{r} a_3^{nym} b^{s''+s'} \pmod{n}$$



choose random u

compute

$t = r + u\ nym \bmod q$

$c' = c\, b^{s'} \bmod n$

proof = $PK\{(\varepsilon, \mu, \rho, \sigma) :$

$$d / a_1^{\#} = c'^{\varepsilon} a_2^{\rho} a_3^{\mu} b^{\sigma} \pmod{n} \ \wedge\ g^{t} = g^{\rho} (g^{u})^{\mu} \}$$

$$PK\{(\varepsilon, \mu, \rho, \sigma) : d / a_1^{\#} = c'^{\varepsilon} a_2^{\rho} a_3^{\mu} b^{\sigma} \pmod{n} \wedge g^{t} = g^{\rho} (g^{u})^{\mu} \}$$

1. $d = c'^{\varepsilon} a_1^{\#} a_2^{\rho} a_3^{\mu} b^{\sigma} \pmod{n}$

   => $(c', \varepsilon, \sigma)$ is a signature on $(\#, \mu, \rho)$

2. $g^{t} = g^{\rho + u\mu}$

   => $t = \rho + u\,\mu \mod q$,
   i.e., t was computed correctly!

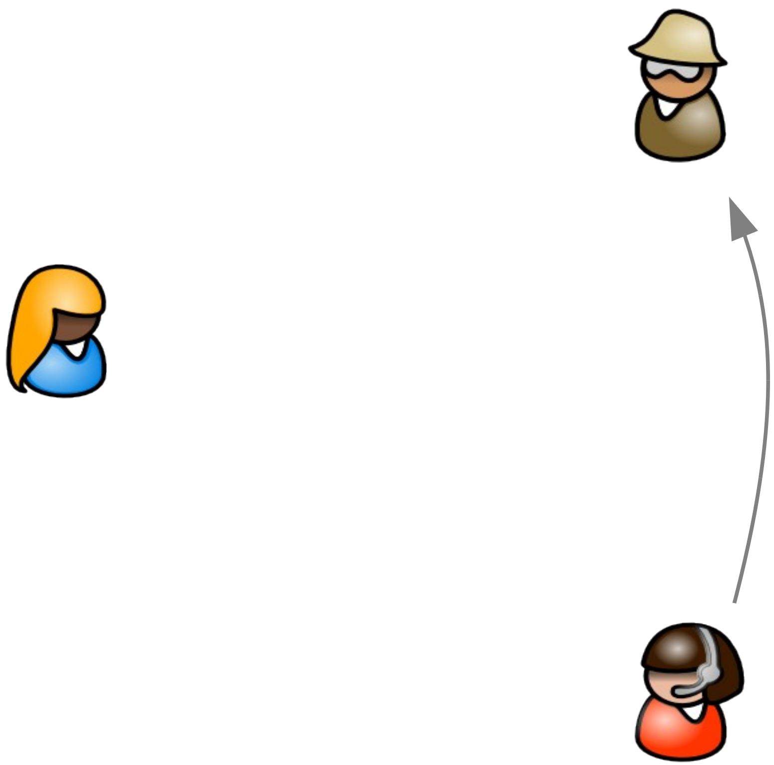
$\# \in L$?

If so:

1. $t = \rho + u\,\mu \pmod{q}$
2. $t' = \rho + u'\,\mu \pmod{q}$

solve for ρ and μ.

=> μ = nym because of proof



u, t, #, proof

Off-line E-cash: Security

- Unforgeable:
  - no more coins than #'s,
    - otherwise one can forge signatures
    - or proofs are not sound
  - if coins with same # appears with different u's => reveals nym

- Anonymity:
  - # and r are hidden from signer upon withdrawal
  - t does not reveal anything about nym (is blinded by r)
  - proof proof does not reveal anything

Extensions and more

e-Cash

- K-spendable cash
  - Multiple serial numbers and randomizers per coin
  - Use PRF to generate serial number and randomizers from seed in coin
- Money laundering preventions
  - Must not spend more that $10000 dollars with same party
  - Essentially use additional coin defined per merchant that controls this

Other protocols from these building blocks, essentially anything with authentication and privacy

- Anonymous credentials, eVoting, ...

Alternative building blocks

- A number of signatures scheme that fit the same bill
- (Verifiable) encryption schemes that work along as well
- Alternative framework: Groth-Sahai proofs plus “structure-preserving” schemes

Advertisement :-)

PhD and Postdocs available at IBM Research – Zurich

Please contact me

Thank you!

- eMail: identity@zurich.ibm.com
- Links:
  - www.abc4trust.eu
  - www.futureID.eu
  - www.au2eu.eu
  - www.PrimeLife.eu
  - www.zurich.ibm.com/idemix
  - idemixdemo.zurich.ibm.com
- Code
  - github.com/p2abcengine & abc4trust.eu/idemix

References

- D. Chaum, J.-H. Evertse, and J. van de Graaf. *An improved protocol for demonstrating possession of discrete logarithms and some generalizations.* In EUROCRYPT ’87, vol. 304 of LNCS, pp. 127–141. Springer-Verlag, 1988.
- S. Brands. *Rapid demonstration of linear relations connected by boolean operators.*In EUROCRYPT ’97, vol. 1233 of LNCS, pp. 318–333. Springer Verlag, 1997.
- Mihir Bellare: Computational Number Theory http://www-cse.ucsd.edu/~mihir/cse207/w-cnt.pdf
- Camenisch, Lysanskaya: *Dynamic Accumulators and Applications to Efficient Revocation of Anonymous Credentials*. Crypto 2002, Lecture Notes in Computer Science, Springer Verlag.
- Ateniese, Song, Tsudik: Quasi-Efficient Revocation of Group Signatures. In Financial Cryptography 2002, Lecture Notes in Computer Science, Springer Verlag.
- Jan Camenisch, Natalie Casati, Thomas Gross, Victor Shoup: Credential Authenticated Identification and Key Exchange. CRYPTO 2010:255-276
- Jan Camenisch, Maria Dubovitskaya, Gregory Neven: Oblivious transfer with access control. ACM Conference on Computer and Communications Security 2009: 131-140

- Ateniese, Song, Tsudik: Quasi-Efficient Revocation of Group Signatures. In Financial Cryptography 2002, Lecture Notes in Computer Science, Springer Verlag.

- M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko: *The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme.* Journal of Cryptology, Volume 16, Number 3. Pages 185 -215, Springer-Verlag, 2003.
- E. Bangerter, J. Camenisch and A. Lyskanskaya: A Cryptographic Framework for the Controlled Release Of Certified Data. In Twelfth International Workshop on Security Protocols 2004. www.zurich.ibm.com/~jca/publications
- Stefan Brands: Untraceable Off-line Cash in Wallets With Observers: In Advances in Cryptology – CRYPTO '93. Springer Verlag, 1993.

References

- J. Camenisch and A. Lyskanskaya: Efficient Non-transferable Anonymous Multi-show Credential System with Optional Anonymity Revocation. www.zurich.ibm.com/~jca/publications
- David Chaum: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. In Communications of the ACM, Vol. 24 No. 2, pp. 84—88, 1981.
- David Chaum: Blind Signatures for Untraceable Payments. In Advances in Cryptology – Proceedings of CRYPTO '82, 1983.
- David Chaum: Security Without Identification: Transaction Systems to Make Big Brother obsolete: in Communications of the ACM, Vol. 28 No. 10, 1985.
- Camenisch, Shoup: Practical Verifiable Encryption and Decryption of Discrete Logarithms. CRYPTO 2003: 126-144
- Victor Shoup: A computational introduction to Number Theory and Algebra. Available from: http://www.shoup.net/ntb/
- D. Chaum: *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms.* In Communications of the ACM.
- D. Chaum: *The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability*. Journal of Cryptology, 1988.
- J. Camenisch and V. Shoup: *Practical Verifiable Encryption and Decryption of Discrete Logarithms.* In Advances in Cryptology - CRYPTO 2003.
- T. ElGamal: *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms.* In Advances in Cryptology - CRYPTO '84.