

# E-Privacy – Privacy in the Electronic Society

---

## Anonymous Credentials III

Jan Camenisch

IBM Zurich Research Laboratory

Rüschlikon

[jca@zurich.ibm.com](mailto:jca@zurich.ibm.com)

# Outline

---

- Recap Tools
- Example Apps: E-Cash
- Example Apps: Revocation of Certs
- Example Apps: Direct Anonymous Attestation

# Recap Technologies

---

Proof Protocols, Commitments, and Signatures.

# Zero Knowledge Proofs

---

Non-interactive (Fiat-Shamir heuristic):

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

Logical combinations:

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

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

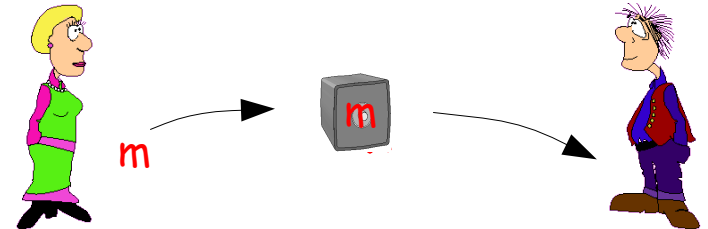
Intervals and groups of different order (under SRSA):

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

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

# Commitment Schemes

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



To commit to element  $x \in \mathbb{Z}_q$ :

- choose  $r \in \mathbb{Z}_q$  and compute  $c = g^x h^r$

To commit to integer  $x \in \mathbb{Z}$ :

- similarly, if order of  $G$  is not known, e.g.,  $G = \text{QR}_n$  for an RSA modulus  $n$

# Commitment Schemes (Group Elements)

- Given group  $G = \langle g \rangle = \langle h \rangle$  of order  $q$  with  $\log_g h$  unknown

To commit to *group* element  $u \in G$ :

- perfectly hiding, computationally binding (under discrete logarithm assumption)

choose  $r_1, r_2 \in \mathbb{Z}_q$  and compute  $c = (ug^{r_1}, g^{r_1}h^{r_2})$

- computationally hiding (under decisional Diffie-Hellman assumption), perfectly binding:

choose  $r \in \mathbb{Z}_q$  and compute  $c = (uh^r, g^r)$

To open first type of commitment:

- reveal  $u$  and  $r_1, r_2$  to verifier
- verifier checks if  $c = (ug^{r_1}, g^{r_1}h^{r_2})$

# Signature Scheme based on SRSA

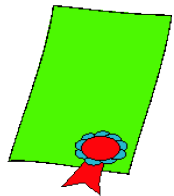
Public key of signer: RSA modulus  $n$  and  $a_i, b, d \in \mathbb{Q}\mathbb{R}_n$ , 

Secret key: factors of  $n$  

To sign  $k$  messages  $m_1, \dots, m_k \in \{0,1\}^\ell$  :

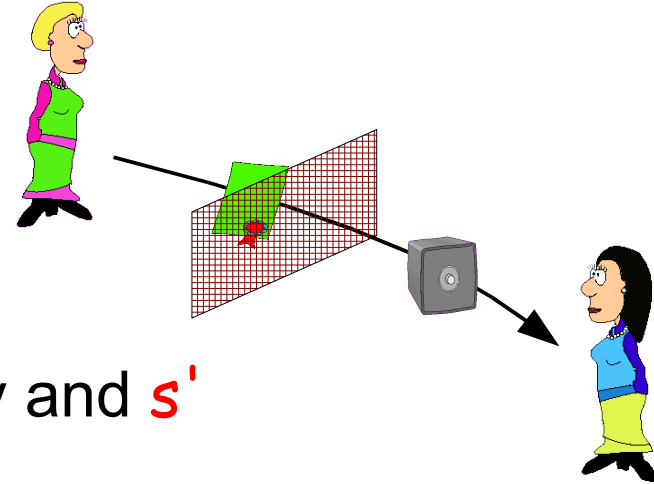
- choose random *prime*  $e > 2^\ell$  and *integer*  $s \approx n$
- compute  $c$  :

$$c = (d / (a_1^{m_1} \cdot \dots \cdot a_k^{m_k} b^s))^{1/e} \bmod n$$



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

# Proving Knowledge of a Signature



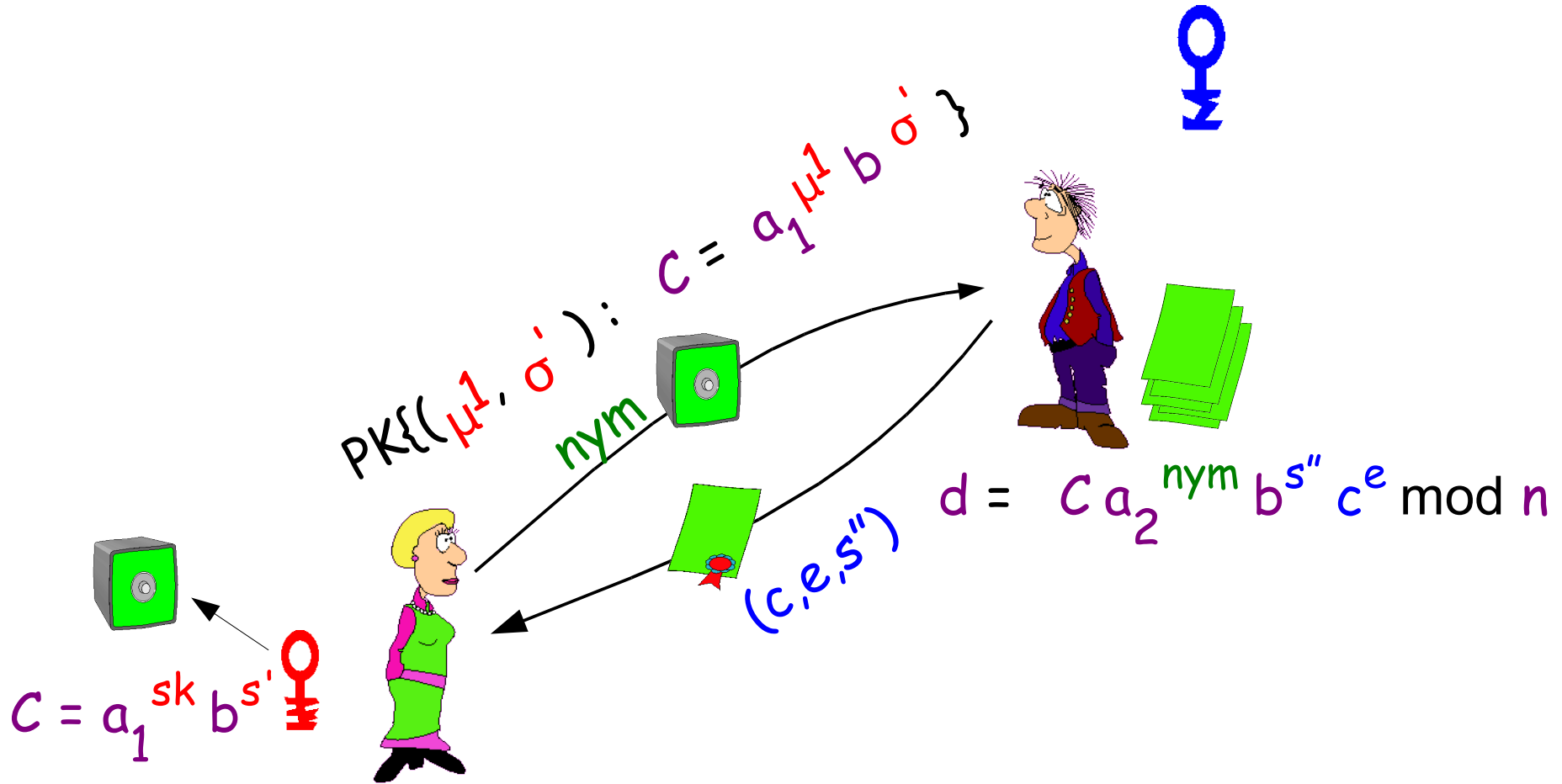
Observe:

- Let  $c' = c b^{s'}$  mod  $n$  with randomly and  $s'$
- then  $d = c'^e a^m b^{s+es'}$  (mod  $n$ ), i.e.,  
 $(c', e, s^* = s+es')$  is also signature on  $m$

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

- provide  $c'$
- $\text{PK}\{(\varepsilon, \mu, \sigma) : d := c'^\varepsilon a^\mu b^\sigma\}$

# Getting a Signature on a Secret Message



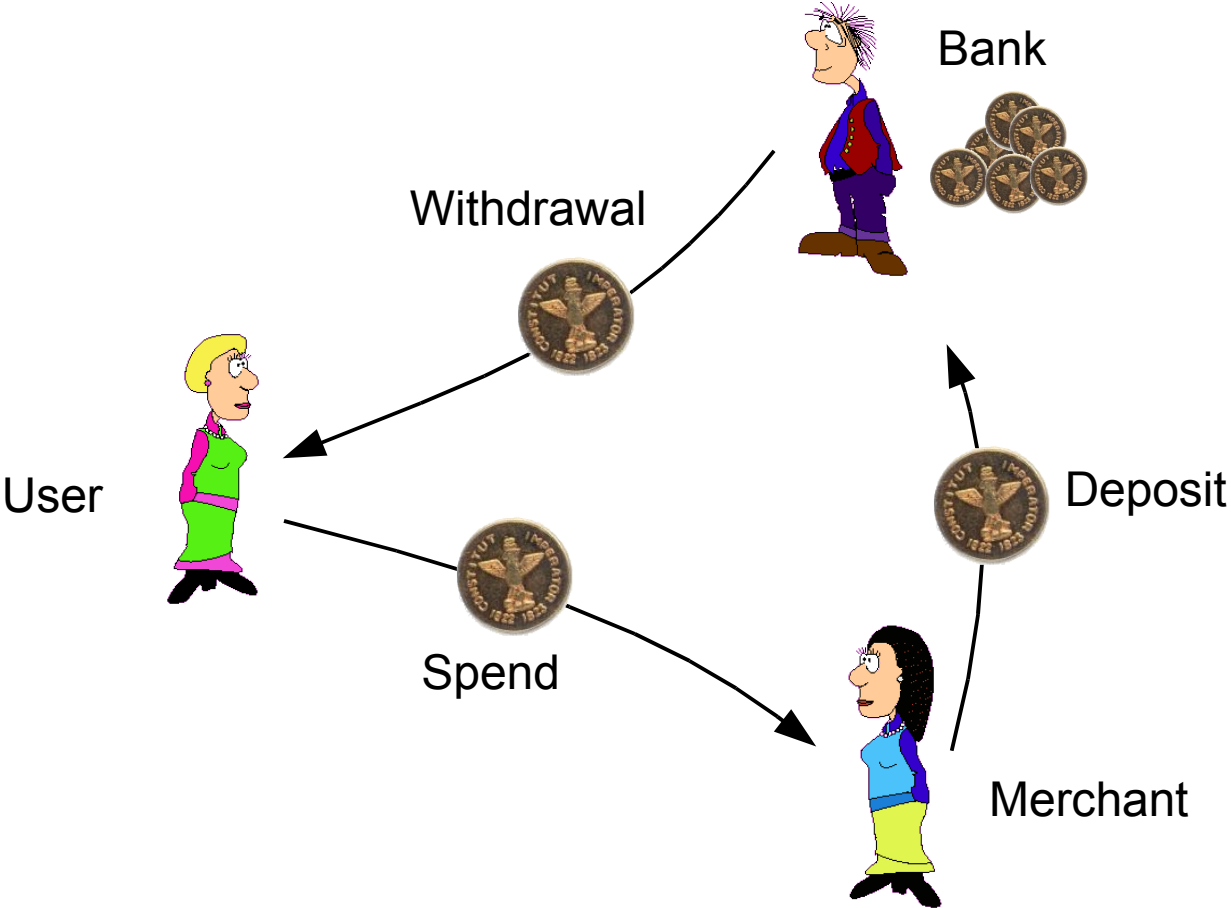
$$d = c^e a_1^{sk} a_2^{nym} b^{s'' + s'} \pmod n$$

# Application

---

E-cash: On-line

# E-Cash Setting



# E-Cash Setting

---

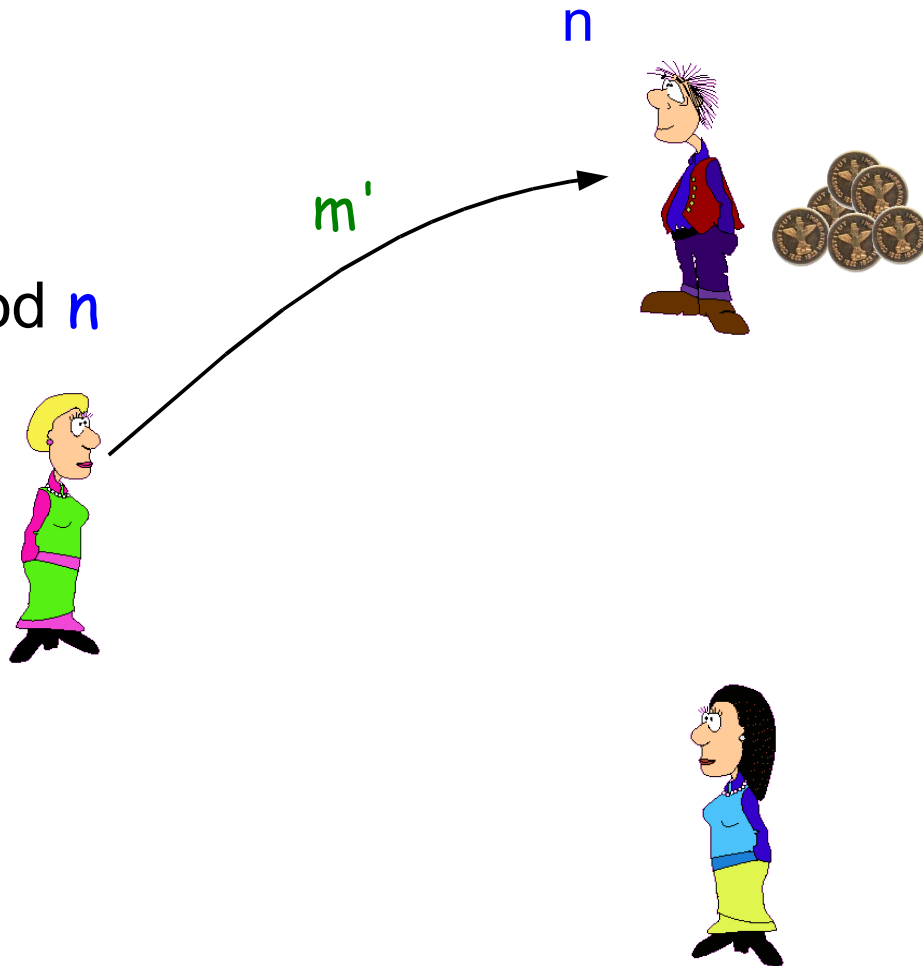
- Anonymity:  
Withdrawal and Deposit must be unlinkable
- Double Spending:  
Coin is bit-strings, can be spend twice
  - Bank *on-line*: merchant checks with bank whether some coin has previously been spend before accepting
  - Bank *off-line*: merchant accepts, get credited, but bank can reveal anonymity of double (multiple) spenders
    - Need to have scheme where spending once is anonymous
    - Spending more than once is not anonymous

# E-cash with (Blind) RSA Signatures

[Chaum 82]

random  $m$  and  $b$

$$m' = H(m) b^e \text{ mod } n$$



# E-cash with (Blind) RSA Signatures

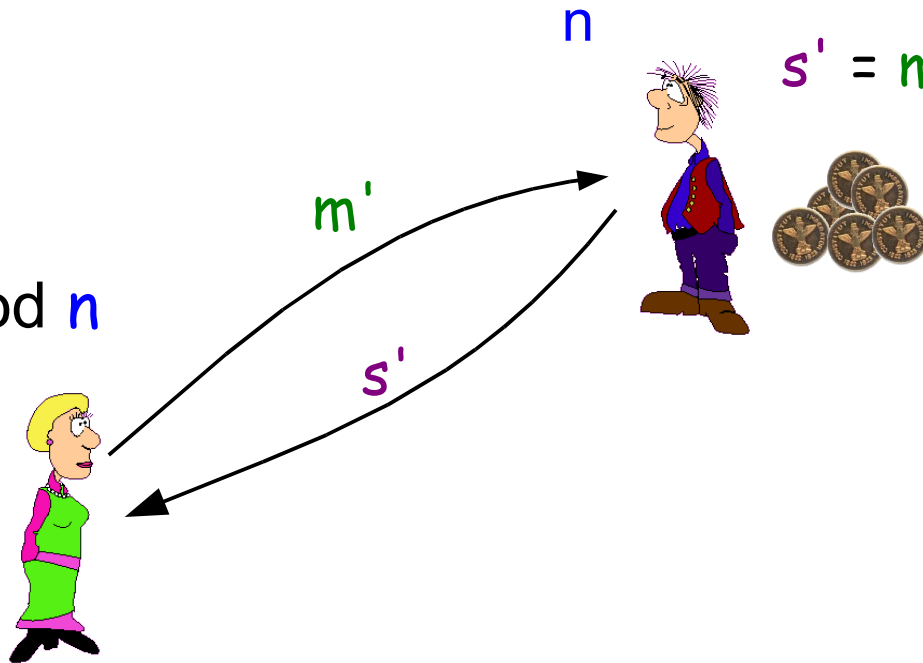
[Chaum 82]

$$d = 1/e \text{ mod } (p-1)(q-1)$$

$$s' = m' d \text{ mod } n$$

random  $m$  and  $b$

$$m' = H(m) b^e \text{ mod } n$$



$$s = s' / b \text{ mod } n$$

$$\Rightarrow s^e = H(m) \text{ mod } n$$

# E-cash with (Blind) RSA Signatures

[Chaum 82]

$$d = 1/e \text{ mod } (p-1)(q-1)$$

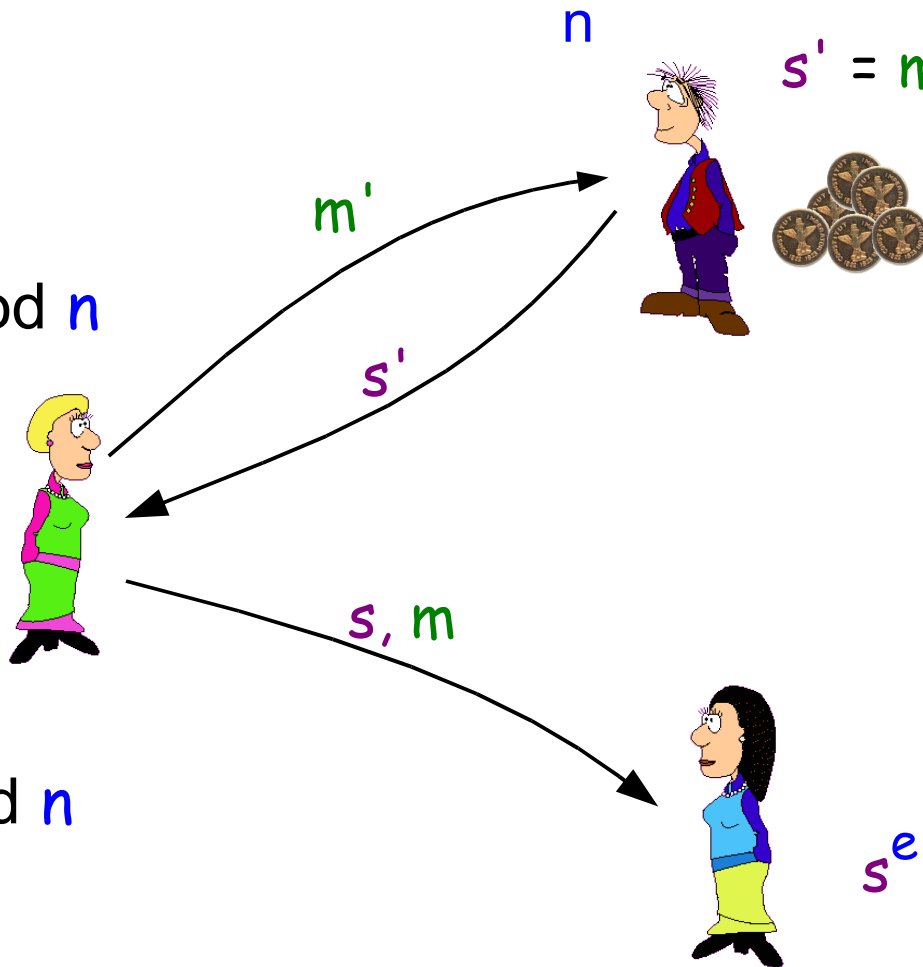
$$s' = m' d \text{ mod } n$$

random  $m$  and  $b$

$$m' = H(m) b^e \text{ mod } n$$

$$s = s' / b \text{ mod } n$$

$$\Rightarrow s^e = H(m) \text{ mod } n$$



# E-cash with (Blind) RSA Signatures

[Chaum 82]

$$d = 1/e \pmod{(p-1)(q-1)}$$

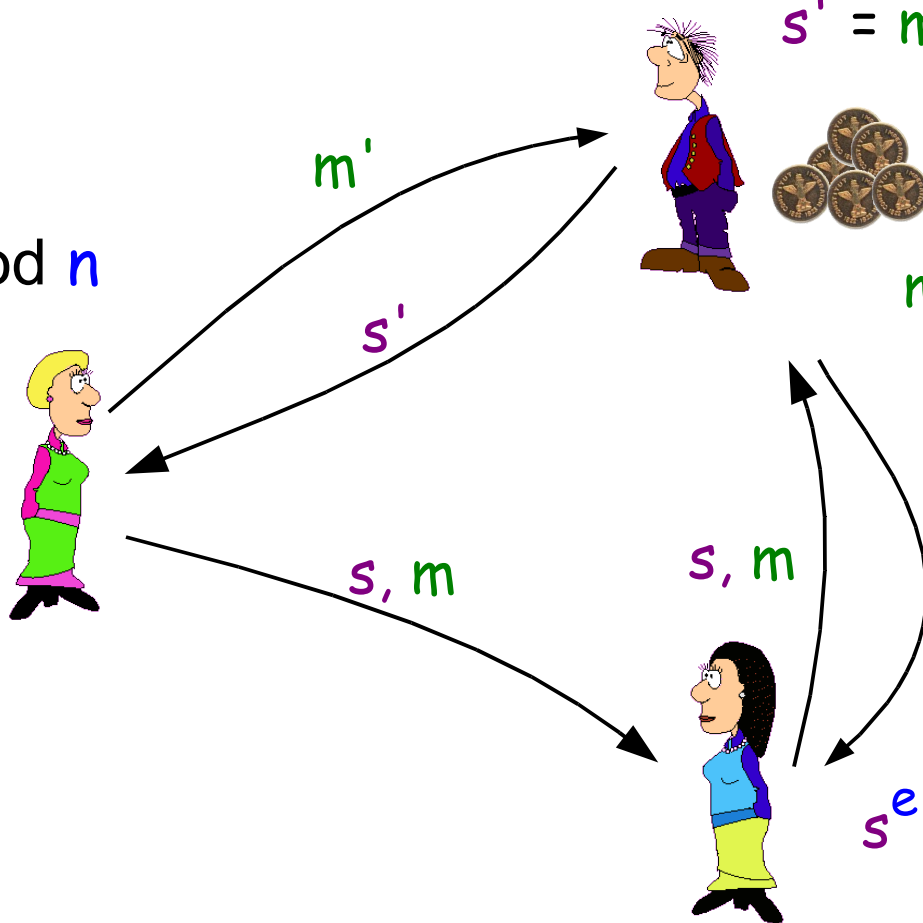
$$s' = m'^d \pmod{n}$$

random  $m$  and  $b$

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

$$s = s' / b \pmod{n}$$

$n$



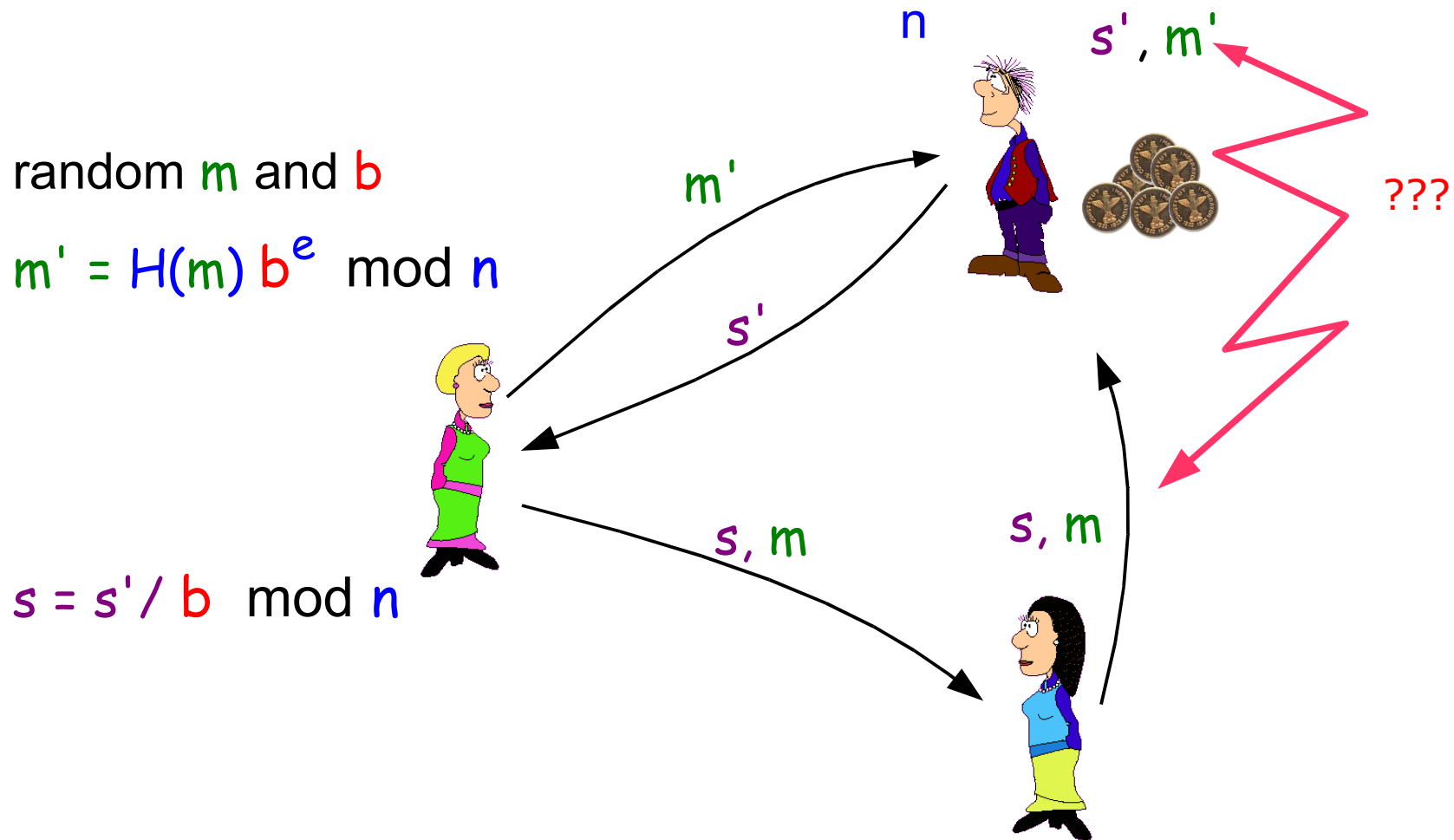
$m \in L?$

OK/ not OK

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

# Blindness of Blind RSA Signatures

[Chaum 82]



# Blindness of Blind RSA Signatures

[Chaum 82]

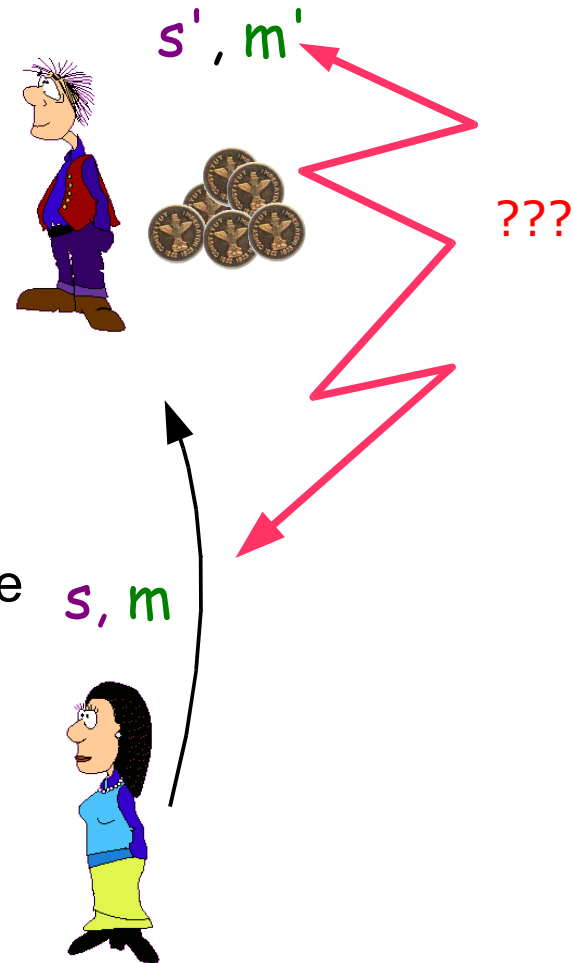
Given any two pairs  $(s, m)$  and  $(s', m')$ .

Let  $b = s'/s \pmod n$

Then:

$$\begin{aligned} m' &= s'^e = (s b)^e = s^e b^e = \\ &= H(m) b^e \pmod n \end{aligned}$$

Therefore the pairs could stem from the same user! Thus the two transactions are perfectly unlinkeable and signature scheme is perfectly blind.



# Security of Blind RSA Signatures

[Michels, Stadler, Sun 98]

Problem:

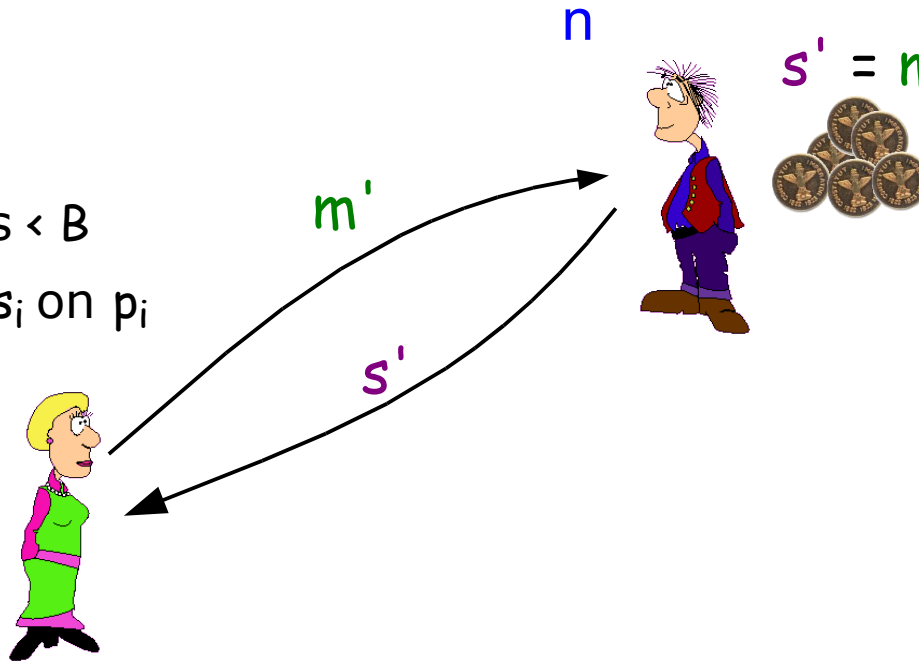
Signer computes  $e$ -th root  
of anything!

$$d = 1/e \pmod{(p-1)(q-1)}$$

$$s' = m'^d \pmod{n}$$

Attack:

Let  $p_i$  be primes  $< B$   
get signatures  $s_i$  on  $p_i$





find message  $m$  and set  $S$  s.t.  $H(m) = \prod_{i \in S} p_i$  (i.e., factor  $H(m)$ )

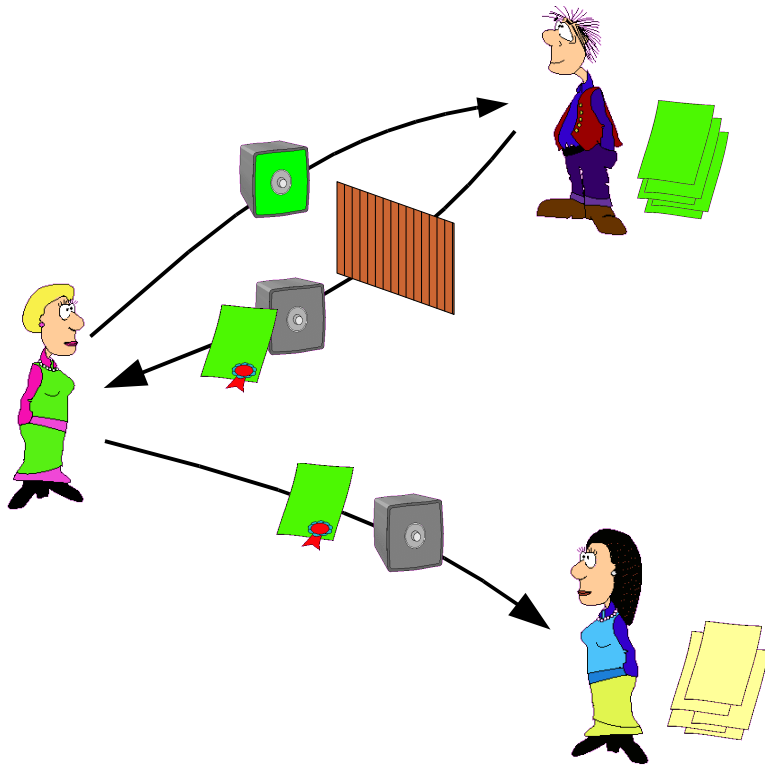
then for  $s = \prod_{i \in S} s_i$  we have

$$s^e = \prod_{i \in S} s_i^e = \prod_{i \in S} p_i = H(m) \pmod{n}.$$

# E-cash with (Blind) RSA Signatures

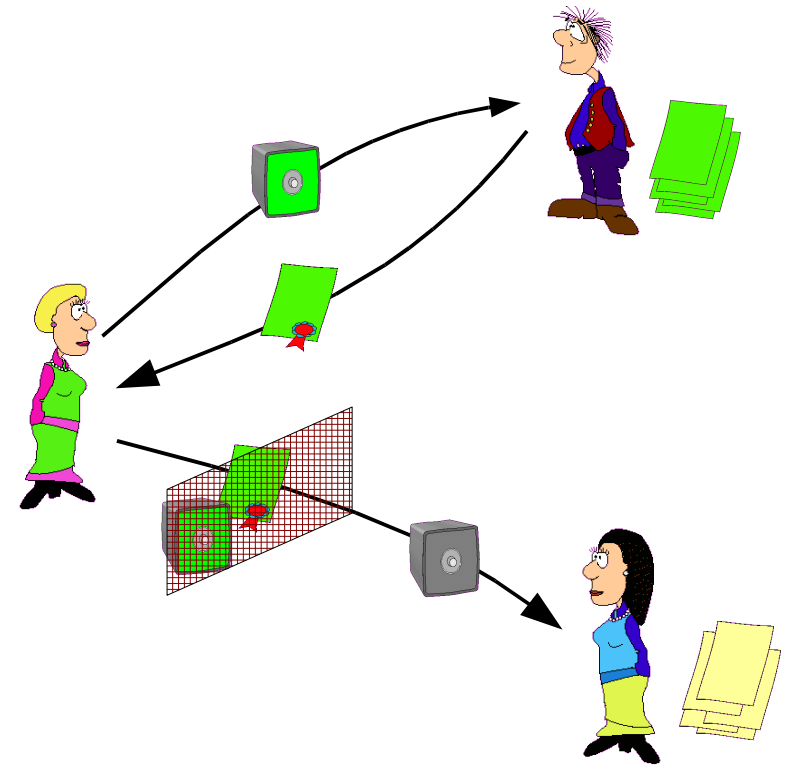
- Attack can be overcome by requiring  $H': \{0,1\}^* \rightarrow [0,n-1]$  instead of  $H: \{0,1\}^* \rightarrow \{0,1\}^\ell$   
Called full-domain hash, e.g., can be obtained padding  
$$H'(m) = H(1||m) || H(2||m) || H(3||m) || \dots || H(k||m)$$
- Security has been proved (in a somewhat weak model) [Bellare et al. 03]
- Can be used for credential scheme:
  - Credential: blind signature  $s$  on message  $m =$  
  - Showing credential:
    - reveal  $s$  and  $m$
    - prove knowledge of secret contained in 
  - But credential can be used only once! :-)

# Using Blind Signatures vs. Proving Possession



Certificates can be used only *once*!

- Chaum 82
- Brands 90'ies



Certificates can be used *multiple* times!

- Camenisch, Lysyanskaya 2000

# Proving Possession of Signature

In principle:  
Prove of Knowledge of  
message  $m$  and signature  $s$  s.t.

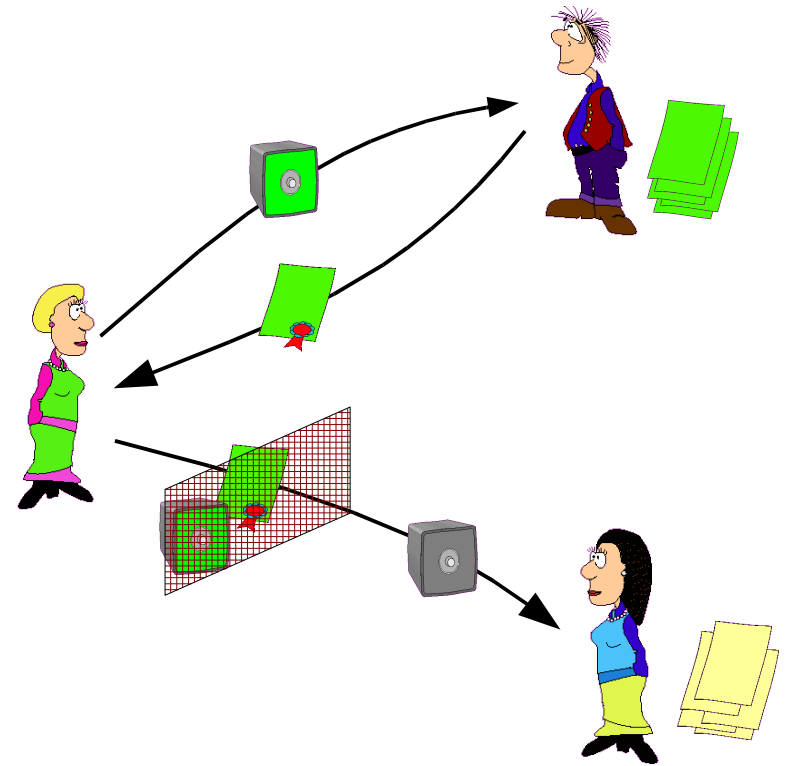
$$\text{ver}(s, m, PK_{\text{signer}}) = 1$$

In the case of RSA:

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

which, however, would not be efficient  
because of hash function, i.e., proof of  
knowledge of pre-image under hash-  
function -> Requires proof over  
representation of hash-function as gates.

=> need other signature scheme!



Certificates can be used *multiple* times!

- Camenisch, Lysyanskaya 2000

# Application

---

E-cash (Off-Line)



# Off-line E-cash

---

## Goal:

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

## Main Idea:

- Include  $\#, id, r$
- Upon spending:  
    reveal  $\#,$  and  $t = id + r c,$   
    with  $c$  randomly chosen by merchant
- $t$  won't reveal anything about  $id!$
- However, given two equations (for the same  $\#, id, r$ )  
     $t_1 = id + r c_1$   
     $t_2 = id + r c_2$   
    one can solve for  $id.$

# Off-line E-cash: Withdrawal

choose random  $\#, r, s'$   
and compute

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



$C + \text{proof}$

$(c, e, s'')$



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

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

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



# Off-line E-cash: Payment

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

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

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



compute

$$t = r + u \text{nym} \pmod{q}$$

$$c' = c b^{s'} \pmod{n}$$

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

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

choose random  $u$

# Off-line E-cash: Payment

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

$$d / a_1^{\#} = c'^{\varepsilon} a_2^{\rho} a_3^{\mu} b^{\sigma} \pmod{n} \wedge g^{\dagger} = g^{\rho} (g^u)^{\mu} \}$$

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

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

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

$\Rightarrow \dagger = \rho + u\mu \pmod{q},$

i.e.,  $\dagger$  was computed correctly!

# Off-line E-cash: Deposit

#  $\in$  L?

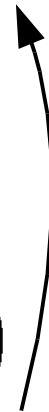
If so:

1.  $t = \rho + u \mu \pmod{q}$

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

solve for  $\rho$  and  $\mu$ .

$\Rightarrow \mu = nym$  because of proof



$u, t, \#, \text{proof}$



# Off-line E-cash: Security

---

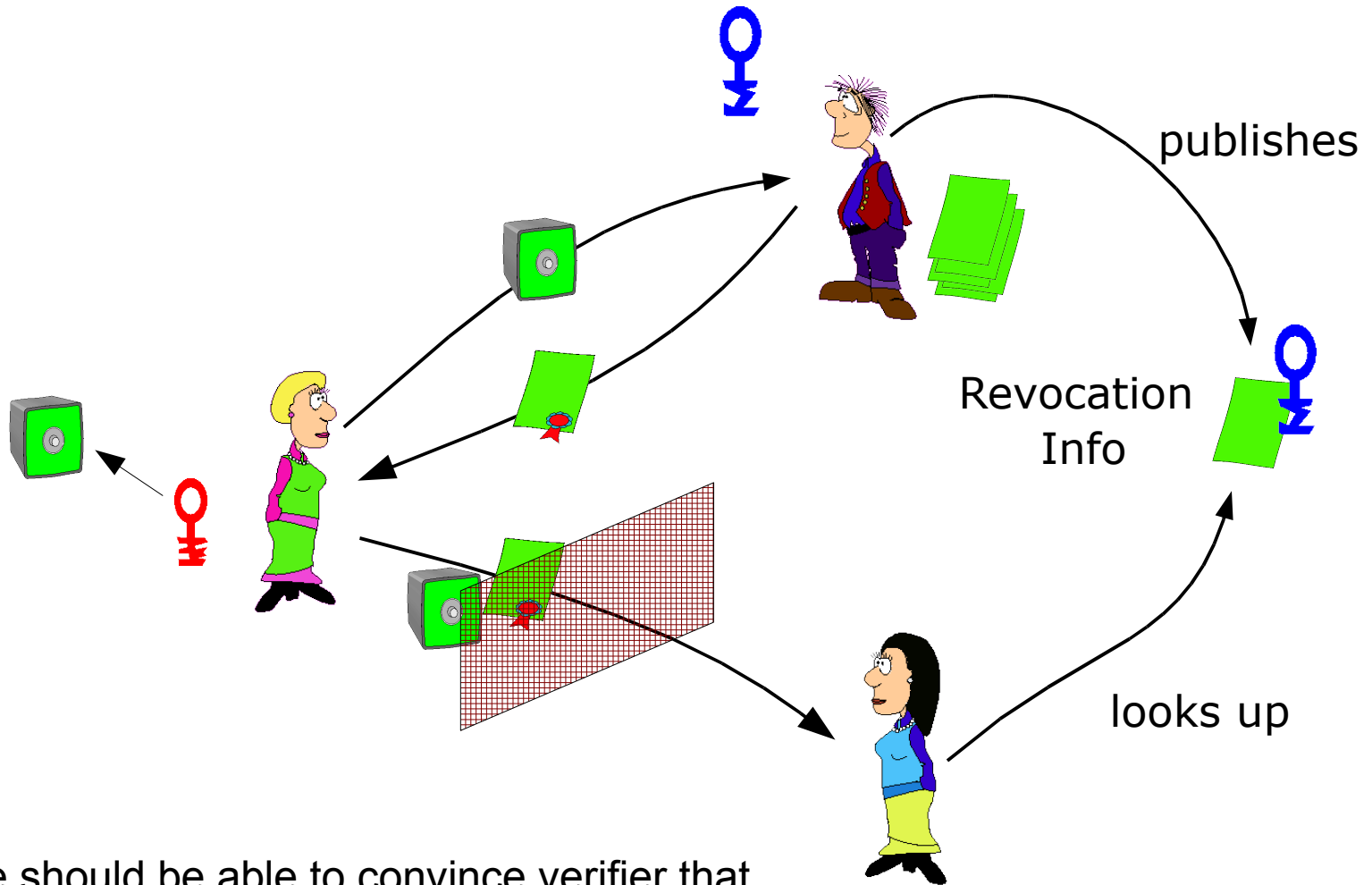
- Unforgeable:
  - no more coins than #'s, otherwise one can forge signatures
  - if coins with same # appears with different c's => reveal nym
- Anonymity:
  - # and r are hidden from signer upon withdrawal
  - † does not reveal anything about nym (is blinded by r)
  - proof proof does not reveal anything

# Application

---

Revocable Credentials

# Revocable Credentials: Possible Solutions



Alice should be able to convince verifier that her credential is among the good ones!

# Revocable Credentials: First Solution

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

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

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

$$(u_1, \dots, u_k)$$

- Alice proves that her  $u_i$  is on the list.

- Compute  $U_j = g^{u_j}$  for  $u_j$  in  $(u_1, \dots, u_k)$

- Prove PK $\{(\varepsilon, \mu, \rho, \sigma) : d = c'^{\varepsilon} a_1^{\rho} a_2^{\mu} b^{\sigma} \pmod{n} \wedge$

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

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

# Revocable Credentials: Second Solution

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

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

- Publish list of all *invalid*  $u_i$ 's.

$$(u_1, \dots, u_k)$$

- Alice proves that her  $u_i$  is on the list.

- Choose random  $h$  and compute  $U = h^{u_i}$

- Prove  $\text{PK}\{(\varepsilon, \mu, \rho, \sigma) : d = c'^{\varepsilon} a_1^{\rho} a_2^{\mu} b^{\sigma} \pmod{n} \wedge U = h^{\mu}\}$

- Verifier checks whether  $U = h^{u_j}$  for all  $u_j$  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  $u_i$ 's public?

# Revocable Credentials: Second Solution

---

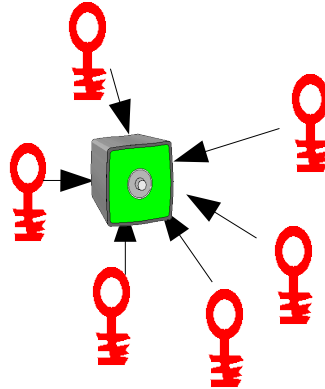
Variation: verifier could choose  $h$  and keep it fixed for a while

- Can pre-compute list  $U_i = h^{u_i}$
- -> 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  $h = H(\text{verifier}, \text{date})$ , so user can check correctness.
  - $\text{date}$  could be the time up to seconds and the verifier could just store all the lists, i.e., pre-compute it.

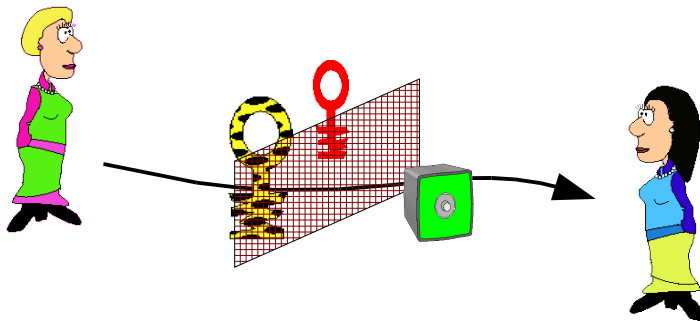
# Revocable Credentials: Third Solution

Using so-called cryptographic accumulators:

- Accumulate:



- Prove that your key is in accumulator: requires witness 

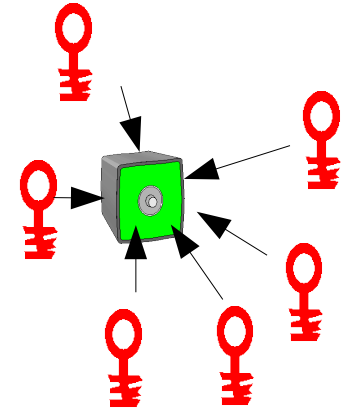


- Security: must work only for values contained in acc. 

# Revocable Credentials: Third Solution

Using so-called cryptographic accumulators:

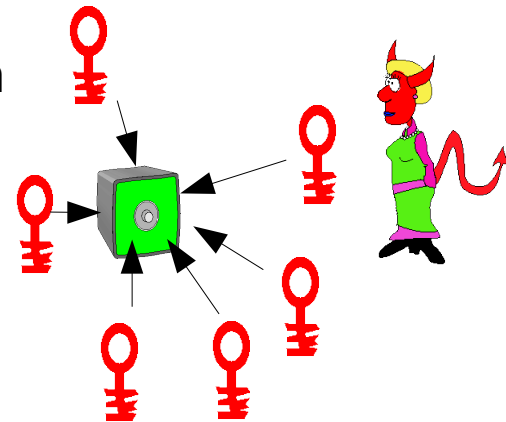
- Key setup: RSA modulus  $n$ , seed  $v$
- Accumulate:
  - values are primes  $e_i$
  - accumulator value:  $z = v^{\prod e_i} \bmod n$
  - publish  $z$  and  $n$
  - witness value  $x$  for  $e_j$ : s.t.  $z = x^{e_j} \bmod n$   
can be computed as  $x = v^{e_1 \dots e_{j-1} \cdot e_{j+1} \dots e_k} \bmod n$
- 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.  $z = x^e \pmod n$  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  $z' = v^{\prod e_i} = z^{1/e_{rev}} \pmod n$

- Witness:

- Use Ext. Euclid to compute  $a$  and  $b$

$$\text{s.t. } a e_{own} + b e_{rev} = 1$$

- Now  $x' = x^b z'^a \pmod n$

- Why: 
$$\begin{aligned} x'^{e_{own}} &= ((x^b z'^a)^{e_{own}})^{e_{rev}^{-1}} \pmod n \\ &= ((x^b z'^a)^{e_{own} e_{rev}})^{1/e_{rev}} \pmod n \\ &= ((x^{e_{own}})^b e_{rev} (z'^{e_{rev}})^a e_{own})^{1/e_{rev}} \pmod n \\ &= (z^{b e_{rev}} z^{a e_{own}})^{1/e_{rev}} \pmod n \\ &= z^{1/e_{rev}} \pmod n \end{aligned}$$

# 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:  $z = v^{\prod e_i} \pmod n$
  - Actually  $v$  never occurs anywhere...  
so:  $v' = v^{1/new} \pmod n$  and  $x = z^{1/new} \pmod n$
  - Thus  $z$  needs not to be changed in case new member joins!
- Witnesses need to be recomputed upon revocation only!

# References

---

- Stefan Brands: *Untraceable Off-line Cash in Wallets With Observers*. In *Advances in Cryptology – CRYPTO '93*. Springer Verlag, 1993.
- Brickell, Camenisch, Chen: *Direct Anonymous Attestation*. ACM CSS 2004, ACM Press
- 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.