CRYPTOGRAPHY

(lecture 7)

Literature:

"A Graduate Course in Applied Cryptography" (ch **13.3, 19.3, 8.10.2** until pg324)
"A note on blind signature schemes" by Matthew Green
"Blind Signatures for Untraceable Payments" by David Chaum, "Digital Signatures" by Tibor Jager
"Lecture Notes on Cryptographic Protocols" by Schoenmakers (ch **8.0,8.1,8.2**)
"Group Signatures: Authentication with Privacy" (ch **1.1.1, 1.2**, **1.3.0, 1.3.1**, 1.4, **1.5.0, 1.5.1**, 1.5.2, **1.5.3, 1.6.4**)
"The Mathematics of Elliptic Curve Cryptography" (on Canvas)

Module 2: Agenda

OW(Trapdoor)Functions
DH Key-Exchange
DL, CDH, DHH
Number Theory
RSA, ElGamal Cryptosystems
IND-CPA and IND-CCA

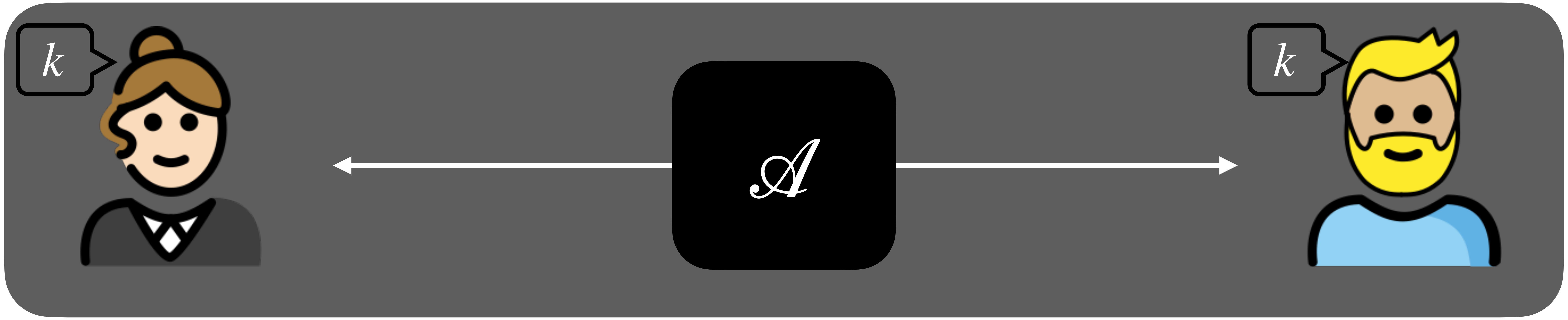
Digital Signatures
- Problem Statement
- Syntax
- RSA Signatures
- The Hash-and-Sign Paradigm
- Proof

Elliptic Curve Cryptography
- Brief Math Background
- ECDSA

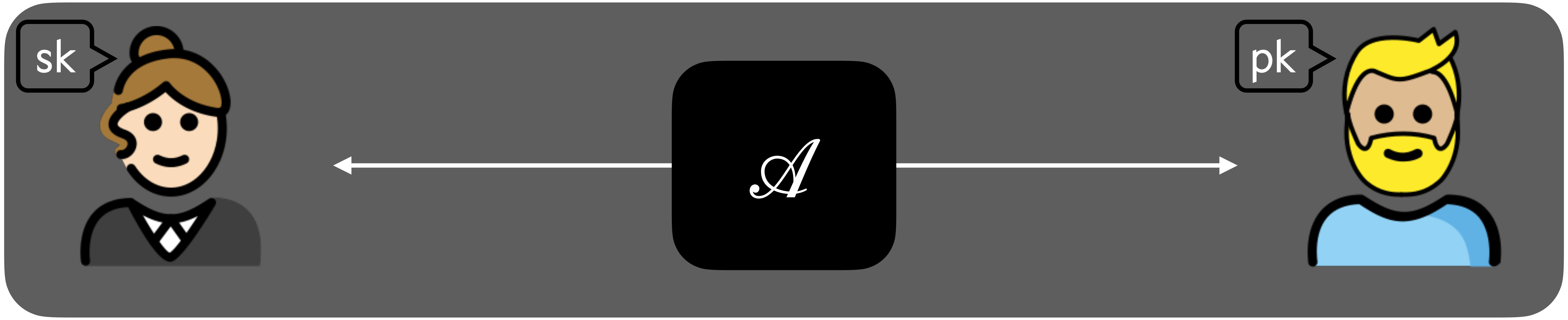
Advanced Properties for Signatures
- Group Signatures
- Blind Signature
- Application: Untraceable eCash

Secure Instant Messaging
Post Quantum Cryptography
The Birthday Paradox

Authenticating the Source of Information Over the Internet



Problem: if both Alice and Bob know k, then cryptographically they are the same person. Bob cannot convince a third party that Alice has produced something (e.g. a MAC) that requires the knowledge of k. *Whatever Alices produces, Bob can produce it as well*!



With **public key cryptography** Alice is the only one to know *sk.* If she uses it to do something that is (computationally) impossible to do without *sk*, then everyone can be convinced she did it.

Digital Signature - Syntax

Definition: Digital Signature

A digital signature scheme is a triple of PPT algorithms $(KeyGen, Sign, Ver)$ defined as follows:

- $KeyGen(n) \to (\mathsf{pk}, \mathsf{sk})$ is a probabilistic key generation algorithm
- $Sign(\mathsf{sk}, m) \to \sigma$ is a (possibly) probabilistic algorithm that outputs a signature σ for a message m
- $Ver(\mathsf{pk}, m, \sigma)$ is a deterministic algorithm that returns '1' (accept) if σ is considered valid for m against pk, or '0' (reject) otherwise.

Correctness

For all key pairs $(\mathsf{pk}, \mathsf{sk}) \leftarrow KeyGen(n)$ it holds that: $Ver(\mathsf{pk}, m, Sign(\mathsf{sk}, m)) = 1$

$$Pr[Ver(\mathsf{pk}, m, \sigma) = 1 \,|\, \sigma \leftarrow Sign(\mathsf{sk}, m)] = 1$$

Towards a Security Notion for Digital Signatures

$\mathscr{A}$

Adversary's Power and Knowledge

Key-Only Attack: $\mathscr{A}$ knows only the singer's *pk*, and therefore only has the capability of checking the validity of signatures of messages

Known Signature Attack: $\mathscr{A}$ knows *pk* and sees message/signature pairs chosen and produced by the legal signer

Chosen Message Attack: $\mathscr{A}$ knows *pk* and can ask the signer to sign a number of messages of the adversary's choice.

Adversary's Goal

Existential Forgery: $\mathscr{A}$ succeeds in creating a valid signature of a new message (never seen before)

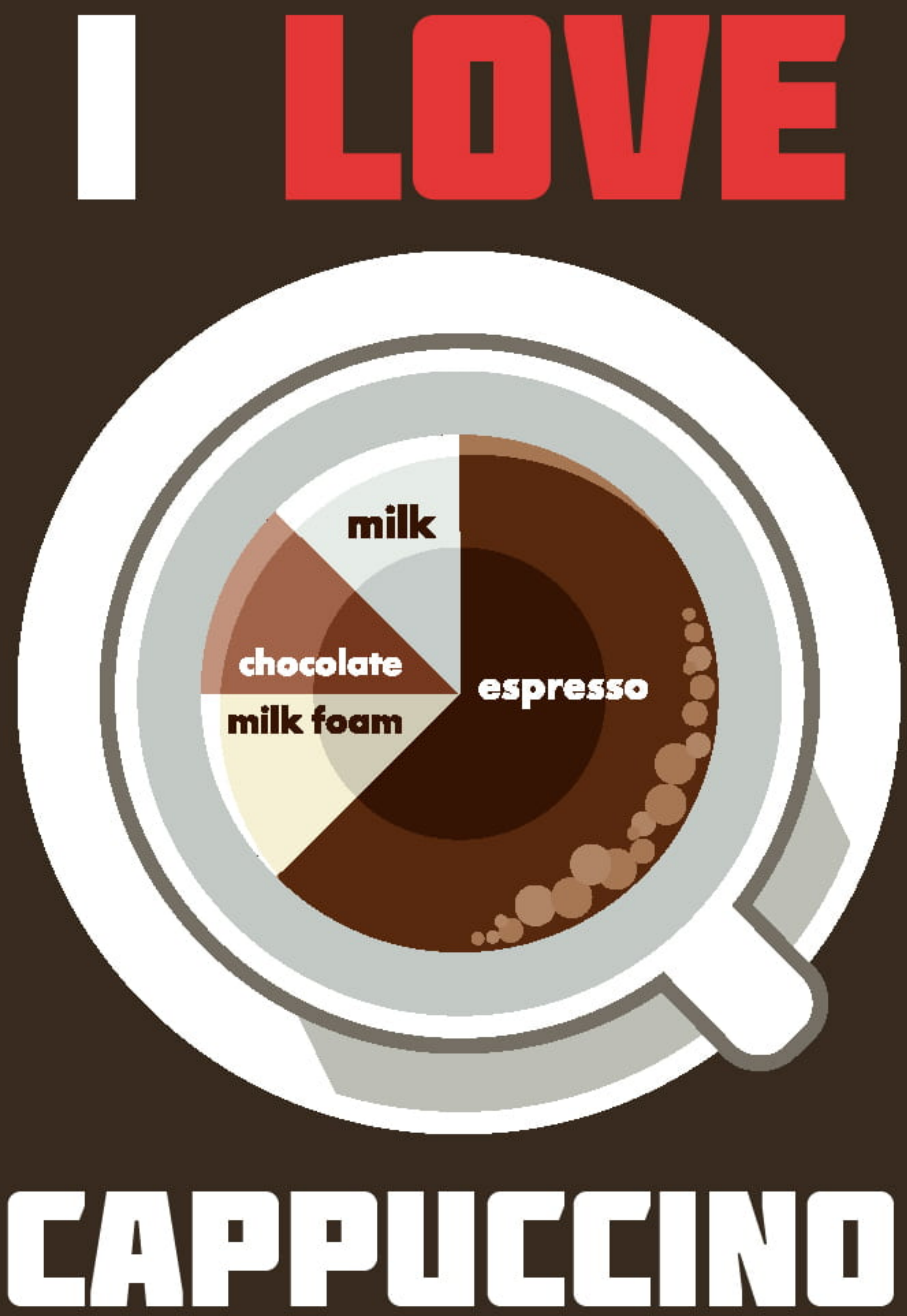
Strong Forgery: $\mathscr{A}$ succeeds in creating a valid signature of some message of $\mathscr{A}$'s choice **and** the signature is different from any signature seen by $\mathscr{A}$

Universal Forgery: $\mathscr{A}$ is able to generate a valid signature for *any* message (but ignores *sk*)

Total Break: $\mathscr{A}$ can compute the signer's secret key *sk*

The recipe for a good security notion:

1. Choose a realistic adversary (PPT, Quantum…)
2. Give to $\mathscr{A}$ the strongest starting knowledge
3. Select the weakest damage to the cryptosystem
4. DONE!



Towards a Security Notion for Digital Signatures

Adversary's Power and Knowledge

Key-Only Attack: $\mathscr{A}$ knows only the singer's *pk*, and therefore only has the capability of checking the validity of signatures of messages *(a bit unrealistic)*

Known Signature Attack: $\mathscr{A}$ knows *pk* and sees message/signature pairs chosen and produced by the legal signer *(in reality, this the minimum one should assume)*

Chosen Message Attack: $\mathscr{A}$ knows *pk* and can ask the signer to sign a number of messages of the adversary's choice. *(this is our standard)*

Adversary's Goal

Existential Forgery: $\mathscr{A}$ succeeds in creating a valid signature of a new message (never seen before)

Strong Forgery: $\mathscr{A}$ succeeds in creating a valid signature of some message of $\mathscr{A}$'s choice **and** the signature is different from any signature seen by $\mathscr{A}$

Universal Forgery: $\mathscr{A}$ is able to generate a valid signature for *any* message (but ignores *sk*)

Total Break: $\mathscr{A}$ can compute the signer's secret key *sk*

Existential Unforgeability Under Chosen Message Attack (EUF-CMA)

Aim: quantify the $\mathscr{A}$'s likelihood in forging a valid signature σ^* for a ***new*** message m^*

$\mathscr{C}$

$KeyGen(n) \rightarrow (\mathsf{pk}, \mathsf{sk})$

$Sign(\mathsf{sk}, m_i) \rightarrow \sigma_i$

$\xrightarrow{\mathsf{pk}}$

$for\ i = 1, \ldots, Q_M = poly(n)$

$\xleftarrow{m_i}$

adaptive queries

$\xrightarrow{\sigma_i}$

$\xleftarrow{(m^*, \sigma^*)}$

$\mathscr{A}$

win or **lose**

$\mathscr{A}$ wins the security game iff:

$Ver(\mathsf{pk}, m^*, \sigma^*) = 1$ **AND** $m^* \notin \{m_1, \ldots, m_{Q_M}\}$

Secure Signature

A Digital Signature Scheme is said to be **secure** (unforgeable under chosen message attack) if **for all efficient** adversaries the probability that $\mathscr{A}$ **wins** the EUF-CMA security game is **negligible**. Formally,

$$Pr[Ver(\mathsf{pk}, m^*, \sigma^*) = 1 \,|\, (m^*, \sigma^*) \leftarrow \mathscr{A}^{\mathcal{O}_{\mathsf{sk}}^{Sign}}(\mathsf{pk}) \wedge m^* \notin \{m_i\}_{i=1}^{Q_M}] \leq negl(n)$$

Textbook RSA Signature Scheme

KeyGen (sec.par) ⇨ (sk, pk)

Pick: p,q two distinct sec.par-bit long primes

Compute: $N=p\cdot q$, and e,d s.t. $e\cdot d=1 \bmod \Phi(N)$

sk = (N, d)

pk = (N, e)

Sign (sk, m) ⇨ σ

The message is m in $\mathbb{Z}_N$

Compute: $\sigma = m^d \bmod N$

Ver (pk, m, σ) ⇨ {0, 1}

Check: $m = \sigma^e \bmod N$?

🧐 *Is this construction EUF-CMA secure?*

[No! Because RSA is **homomorphic**]

The RSA-FDH Signature Scheme

KeyGen (sec.par) ⇨ (sk, pk)
Pick: p,q two distinct sec.par-bit long primes
Compute: $N=p\cdot q$, and e,d s.t. $e\cdot d=1 \bmod \Phi(N)$
sk = (N, d)
pk = (N, e)

Sign (sk, msg) ⇨ σ
Hash the message: H(msg)=h
Compute: $\sigma = h^d \bmod N$

Verify (pk, msg, σ) ⇨ {0, 1}
Hash the message: H(msg)=h
Check: $h = \sigma^e \bmod N$

🧐 *Can we use sha256?*

[No! We need a long-output hash function *full domain* hash (FDH), N~2048bits]

A More General Look: the Hash-and-Sign Paradigm

```
KeyGen (sec.par) ⇨ (sk, pk)
 Pick: p,q two distinct sec.par-bit long primes
 Compute: N=p·q, and e,d s.t. e·d=1 mod Φ(N)
 sk = (N, d)
 pk = (N, e)
```

```
Sign (sk, msg) ⇨ σ
 Hash the message: H(msg)=h
 Compute: σ = h^d mod N
```

```
Verify (pk, msg, σ) ⇨ {0, 1}
 Hash the message: H(msg)=h
 Check: h = σ^e mod N
```

<u>Full Domain Hash + One-Way Trapdoor Permutation = Secure Digital Signature</u>

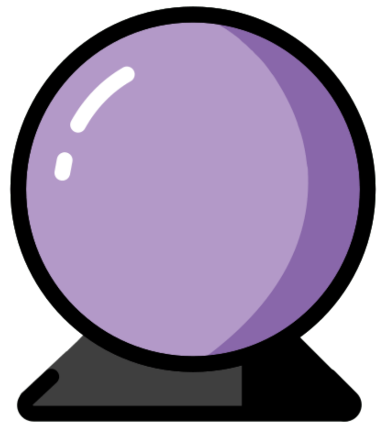
Sig.KeyGen : $OWTF\,.\,KeyGen(n) \rightarrow$ (pk, sk)

Sig.Sign(sk, msg) : $I(\text{sk}, H(msg)) = \sigma$

Sig.Ver(pk,msg,σ) : test $F(\text{pk}, \sigma) = H(msg)$?

Security Proof

The RSA-FDH signature scheme is EUF-CMA secure in the Random Oracle Model under the RSA assumption ***[given (N,e,c) find m such that $c^d = m \bmod N$].***

The hash function H is modelled as if it was a truly random function $\mathscr{O}$

How do we prove security? As in Module1, proof by contradiction.

Reasoning: **if** $\mathscr{A}$ *breaks* the EUF-CMA security of RSA-FDH with non-negligible probability, **then** we can build a new adversary (called *reduction*) $\mathscr{B}$ that *uses* $\mathscr{A}$ to *break* the *RSA assumption*, with non-negligible probability.

Proof: the Reduction

$\mathscr{B}$ simultaneously acts as attacker against the RSA problem and as challenger in the EUF-CMA security game with $\mathscr{A}$

RSA challenger

$\mathscr{C}$

RSA setting:
N=pq
ed =1 mod ϕ(N)
c $\leftarrow \$\mathbb{Z}_N$

$\longrightarrow (N, e, c) \longrightarrow$

$\longleftarrow \tilde{m}^* \longleftarrow$

Answering R.O. queries

Give consistent replies.

For a new message $(m_j, \cdot, \cdot) \notin L$

$r_j \leftarrow \$\mathbb{Z}_N^*$

With probability f: $h_j \leftarrow r_j^e \mod N$

With probability (1-f): $h_j \leftarrow c \cdot r_j^e \mod N$

Store (m_j, h_j, r_j) in L, return h_j

Answering Signing queries

If $(m_i, \cdot, \cdot) \notin L$: call R.O.

If $(m_i, \cdot, \cdot) \in L$: check:

if $h_i = r_i^e \mod N$: return $\sigma_i = r_i$

if $h_i = c \cdot r_i^e \mod N$: *Abort*

$\xrightarrow{pk = (N, e)}$

$\xleftarrow{m_j \text{ (R.O.)}}$

$\xrightarrow{h_j}$

$\xleftarrow{m_i}$

$\xrightarrow{\sigma_i}$

$\xleftarrow{(m^*, \sigma^*)}$

$\mathscr{A}$

Proof: the Reduction

$\mathscr{B}$ acts simultaneously as attacker against the RSA problem and as challenger in the EUF-CMA security game with $\mathscr{A}$

RSA challenger

$\mathscr{C}$

RSA setting:
N=pq
ed =1 mod ϕ(N)
c $\leftarrow \$\mathbb{Z}_N$

$\xrightarrow{(N, e, c)}$

With probability f: $h_j \leftarrow r_j^e \mod N$

With probability (1-f): $h_j \leftarrow c \cdot r_j^e \mod N$

Store (m_j, h_j, r_j) *in* L, return h_j

if $h_i = r_i^e \mod N$: return $\sigma_i = r_i$

if $h_i = c \cdot r_i^e \mod N$: *Abort*

$\xrightarrow{\text{pk} = (N, e)}$

$\xleftarrow{m_j \text{ (R.O.)}}$

$\xrightarrow{h_j}$

$\xleftarrow{m_i}$

$\xrightarrow{\sigma_i}$

$\mathscr{A}$

$\xleftarrow{\tilde{m}^*}$

If there exists an index i s.t.

1) $H(m^*) = h_i = c \cdot (r_i)^e \mod N$

And

2) $Ver(\text{pk}, m^*, \sigma^*) = 1$

Return to $\mathscr{C}$: $\tilde{m}^* = \sigma^* \cdot r_i^{-1} \mod N$

$\xleftarrow{(m^*, \sigma^*)}$

Proof: Finalising the Reasoning

Now we have a full description of the reduction $\mathscr{B}$. We need to prove a few properties:

1) $\mathscr{B}$ perfectly simulates the EUF-CMA game to $\mathscr{A}$:

- The values h_j returned by $\mathscr{B}$ look random 👍 *because* $r_j \leftarrow \$\mathbb{Z}_N^*$
- The signatures σ_i look proper 👍 *because when $\mathscr{B}$ does not abort,* $\sigma_i = r_i$ *and* $H(m_i) = h_i = r_i^e \mod N$. *So* $\sigma_i^e = r_i^e = H(m) \mod N$

2) $\mathscr{B}$'s output is a correct.

👍 *because* $Ver(\mathsf{pk}, m^*, \sigma^*) = 1$ *iff* $(\sigma^*)^e = H(m^*) = c \cdot r_i^e = (c^d \cdot r_i)^e \mod N$ *iff* $\sigma^* = c^d \cdot r_i$

(Proof: Cleaning the Details - Not Needed for the Exam)

3) $\mathscr{B}$ does not abort with probability f^{Q_M}.

5) If $\mathscr{B}$ works (i.e., it does not abort), then $\mathscr{B}$ can use $\mathscr{A}$'s forgery to break RSA (invert the encryption) with probability 1-f.

5) If $\mathscr{A}$ succeeds with non-negligible probability δ then $\mathscr{B}$ succeeds with non-negligible probability

$$\frac{(1-f) \cdot f^{Q_M} \cdot \delta}{Q_M}$$

For missing details check "On the exact security of full domain hash" or these slides

Module 2: Agenda

OW(Trapdoor)Functions
DH Key-Exchange
DL, CDH, DHH
Number Theory
RSA, ElGamal Cryptosystems
IND-CPA and IND-CCA

Digital Signatures
- Problem Statement
- Syntax
- RSA Signatures
- The Hash-and-Sign Paradigm
- Proof

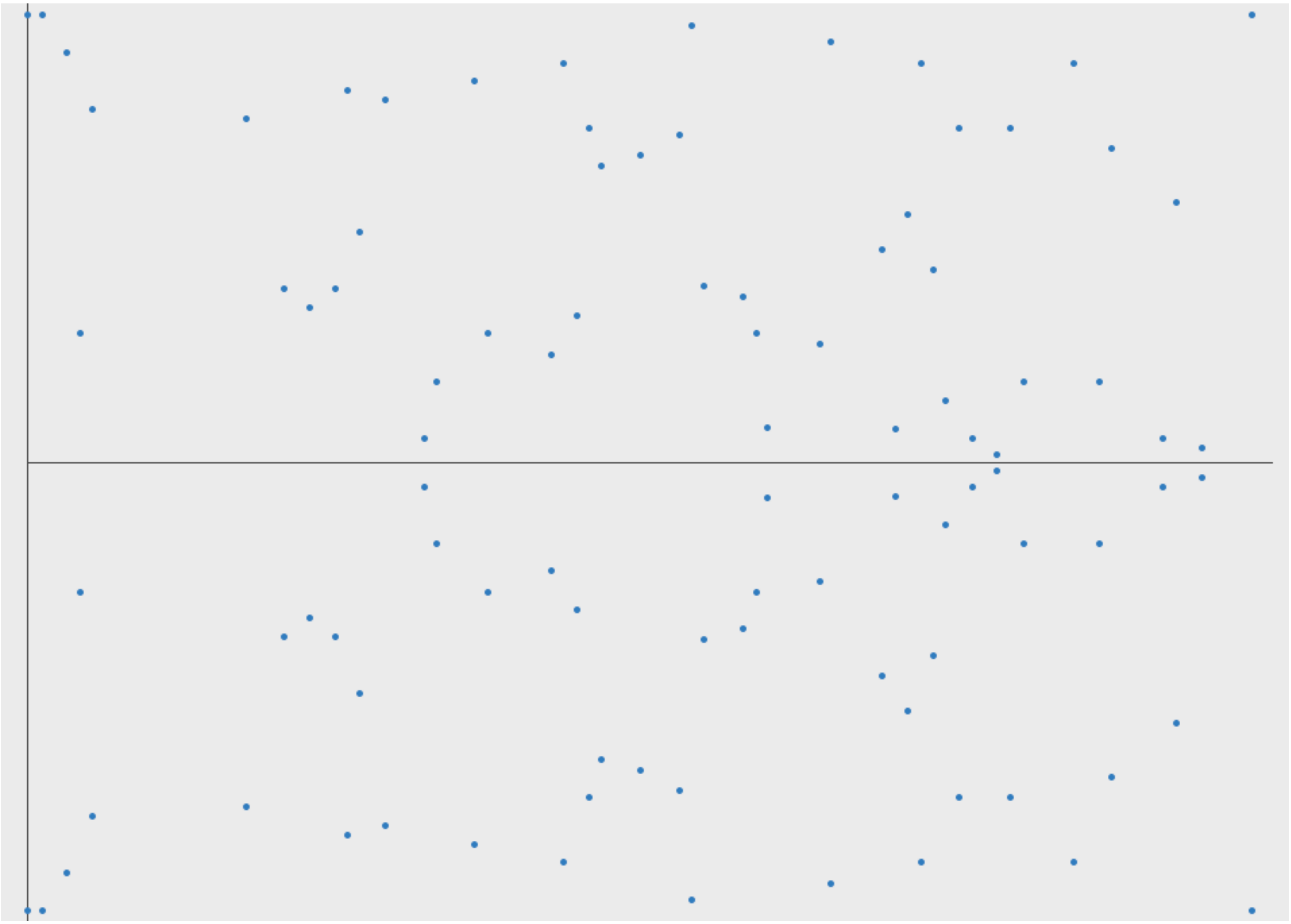
Elliptic Curve Cryptography
- Brief Math Background
- ECDSA

Advanced Properties for Signatures
- Group Signatures
- Blind Signature
- Application: Untraceable eCash

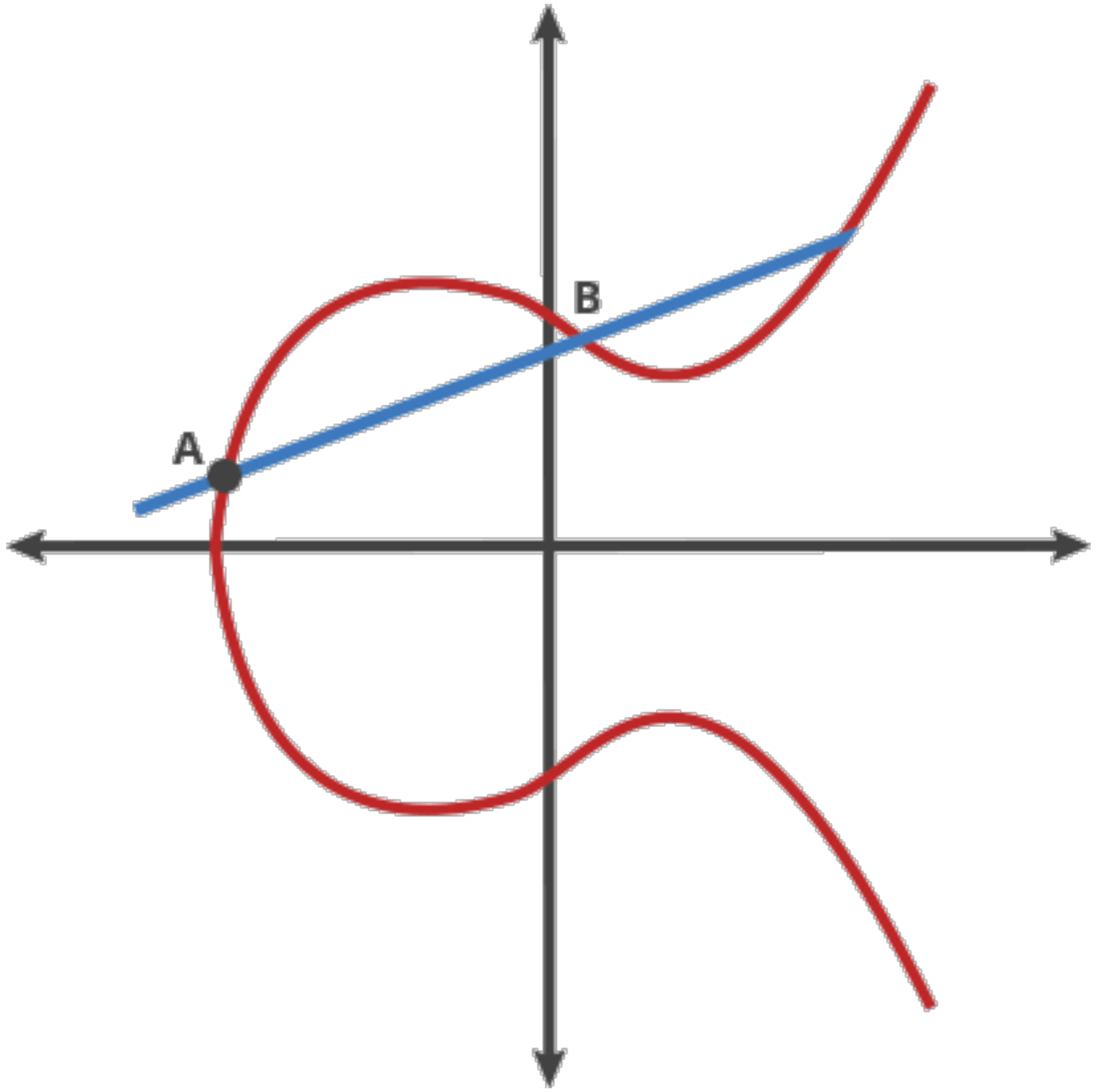
Secure Instant Messaging
Post Quantum Cryptography
The Birthday Paradox

ECDSA - Background on Elliptic Curve Cryptography

$$y^2 = x^3 - x + 1 \mod 97$$



$$y^2 = x^3 + ax + b$$



*Elliptic curves have a **group** structure*

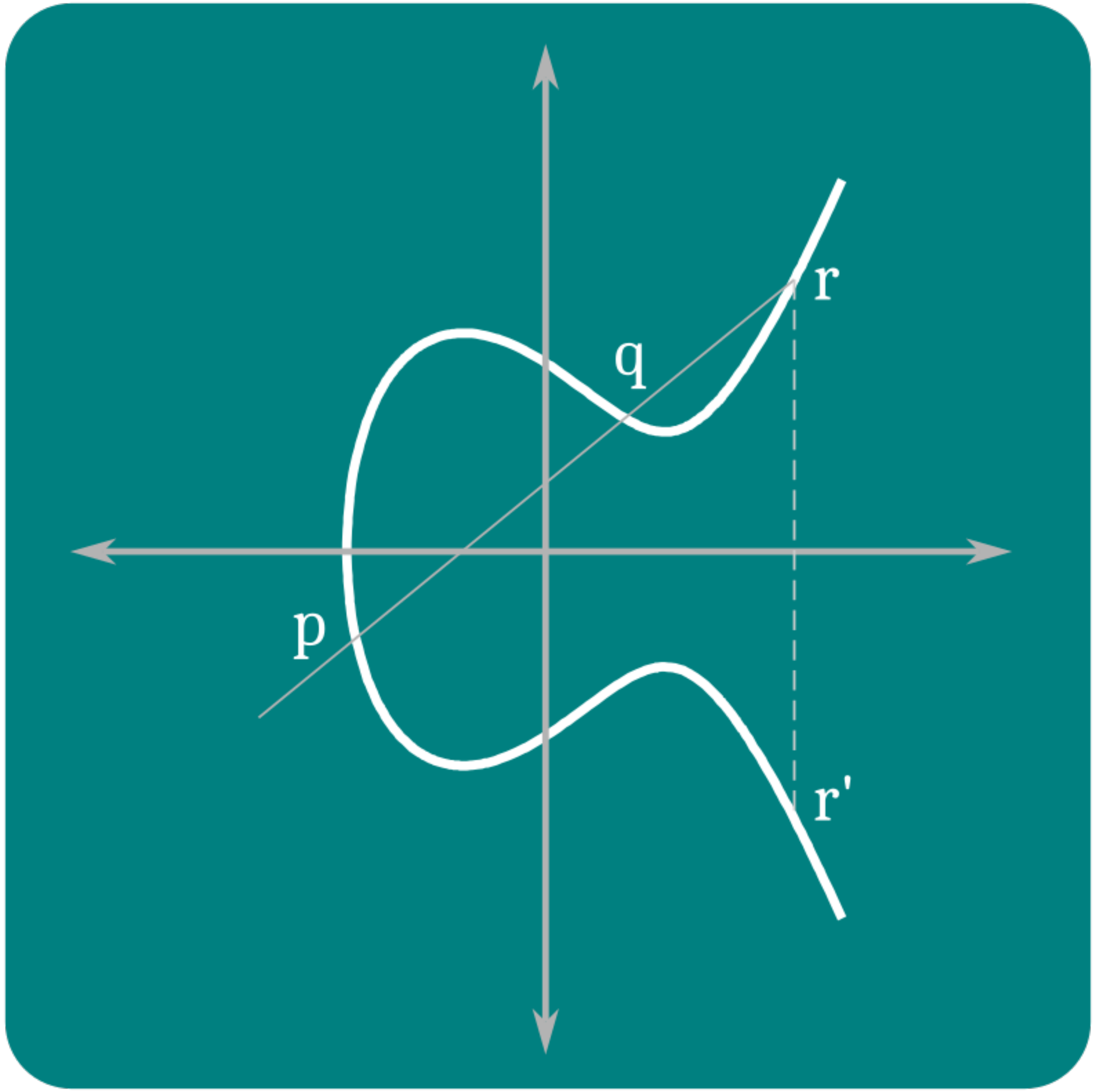
[gifs from arstechnica]

ECDSA - Algorithms

```
KeyGen (sec.par) ⇨ (sk, pk)
  d ←$── [0 ... n-1]
  sk = d
  pk = Q = d*G
```

```
Sign (sk, msg) ⇨ sgn
  k ←$── [0 ... n-1]
  R = k*G
  r = R_x mod n
  z = sha256(msg)
  s = inv(k)·(z + d·r) mod n
  sgn = (r, s)
```

```
Verify(pk, msg, sgn) ⇨ {0, 1}
  z = sha256(msg)
  T = [z·inv(s) mod n]*G
  P = [inv(s)·r mod n]*Q
  if R == T+P return 1
  else return 0
```



ECDSA - the Good

- ★ Shorter keys and better security than the RSA signature scheme
- ★ Non malleable
- ★ IoT friendly
- ★ In wide adoption (TLS, DigiCert (Symantec), Sectigo (Comodo) …)

ECDSA - the Bad

PS3 hacked through poor cryptography implementation

repeated nonce attack Bonus 2

A group of hackers named fail0verflow revealed in a presentation how they ...

CASEY JOHNSTON - 12/30/2010, 6:25 PM

what happens if the same nonce k is used to sign two different messages?

```
k ←$── [0 ... n-1]
R = k*G
r = R_x mod n
z = sha256(msg)
s = inv(k)·(z + d·r) mod n
```

{* SECURITY *}

Android bug batters Bitcoin wallets

Old flaw, new problem

Richard Chirgwin

Mon 12 Aug 2013 // 00:43 UTC

LadderLeak: Side-channel security flaws exploited to break ECDSA cryptography

Charlie Osborne 28 May 2020 at 14:07 UTC

Updated: 28 June 2021 at 09:05 UTC

What now?

EdDSA

Check out this blog for comparison between ECDSA and EdDSA ('conclusions' gives a very good summary)

Advanced Properties for Digital Signatures

Attribute-Based Signatures

Group Signatures

Key-Homomorphic Signatures

Forward Secure Signatures

Threshold Signatures

Aggregate Signatures

Identity-Based Signatures

Ring Signatures

Blind Signatures

Homomorphic Signatures

Functional Signatures

Structure Preserving Signatures

Anonymous Signatures

Proxy Signatures

Redactable Signatures

Multi Signatures

Sequential Signatures

Module 2: Agenda

OW(Trapdoor)Functions
DH Key-Exchange
DL, CDH, DHH
Number Theory
RSA, ElGamal Cryptosystems
IND-CPA and IND-CCA

Digital Signatures
- Problem Statement
- Syntax
- RSA Signatures
- The Hash-and-Sign Paradigm
- Proof

Elliptic Curve Cryptography
- Brief Math Background
- ECDSA

Advanced Properties for Signatures
- Group Signatures
- Blind Signature
- Application: Untraceable eCash

Secure Instant Messaging
Post Quantum Cryptography
The Birthday Paradox

Group Signatures

Group Signatures

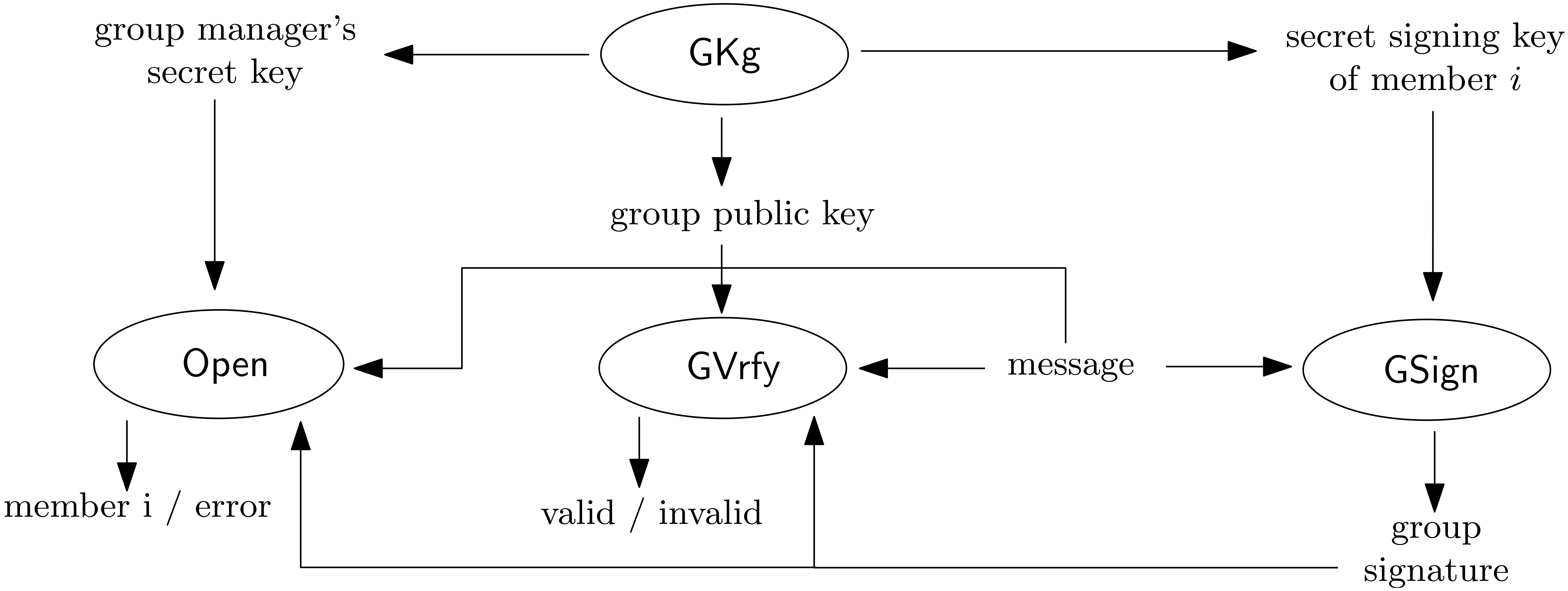


Figure 1.1.: Static Group Signatures

Blind Signatures

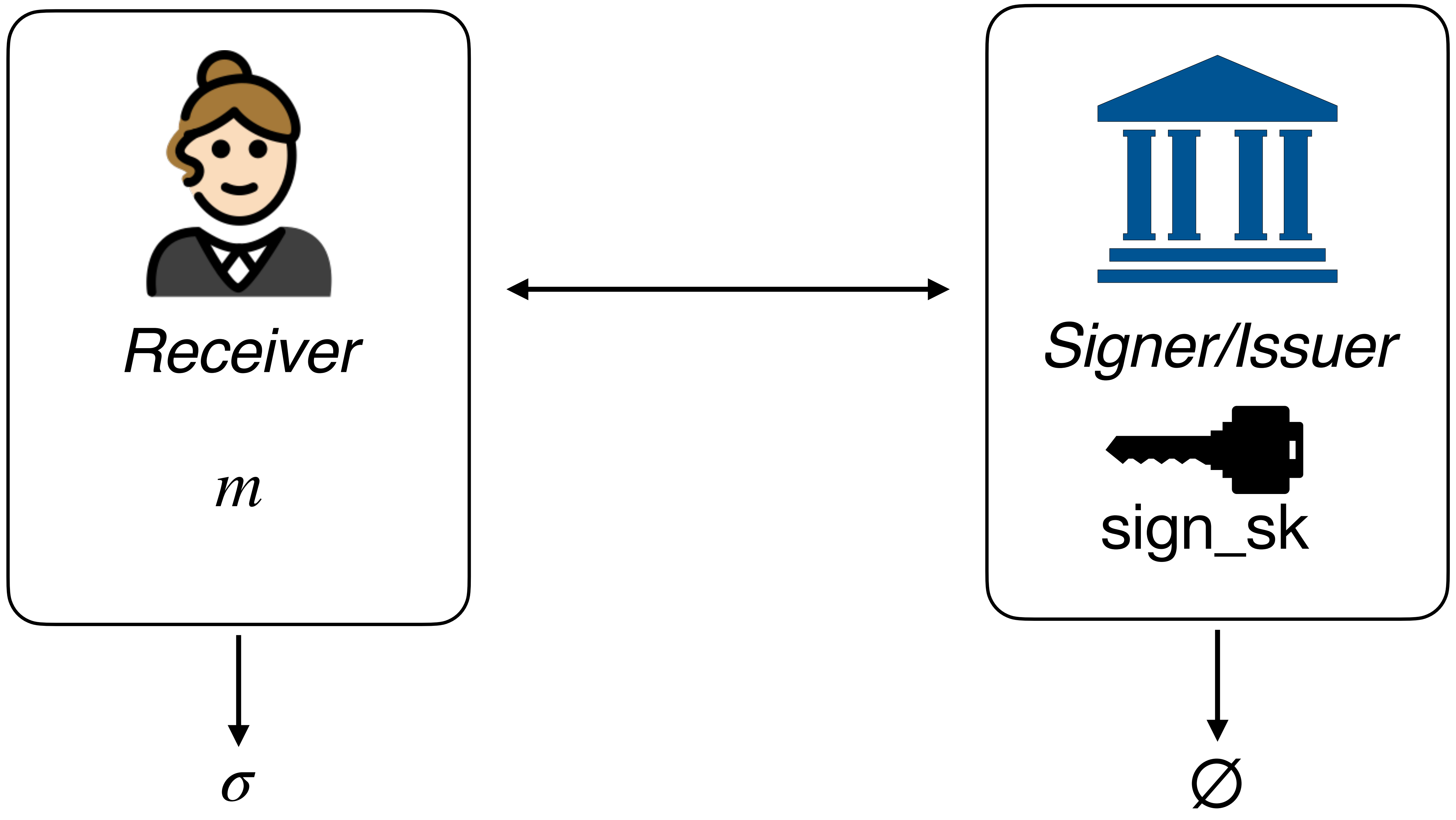
Blind Signatures

Definition: Blind Signature

A blind signature scheme is a signature scheme where the signing algorithm algorithms $Sign$ is replaced by an *interactive protocol* run between a signer/issuer (S) and a receiver (R).

The protocol starts with R who has as input a message m, and S who has as input a secret key sk.

At the end of the interaction R obtains a signature σ on m, and S learns nothing about m or σ.

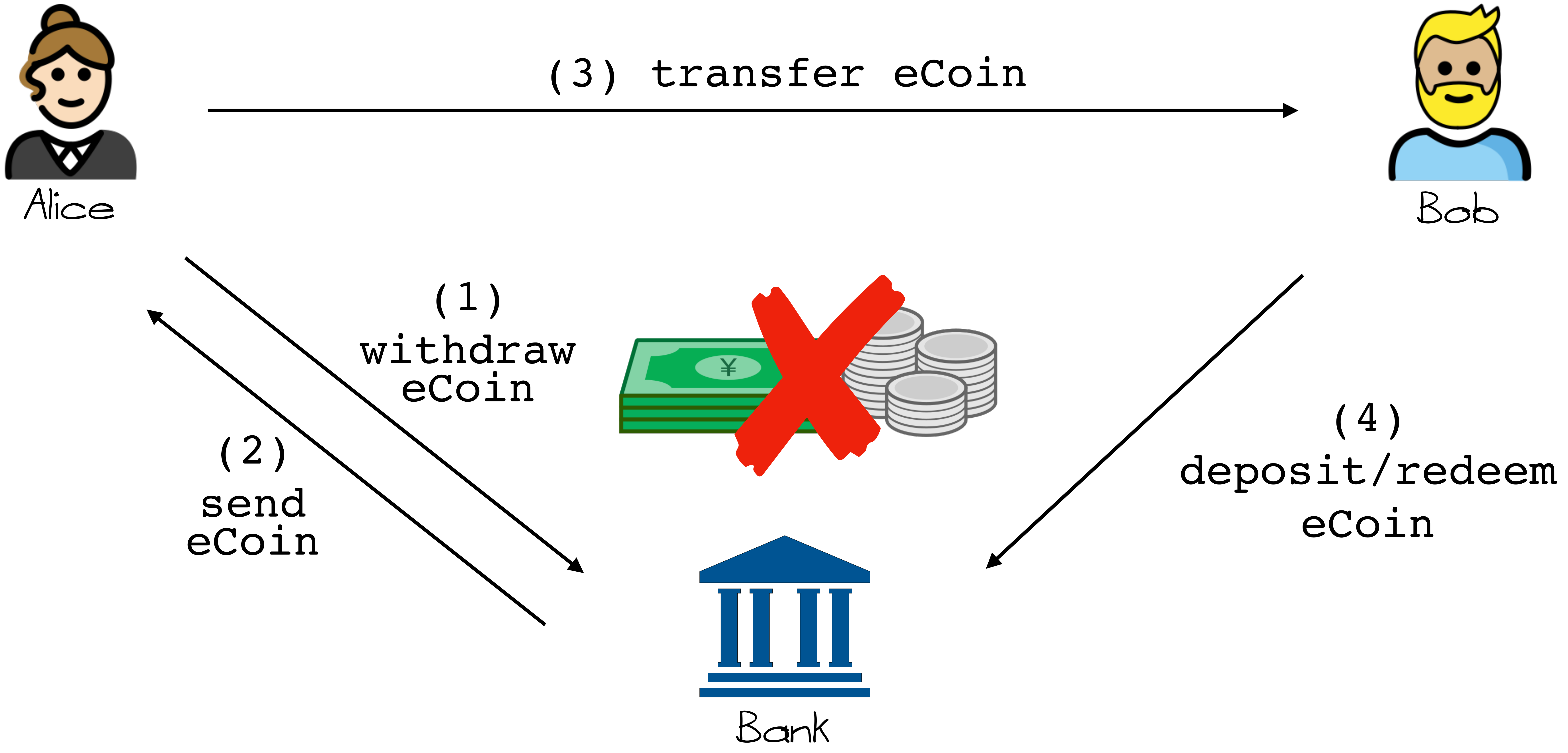


🧐 *where can this be useful?*

untraceable electronic payment system

attribute-based credentials *[ABC, lecture 12 by Victor]*

Chaum's Untraceable eCash System



Property Wishlist

1. Only the Bank can generate eCoins
2. Users cannot double spend eCoins (money cloning)
3. eCoins should be untraceable, like physical cash

1. How To Make Sure Only the Bank Creates eCoins?

Solution: eCoin is a bit string together with a digital signature generated using the Bank's sk

unforgeability ensures that $\mathscr{A}$ cannot generate eCoins

2. How To Prevent Double Spending?

Easy option:

report to the bank every eCoin ever spent (upon payment the eCoin looses its value, the bank produces a new eCoin of the appropriate value for the seller)

🧐 *does this work?*

A *better* option:

remove buyer anonymity only if (s)he attempts to double spend a eCoin (blind signatures)

2&3 Prevent Double Spending and Keep eCoins Untraceable

Aim: the Bank should be able to sign an eCoin, **without knowing** *what* eCoin it is

The eCoin withdrawal procedure with RSA (blind) signatures

Alice:

pick a random r

$$B = r^e H(x) \mod n$$

$\xrightarrow{B}$ blinded message

Bank:

$$\bar{S} = (B)^d \mod n$$

$\xleftarrow{\bar{S}}$ blind signature

Alice:

extract signature for x

$$S = r^{-1}\bar{S} \mod n$$

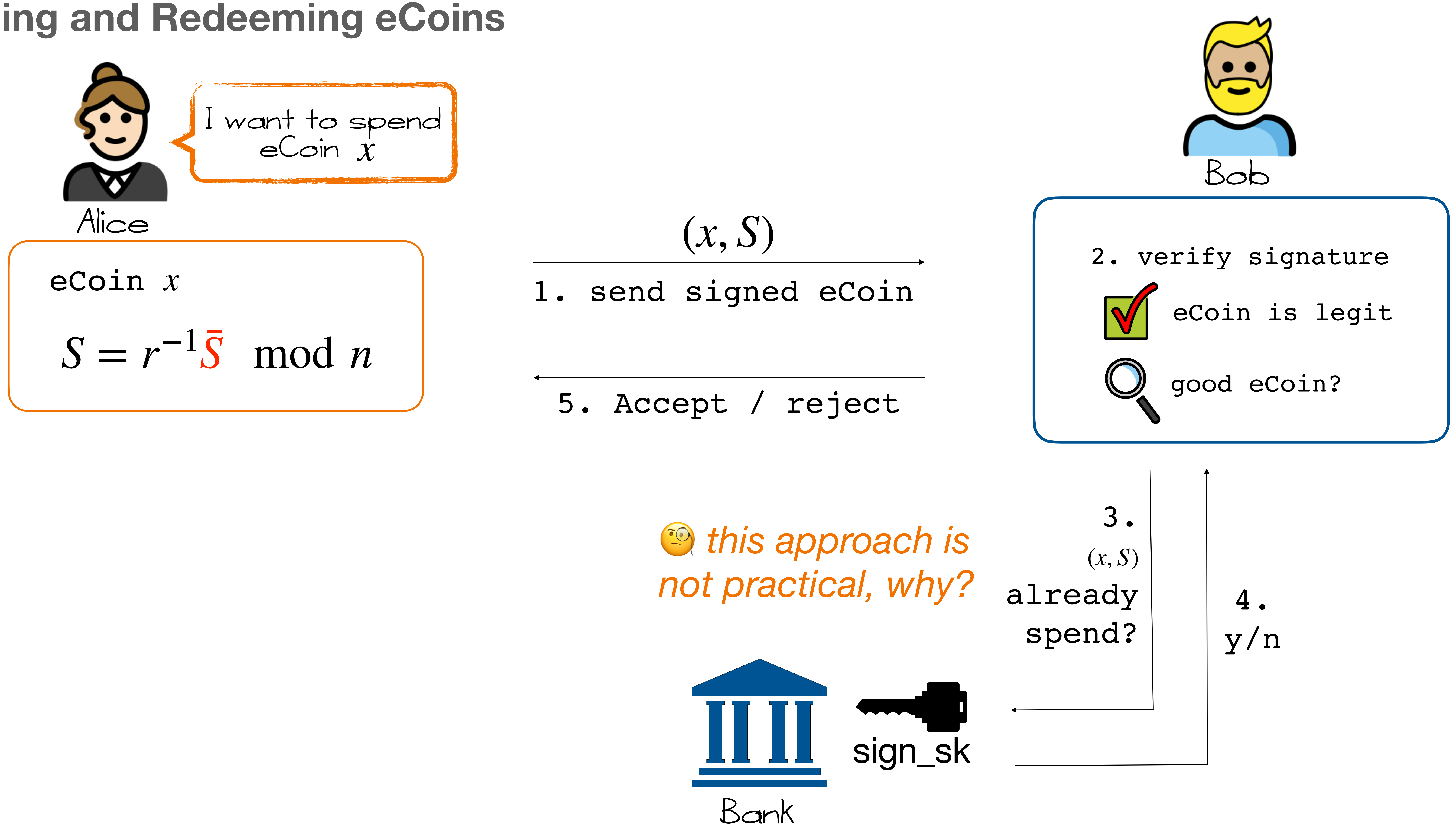
🧐 *is S a valid signature?*

S is a valid signature for x, and the Bank has never seen x or $H(x)$!

For simplicity assume all eCoins have value 1 (this does not mean x=1)

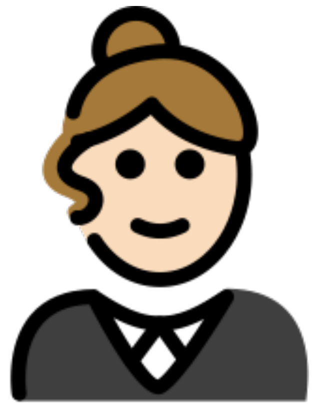
2&3 Prevent Double Spending and Keep eCoins Untraceable

Spending and Redeeming eCoins



A *Better* Untraceable eCash Protocol - Withdrawal

Aim: Alice looses her anonymity (ID_A gets disclosed) if and only if she tries to spend the same coin twice



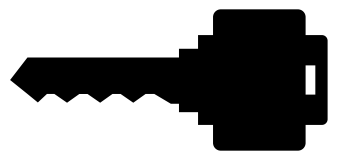
Alice

ID_A

cut and choose technique

ID_A

Bank



sign_sk

sk = d

Alice:

pick $2k$ 4-tuples of random numbers:

$$\{a_i, b_i, c_i, r_i\}_{i=1}^{2k}$$

let: $x_i = h(a_i, b_i)$

$$y_i = h(a_i \oplus ID_A, c_i)$$

compute:

$$B_i = r_i^e h(x_i, y_i) \mod n$$

$\xrightarrow{B_1, \ldots, B_{2k}}$ blinded values

Bank:

probabilistically verify that Alice has put her identity in every blinded value using the **Cut-and-Choose technique**

pick k random indexes:

$$I = \{i_1, i_2, \ldots, i_k\}$$

$\xleftarrow{I}$ indexes to check

Alice:

reveal the asked values

$\xrightarrow{\{a_i, b_i, c_i, r_i\}_{i \in I}}$ reveal values

Bank:

re-compute the B_i for $i \in I$ and check that they contain ID_A. If Alice did not cheat, sign the blind value on the unblinded indexes:

$$\bar{S} = (\Pi_{i \notin I} B_i)^d \mod n$$

$\xleftarrow{\bar{S}}$

Alice:

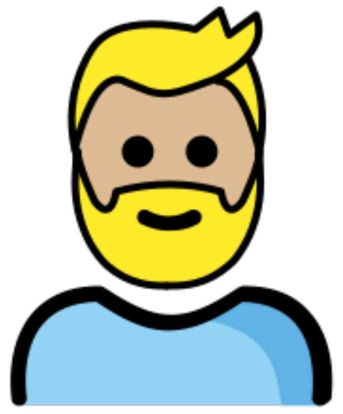
extract a signature S for the coin $x = \Pi_{i \notin I} h(x_i, y_i)$:

$$S = r^{-1} \bar{S} \mod n$$

A *Better* Untraceable eCash Protocol - Spending

I want to spend coin x

Alice



Bob

Alice:

coin x and signature S

disclose the values:

$$R_j = \begin{cases} (x_j, a_j \oplus ID_A, c_j), & \text{if } z_j = 0 \\ (a_j, b_j, y_j) & \text{if } z_j = 1 \end{cases}$$

Messages:

(x, S) → send signed coin

$Z = (z_1, z_2, \ldots, z_k)$ ← challenge Alice

$R = (R_1, R_2, \ldots, R_k)$ → response

Bob:

verify Bank's signature

is the coin good?

pick k rand bits $z_i \in \{0,1\}$

verify that Alice knows how to construct x

R 3. is good?

4.y/n

if Alice tries to spend the same coin twice then with high probability $Z \neq Z'$ so

$\exists j,\ z_j = 0,\ z'_j = 1$

$R_j \oplus R'_j = a_j \oplus ID_A \oplus a_j = \ ID_A$

$\Rightarrow$ Alice looses her anonymity to the Bank