Modern Cryptography
explained via Two-View Principle

Phan Duong Hieu

Telecom Paris, IPP
hieu.phan@telecom-paris.fr
https://www.di.ens.fr/~phan/

New Technologies and the Risks for Privacy

Privacy

- *Privacy:* “the right to be left alone”
- *Privacy protection* allows individuals to have control over how their personal information is collected and used

Big Data, Cloud computing

- Easy to collect and store user data
- Combined with powerful tools (e.g., machine learning)

$\rightarrow$ Attractive applications but Huge risk of mass surveillance, social credit systems.

Cryptography

Security of Data

- Integrity with hash function
- Confidentiality with encryption
- Authenticity with MAC, signature

New Technologies $\rightarrow$ Advanced cryptographic primitives

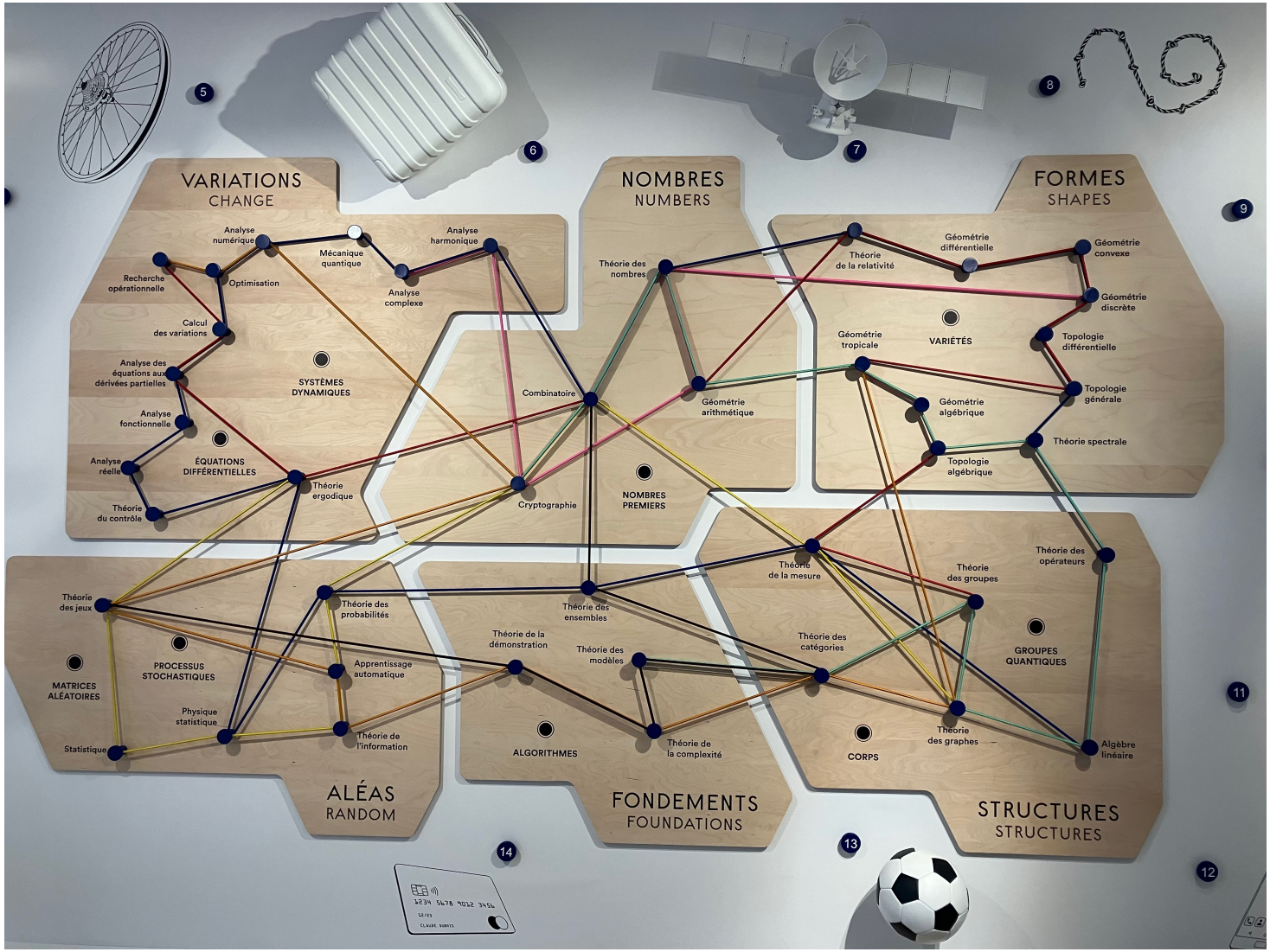
- Big Data, Cloud Computing $\rightarrow$ widespread real-life applications
- Privacy: protect personal information.
  - Security
  - Trust on Authorities

  $\rightarrow$ **Security of Computation on Untrusted Machines.**

Documentation:

`https://www.di.ens.fr/users/phan/cryptographie.html`

Cryptography in Museum of Mathematics



Some directions for the future:

- How to protect privacy in the AI era
- How to protect privacy against powerful adversaries (e.g., anamorphic encryption)
- How to implement the "Right to be Forgotten"
- How to use powerful tools (e.g., quantum machines) to protect data and privacy (e.g., key leasing)

Some directions for the future:

- How to protect privacy in the AI era
- How to protect privacy against powerful adversaries (e.g., anamorphic encryption)
- How to implement the "Right to be Forgotten"
- How to use powerful tools (e.g., quantum machines) to protect data and privacy (e.g., key leasing)

This course

- How new concepts were invented and their impact
- How security can be proven

Modern Cryptography from the **Two-View Principle**

Modern Cryptography from the **Two-View Principle**

Secret Communication: Sender vs. Receiver Views

- **Symmetric Encryption:** The same key is used for both encryption (locking) and decryption (unlocking).

Modern Cryptography from the **Two-View Principle**

Secret Communication: Sender vs. Receiver Views

- **Symmetric Encryption:** The same key is used for both encryption (locking) and decryption (unlocking).
- **Asymmetric Encryption:** Different keys are used for encryption and decryption $\rightarrow$ the public key is used for encryption, and the private key for decryption.

Public-key Encryption (Diffie-Helmann 1976)

- Encryption key could be published $\rightarrow$ encryption can be publicly computed.
- **RSA scheme**

$$(m^e)^{(e^{-1} \mod \phi(N))} = m \mod N, \text{ where } N = pq$$

Public-key Encryption (Diffie-Helmann 1976)

- Encryption key could be published $\rightarrow$ encryption can be publicly computed.
- **RSA scheme**

$$(m^e)^{(e^{-1} \mod \phi(N))} = m \mod N, \text{ where } N = pq$$

- **Elgamal scheme**

$$\frac{m(g^d)^r}{(g^r)^d} = m, \text{ where } g \text{ is a generator of a prime-order cyclic group}$$

Modern Cryptography

Beyond Encryption:

- Interactive proofs, zero-knowledge proofs, PCP
- Identification, Digital Signature
- Computation on Encrypted Data (Functional Encryption, FHE)
- Decentralized computation/ Verifiable computation (beyond data security)
- Multi-party computation (for doing any cryptographic task imaginable!)

Main Theoretical Question (Complexity)

Does Cryptography really exist?

Central Question of Complexity: P vs. NP from the **Two-View Principle**

Central Question of Complexity: P vs. NP from the **Two-View Principle**

On a Mathematical Problem: Solver vs. Verifier Views

- In mathematics: **Solving** a problem is often more difficult than **Verifying** a proposed solution.

Central Question of Complexity: P vs. NP from the **Two-View Principle**

On a Mathematical Problem: Solver vs. Verifier Views

- In mathematics: **Solving** a problem is often more difficult than **Verifying** a proposed solution.
- In computer science: Tackle this distinction $\rightarrow$ formal notion of efficiency and difficulty $\rightarrow$ Computational Models & Algorithms.

Centre question of Complexity: P vs. NP

- P: Problems for which solutions can be "efficiently" found
- NP: Problems for which solutions can be "efficiently" verified

Centre question of Complexity: P vs. NP

- P: Problems for which solutions can be "efficiently" found
- NP: Problems for which solutions can be "efficiently" verified

Efficiency

- Formal definition of algorithm (Turing machine)
- Church-Turing Thesis: everything that nature computes, can be emulated on a Turing machine
- Efficient algorithm: number of basic steps is bounded by a polynome on the size of the input

Centre question of Complexity: P vs. NP

- P: Problems for which solutions can be "efficiently" found
- NP: Problems for which solutions can be "efficiently" verified

Efficiency

- Formal definition of algorithm (Turing machine)
- Church-Turing Thesis: everything that nature computes, can be emulated on a Turing machine
- Efficient algorithm: number of basic steps is bounded by a polynome on the size of the input
- Example
  - P: multiplication, exponentiation modulo a prime number,...
  - NP: factorisation, discrete logarithm, 3-coloring problem, sodoku,...

Centre question of Complexity: P vs. NP

- P: Problems for which solutions can be "efficiently" found
- NP: Problems for which solutions can be "efficiently" verified

Centre question of Complexity: P vs. NP

- P: Problems for which solutions can be "efficiently" found
- NP: Problems for which solutions can be "efficiently" verified

Definition of an NP Language

A language $\mathcal{L}$ is an NP-language if there is a polynomial-time verifier V such that:

- **Completeness:** True theorems have (short) proofs.
  For all $x \in \mathcal{L}$, there is a polynomial($|x|$))-size witness (proof) $w \in \{0,1\}^*$ such that $V(x,w) = 1$.
- **Soundness:** False theorems have no short proofs.
  For all $x \notin \mathcal{L}$, there is no witness.
  i.e., for all polynomially long $w \in \{0,1\}^*$, $V(x,w) = 0$.

Cryptography and the P vs. NP problem

(Trapdoor) one-way functions

A function $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ is a (trapdoor) function if it is

- Efficiently computable: $f(x)$ is efficiently computable for any $x \in_R \{0,1\}^n$
- Hard to invert: for a random $x \in_R \{0,1\}^n$, given $y = f(x)$, it is hard to find a $\bar{x}$ such that $y = f(\bar{x})$

Cryptography and the P vs. NP problem

(Trapdoor) one-way functions

A function $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ is a (trapdoor) function if it is

- Efficiently computable: $f(x)$ is efficiently computable for any $x \in_R \{0,1\}^n$
- Hard to invert: for a random $x \in_R \{0,1\}^n$, given $y = f(x)$, it is hard to find a $\bar{x}$ such that $y = f(\bar{x})$: for all PPT adversary A:

$$\Pr_{x \in_R \{0,1\}^n}\left[A(1^n, f(x)) = x' \text{ et } f(x) = f(x')\right] \text{ is negligible.}$$

Cryptography and the P vs. NP problem

(Trapdoor) one-way functions

A function $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ is a (trapdoor) function if it is

- Efficiently computable: $f(x)$ is efficiently computable for any $x \in_R \{0,1\}^n$
- Hard to invert: for a random $x \in_R \{0,1\}^n$, given $y = f(x)$, it is hard to find a $\bar{x}$ such that $y = f(\bar{x})$: for all PPT adversary A:

$$\Pr_{x \in_R \{0,1\}^n}\left[A(1^n, f(x)) = x' \text{ et } f(x) = f(x')\right] \text{ is negligible.}$$

- Trapdoor: given a trapdoor, it is easy to invert the function f.

Cryptography and the P vs. NP problem

(Trapdoor) one-way functions

A function $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ is a (trapdoor) function if it is

- Efficiently computable: $f(x)$ is efficiently computable for any $x \in_R \{0,1\}^n$
- Hard to invert: for a random $x \in_R \{0,1\}^n$, given $y = f(x)$, it is hard to find a $\bar{x}$ such that $y = f(\bar{x})$: for all PPT adversary A:

$$\Pr_{x \in_R \{0,1\}^n}\left[A(1^n, f(x)) = x' \text{ et } f(x) = f(x')\right] \text{ is negligible.}$$

- Trapdoor: given a trapdoor, it is easy to invert the function f.

Necessary conditions for the existence of cryptography

- One-way function for secret-key cryptography

Cryptography and the P vs. NP problem

(Trapdoor) one-way functions

A function $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ is a (trapdoor) function if it is

- Efficiently computable: $f(x)$ is efficiently computable for any $x \in_R \{0,1\}^n$
- Hard to invert: for a random $x \in_R \{0,1\}^n$, given $y = f(x)$, it is hard to find a $\bar{x}$ such that $y = f(\bar{x})$: for all PPT adversary A:

$$\Pr_{x \in_R \{0,1\}^n}\left[A(1^n, f(x)) = x' \text{ et } f(x) = f(x')\right] \text{ is negligible.}$$

- Trapdoor: given a trapdoor, it is easy to invert the function f.

Necessary conditions for the existence of cryptography

- One-way function for secret-key cryptography
- Trapdoor one-way function for public-key cryptography

Cryptography and the P vs. NP problem

(Trapdoor) one-way functions

A function $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ is a (trapdoor) function if it is

- Efficiently computable: $f(x)$ is efficiently computable for any $x \in_R \{0,1\}^n$
- Hard to invert: for a random $x \in_R \{0,1\}^n$, given $y = f(x)$, it is hard to find a $\bar{x}$ such that $y = f(\bar{x})$: for all PPT adversary A:

$$\Pr_{x \in_R \{0,1\}^n}\left[A(1^n, f(x)) = x' \text{ et } f(x) = f(x')\right] \text{ is negligible.}$$

- Trapdoor: given a trapdoor, it is easy to invert the function f.

Necessary conditions for the existence of cryptography

- One-way function for secret-key cryptography
- Trapdoor one-way function for public-key cryptography

5 Worlds in Impagliazzo's view

W1-Algorithmica: P = NP

One could use the method of verifying the solution to automatically solve the problem!

5 Worlds in Impagliazzo's view

W1-Algorithmica: P = NP

One could use the method of verifying the solution to automatically solve the problem!

W2-Heuristica: NP problems are hard in the worst case but easy on average.

There exist hard instances of NP problem, but to find such hard instances is itself a hard problem.

5 Worlds in Impagliazzo's view

W1-Algorithmica: P = NP

One could use the method of verifying the solution to automatically solve the problem!

W2-Heuristica: NP problems are hard in the worst case but easy on average.

There exist hard instances of NP problem, but to find such hard instances is itself a hard problem.

W3-Pessiland: NP problems hard on average but no one-way functions exist

It's easy to generate many hard instances of NP-problems, but no way to generate hard instances where we know the solution.

5 Worlds in Impagliazzo's view (cont.)

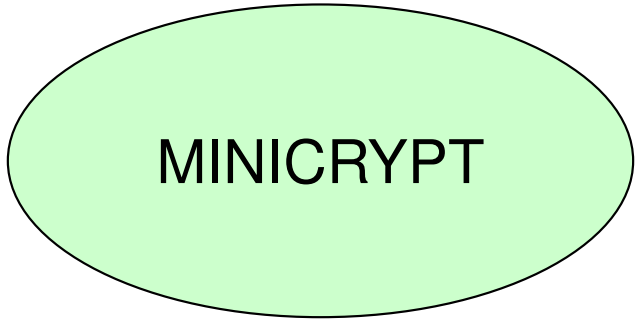
Minicrypt: One-way functions exist but public-key cryptography does not exist.

5 Worlds in Impagliazzo's view (cont.)

Minicrypt: One-way functions exist but public-key cryptography does not exist.

Cryptomania: Public-key cryptography is possible

It is possible for two parties to agree on a secret message using only public accessible channels


MINICRYPT

(Zero-knowledge) Interactive Proof: Idea

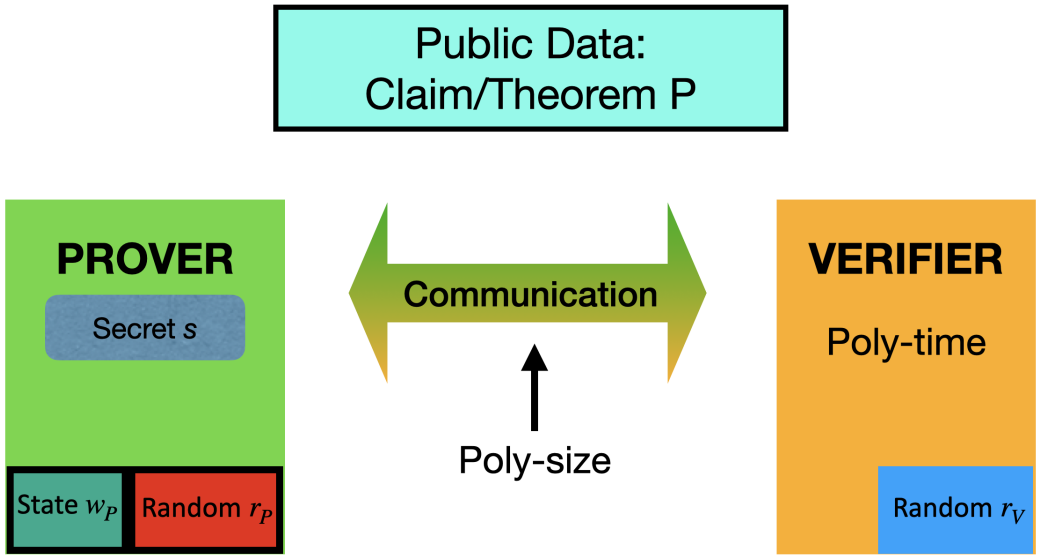
Interactive proofs [Goldwasser, Micali, Rackoff '85]

"A proof is whatever convinces me" (Shimon Even)

Zero-knowledge proofs: the verifier gets no information

- A toy example: Distinguishing the wines of Bordeaux and Côtes du Rhone
- ZKP for all NP problems (Goldreich-Micali-Wigderson '91)
  - Verifier is poly-time TM, Prover could be all powerful
  - Exemple: Graph non-isomorphism
  - Simulation (zero-knowledge)
- Zero-knowledge proof of knowledge.
  - Verifier is poly-time TM, Prover is often poly-time TM as well
  - Simulation (zero-knowledge) + Extraction (proof of knowledge)

Interactive Proofs



$\mathcal{L}$ is an **IP-language** if there is a **probabilistic poly-time** verifier V:

- **Completeness:** If $x \in \mathcal{L}$,

$$\Pr[(P, V)(x) = \text{accept}] = 1.$$

- **Soundness:** If $x \notin \mathcal{L}$, for every P^*,

$$\Pr[(P^*, V)(x) = \text{accept}] \text{ is negligible.}$$

Security
from the **Two-View Principle**

Security
from the **Two-View Principle**

Communication: Insider vs. Outsider Views

- Security is often established by showing that communication generated by an **insider** (who knows the secret) can be **simulated** by an **outsider** (who does not know the secret).

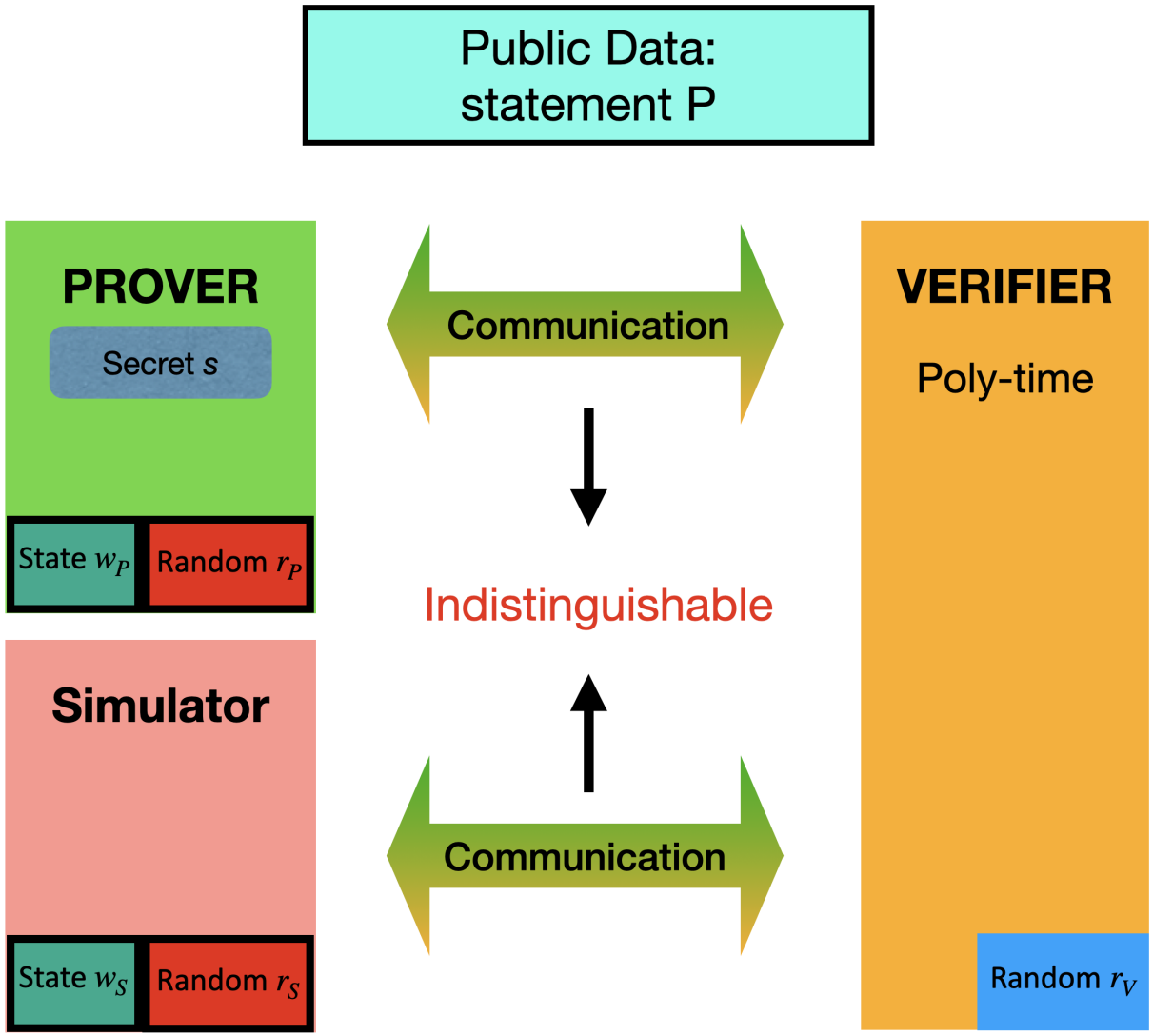
Security
from the **Two-View Principle**

Communication: Insider vs. Outsider Views

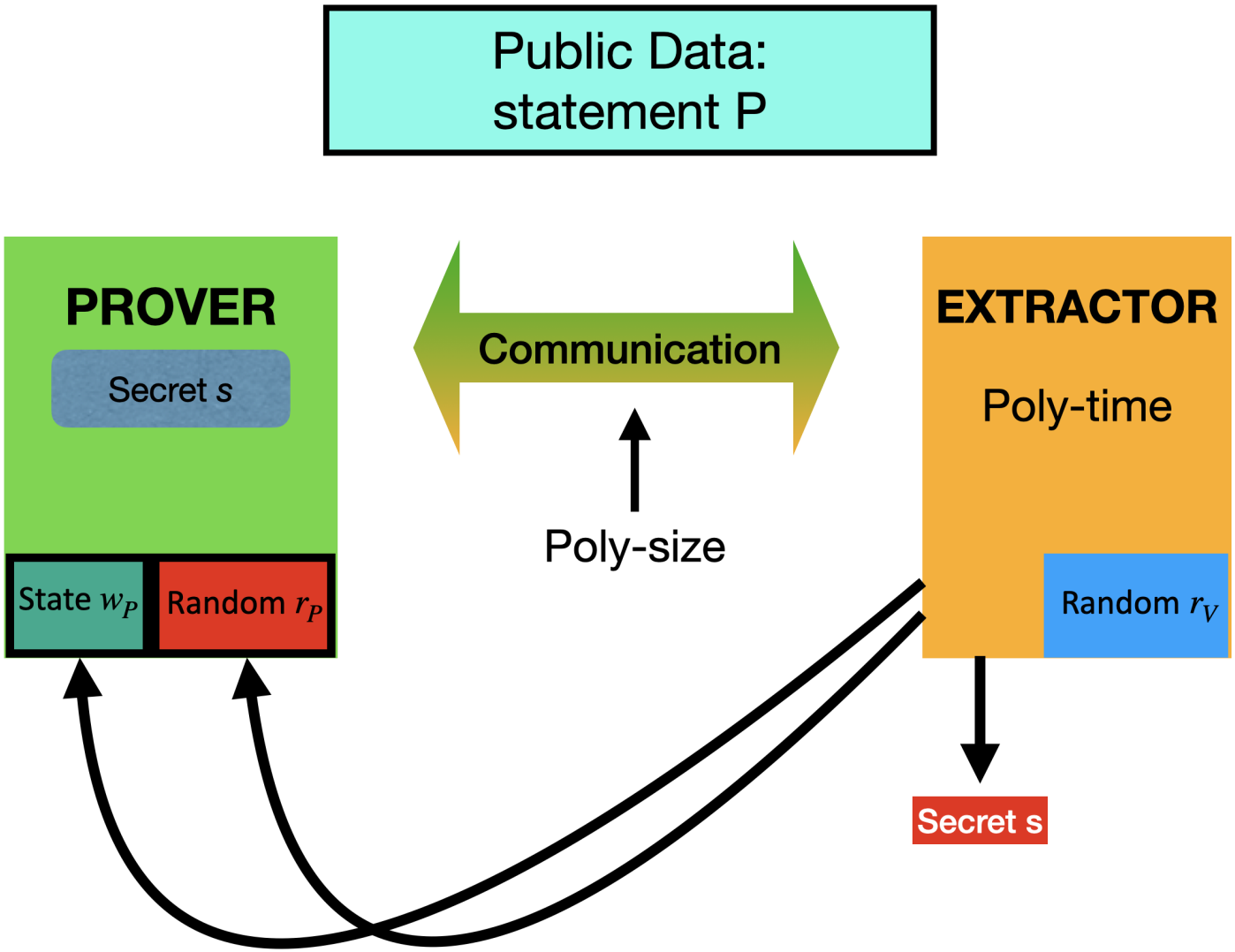
- Security is often established by showing that communication generated by an **insider** (who knows the secret) can be **simulated** by an **outsider** (who does not know the secret).

$\rightarrow$ This implies that the communication does not leak the secret.

Zero-knowledge Proof: Simulator



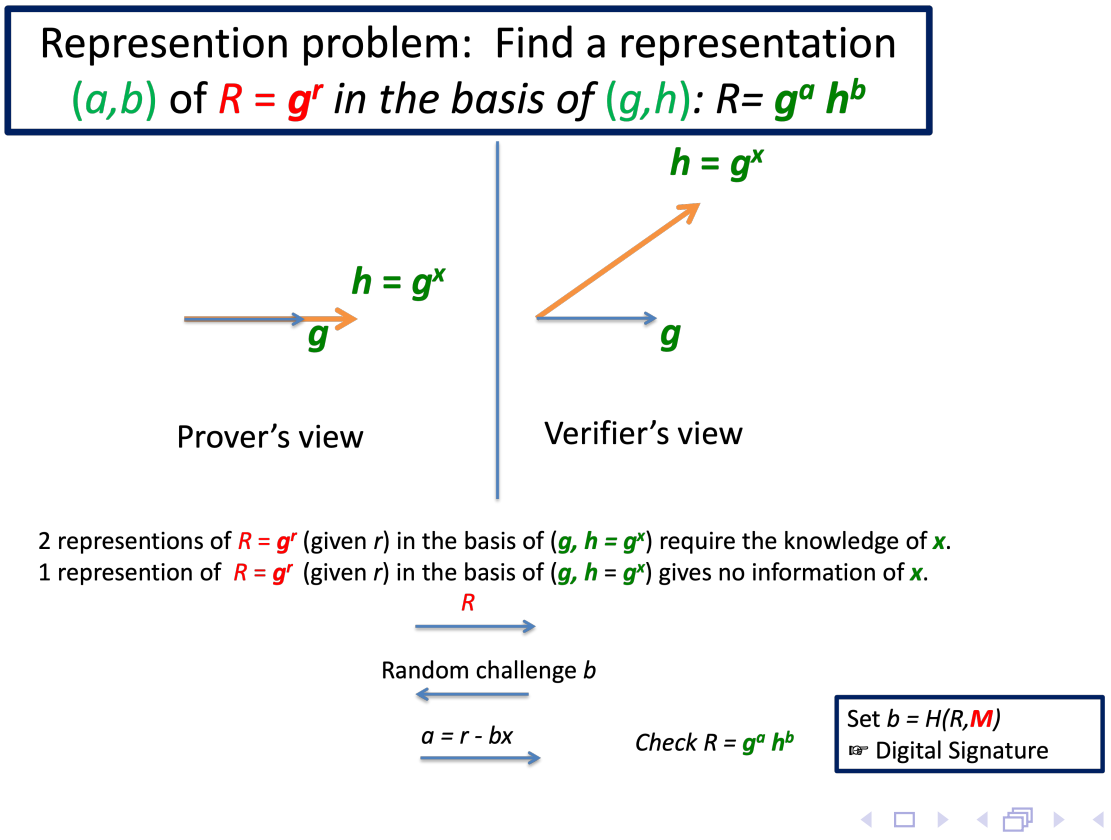
Zero-knowledge Proof of Knowledge: Extractor



Zero-knowledge Proof of Knowledge on DL

Zero-knowledge

DL Assumption: $G =< g >$, given $h = g^x$, it is hard to compute x.
ZKP: Given g and $h = g^x$, I can convince you that I know x without revealing it.



Zero-knowledge Proof for DDH

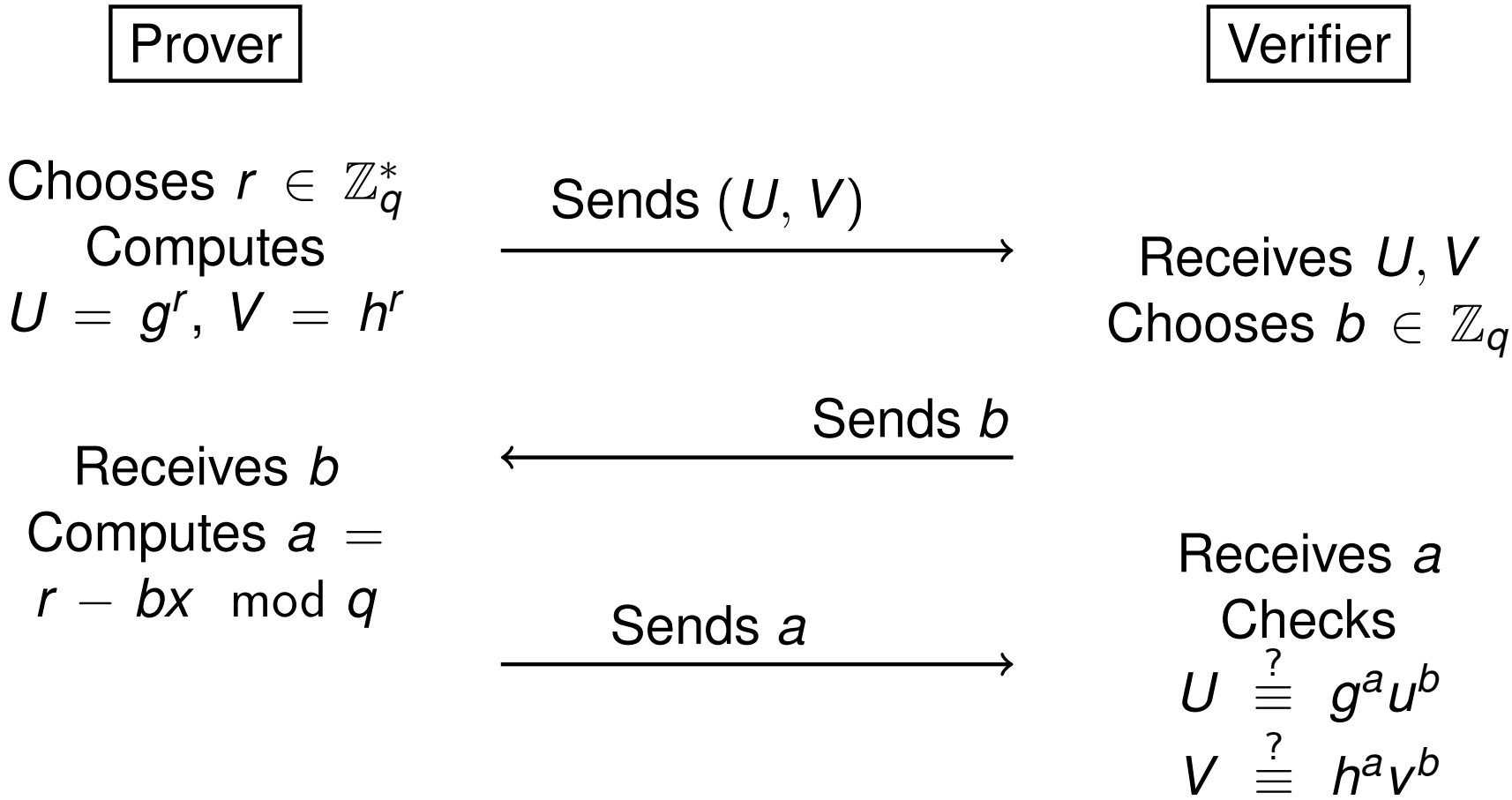
In a group $\mathbb{G} = \langle g \rangle$ of prime order q, the **DDH**(g, h) assumption states it is hard to distinguish

$$\mathcal{L} = (u = g^x, v = h^x) \quad \text{from} \quad \mathcal{G}^2 = (u = g^x, v = h^y)$$

$\mathcal{P}$ knows x, such that $(u = g^x, v = h^x) \in \mathcal{L}$ and wants to prove it to $\mathcal{V}$

Zero-Knowledge Proof for DDH

Prover knows x such that $(u = g^x, v = h^x) \in \mathcal{L}$



Completeness and Soundness

Completeness

- **Definition (recall):** If the prover knows the secret x such that $(u = g^x, v = h^x) \in \mathcal{L}$ and follows the protocol correctly, the verifier will always accept the proof.
- **Indeed:** If $\mathcal{P}$ knows x, both checks will hold and the verifier will accept the proof.

Soundness

- **Definition (recall):** If the statement is false (i.e., $(u = g^x, v = h^{x'}) \notin \mathcal{L}, x \neq x'$), no prover can convince the verifier except with negligible probability.
- **Indeed:** The prover can consistently compute a that satisfies both conditions: $U = g^r = g^a u^b$ and $V = h^{r'} = h^a v^b$, which implies $r - r' = b(x - x')$, for random b with probability $1/q$, which is negligible:

Zero-knowledge Proof: Simulator

- **Definition:** The verifier learns nothing beyond the validity of the statement.
- **Simulator Construction:**
  1. Simulator selects a random $b \in \mathbb{Z}_q$ and $a \in \mathbb{Z}_q$.
  2. Computes $U = g^a u^b$ and $V = h^a v^b$.
  3. Output (a, b, U, V).
- **Indistinguishability:**
  - Verifier cannot distinguish between real and simulated interactions.
  - Transcripts are statistically indistinguishable.

Extractor

- **Extractor Role:** To show that the prover knows x, an extractor can derive x from multiple valid responses.
- **Extractor Construction:**
  1. For fixed (U, V), two valid answers s and s' satisfy:

$$\begin{aligned} g^a u^b &= U = g^{a'} u^{b'} \\ h^a v^b &= V = h^{a'} v^{b'} \end{aligned}$$

  2. If one sets $x^\star$ such that:

$$x^\star = (a - a')(b' - b)^{-1} \mod q$$

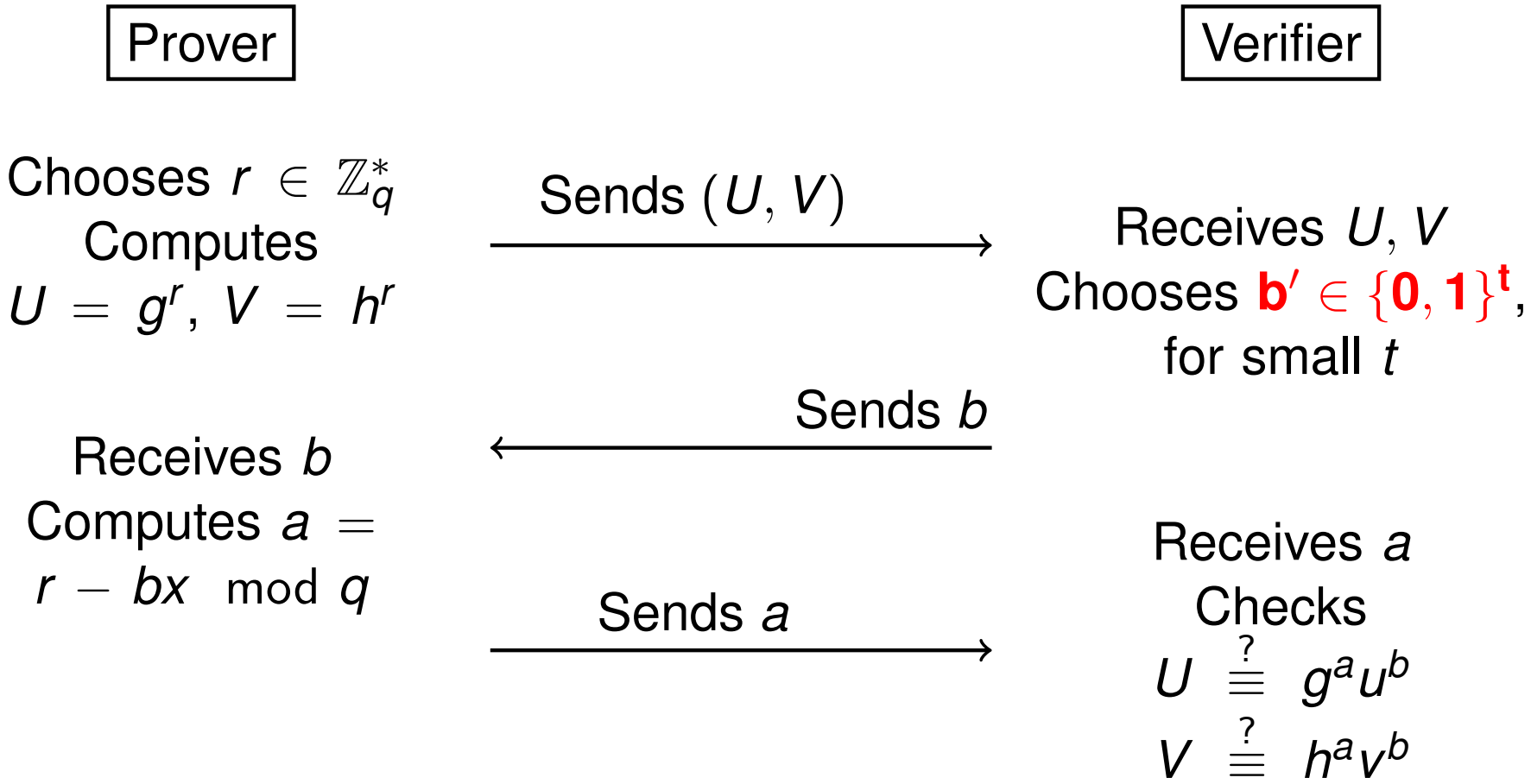
  3. This implies:

$$u = g^{x^\star} \quad \text{and} \quad v = h^{x^\star}$$

- **Conclusion:** The existence of the extractor confirms that the prover knows x.

Zero-Knowledge Proof for DDH: Malicious Verifier

Prover knows x such that $(u = g^x, v = h^x) \in \mathcal{L}$



Zero-Knowledge Proof: Simulator

- **Definition:** The verifier learns nothing beyond the validity of the statement.
- **Simulator Construction:**
  1. Simulator selects a random $\mathbf{b'} \in \{\mathbf{0}, \mathbf{1}\}^{\mathbf{t}}$ (recall that t is small so that $2^t \in poly(\log q)$) and $a' \in \mathbb{Z}_q$.
  2. Computes $U = g^{a'} u^{b'}$ and $V = h^{a'} v^{b'}$.
  3. Sends (U, V) to the verifier.
  4. Verifier sends back a challenge $b \in \{0, 1\}^t$.
  5. If $b = b'$, simulator sets $a = a'$ and outputs (a, b, U, V)
  6. If $b \neq b'$, simulator restarts.
- **Indistinguishability:**
  - Verifier cannot distinguish between real and simulated interactions.
  - Transcripts are statistically indistinguishable.
- **Conclusion:** The proof is zero-knowledge as the verifier learns nothing about x.

Minicrypt: Commitment

- Alice **commits** herself to some message m by giving Bob: $c = \text{Commit}(m, r)$, for a ramdom r.
- Bob should not learn anything about m given the commitment c.
- Alice can **open** the commitment by giving (m, r) to Bob to convince him that m was the value she committed herself to.

Two properties:

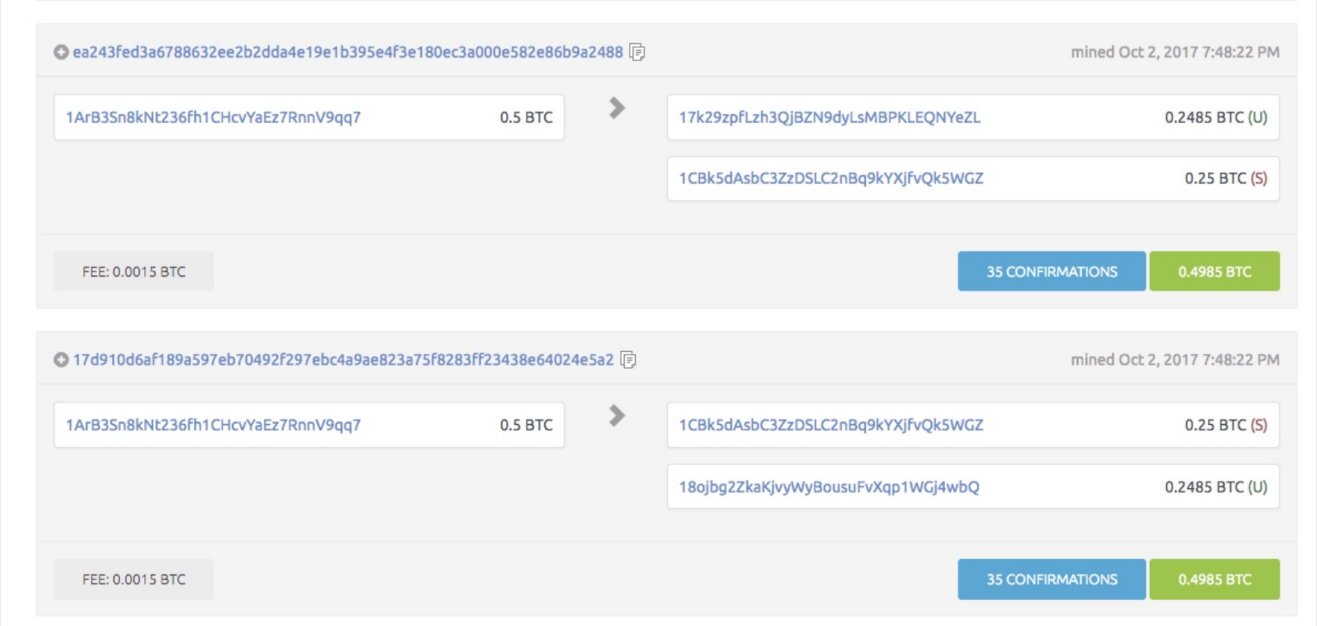
- **Hiding** Commitment c hides information on m
- **Binding** Alice cannot open c to $(m', r') \neq (m, r)$

Example & Application

- Pederson's construction
- General construction from one-way function
- $\rightarrow$ In Minicrypt: ZKP for all NP problem (on ZKP for G3C)

ZKP in Practice: Privacy in Blockchain

A Bitcoin transaction



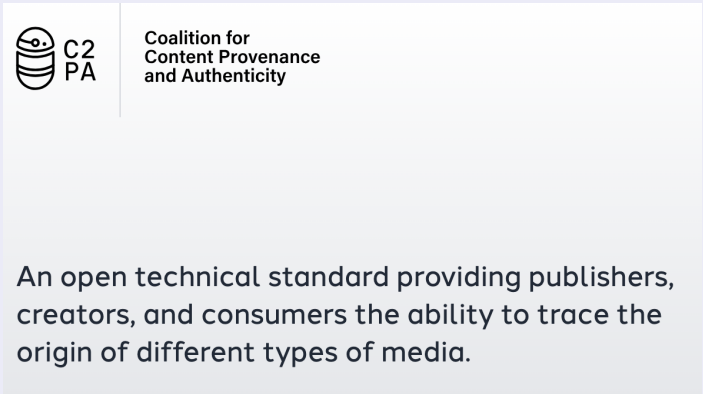
Privacy

- What is the problem with privacy in bitcoin?
- How we can use ZKP to solve this? $\rightarrow$ zkSNARKS.

Signatures/Commitment in Practice

(beyond classical examples)

C2PA and the need of Short Polynomial Commitment



Polynomial Commitment

Given a polynomial $P(x) = \sum_{i=0}^{n-1} a_i x^i$. We want the sender to commit P in such a way that it can prove to the receiver that (u, v) satisfies: $P(u) = v$.

- linear-size commitment: exercice
- constant-size commitment: KZG10, using pairings

KZG10 Polynomial Commitment: Setup

- **Setup:**
  1. Select a prime field $\mathbb{F}_p$ and a generator g of a group $\mathbb{G}$ of prime order p.
  2. Choose a random $s \in \mathbb{F}_p$.
  3. Compute $\{g_0 = g, g_1 = g^s, g_2 = g^{s^2}, \ldots, g_d = g^{s^d}\}$ for a polynomial of degree d.
  4. Publish the setup parameters $\mathsf{PP} = \{g_0, g_1, g_2, \ldots, g_d\}$.
- **Trusted Setup Assumption:** The trusted setup randomly generates s, compute $\{g_0, g_1, g_2, \ldots, g_d\}$, then erase s.

KZG10 Polynomial Commitment: Commitment

- **Polynomial:** $P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_d x^d$
- **Commitment:**

$$\begin{aligned} C &= g_0^{a_0} \cdot g_1^{a_1} \cdot g_2^{a_2} \cdot \cdots \cdot g_d^{a_d} \\ &= g^{a_0} \cdot (g^s)^{a_1} \cdot (g^{s^2})^{a_2} \cdot \cdots \cdot (g^{s^d})^{a_d} \\ &= g^{P(s)} \end{aligned}$$

- **Result:** The commitment C is a single group element in $\mathbb{G}$.

KZG10 Polynomial Commitment: Opening

- **Opening:**
  - To open the commitment at point $x = u$, compute the evaluation $v = P(u)$.
  - u is a root of $P(x) - v$.
  - We can write thus $P(x) - v = (x - u)Q(x)$ and can compute $Q(x)$.
  - Generate proof $\pi = g^{Q(s)}$.
- **Send to Verifier:** $\{u, v, \pi\}$

KZG10 Polynomial Commitment: Verification

- **Verification:**
  1. Verifier receives $\{u, v, \pi\}$ and the commitment C.
  2. IDEA: Check at the random point s if $P(s) - v = (s - u)Q(s)$
  3. This check can only performed with a pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$
  4. Compute:

$$e(g_1 g^{-u}, \pi) \stackrel{?}{=} e(g, Cg^{-v})$$

- **Result:** If the equality holds, the proof is valid and the polynomial evaluates to $v = P(u)$ at point u.

Minicrypt: Digital Signatures (Idea)

If one-way functions exist, then every NP problem has a zero-knowledge proof. [Goldreich, Micali, Wigderson 91]

From zero-knowledge proof to digital signature (Schnorr scheme)

Given g and $y = g^x$, sign on the message m with the secret key x

- I take a random r and send to you g^r
- k is set to be $H(g^r, m)$ (H is modeled as a random oracle)
- I finally send to you the signature $(m, g^r, t = r - kx)$.
- Verification: checking whether $g^r = g^t y^{H(g^r, m)}$

Minicrypt: Digital Signatures

In Random Oracle Model

If one-way functions exist, then one can construct digital signature.

Minicrypt

- Zero-knowledge proofs, Identification, Digital Signature inspire from the notion of PKE.
- However, even if PKE dies one day, the above primitives would still be alive!

Digital Signatures: Formal Treatment

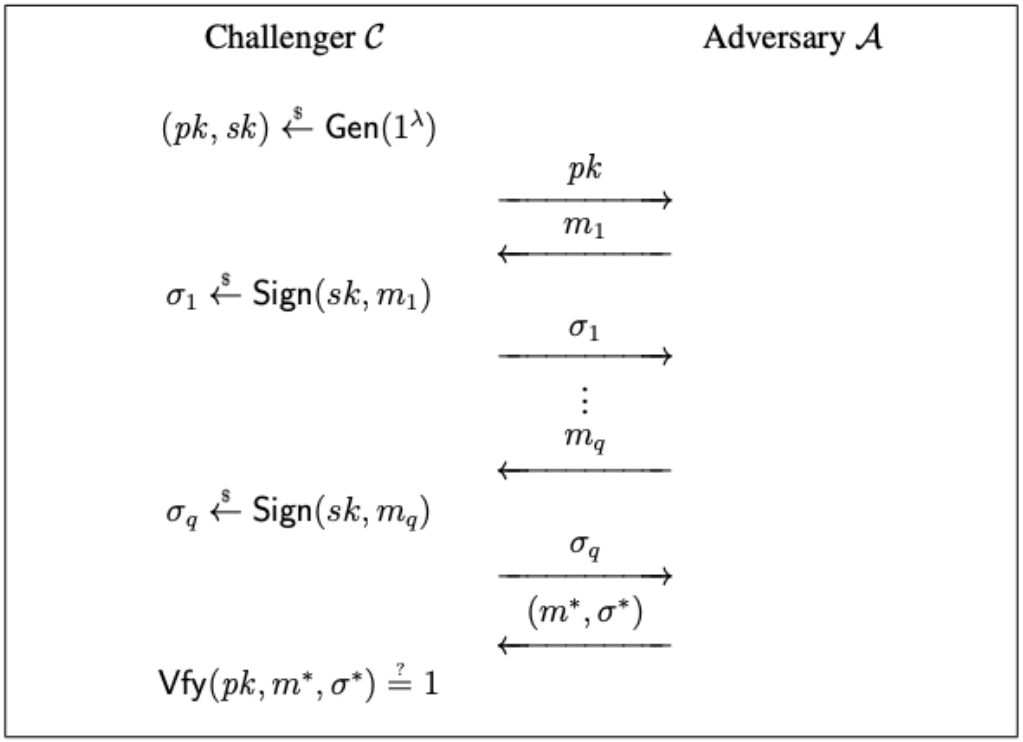
A signature scheme $S = (G, S, V)$:

- **Key Generation**: $G(1^\lambda) \to (pk, sk)$ is a probabilistic algorithm that takes a security parameter λ and outputs a secret signing key sk and a public verification key pk.
- **Signing**: $S(sk, m) \to \sigma$ is a probabilistic algorithm that outputs a signature σ.
- **Verification**: $V(pk, m, \sigma)$ outputs either accept (1) or reject (0).

We require that a signature generated by S is always accepted by V:

$$\Pr[V(pk, m, S(sk, m)) = \text{accept}] = 1$$

Digital Signatures: attack model (EUF-CMA)



Existential unforgeability under adaptive chosen message attacks

$$Adv(\mathcal{A}) = \Pr[Vfy(pk, m^\star, \sigma^\star) = 1]$$

The scheme is EUF-CMA secure si $\forall \mathcal{A}, Adv(\mathcal{A})$ is negligible.

Lamport's One-time Signatures from OWF f

- $Gen(1^\lambda) \to (pk, sk)$:

$$sk = \begin{pmatrix} x_{1,0} & x_{2,0} & \cdots & x_{\ell,0} \\ x_{1,1} & x_{2,1} & \cdots & x_{\ell,1} \end{pmatrix}$$

$$pk = \begin{pmatrix} y_{1,0} & y_{2,0} & \cdots & y_{\ell,0} \\ y_{1,1} & y_{2,1} & \cdots & y_{\ell,1} \end{pmatrix}$$

  where $x_{i,b} \in \{0,1\}^n, y_{i,b} = f(x_{i,b})$

- $Sign(sk, m = m_1 m_2 \ldots m_\ell \in \{0,1\}^\ell) \to \sigma$

$$\sigma = x_{1,m_1} x_{2,m_2} \ldots x_{\ell,m_\ell}$$

- $Vfy(pk, m, \sigma)$ check if $y_{i,m_i} = f(\sigma_i = x_{i,m_i}), \forall i = 1 \ldots \ell$

Theorem

If f is one-way, then the one-time signature is EUF-CMA.

Complete proof: OWF $\rightarrow$ Digital Signature

Complete proof: OWF $\rightarrow$ Digital Signature

- 1-time signature $\rightarrow$ Stateful 2-time signature $\rightarrow$ Stateful poly-time signature
- Stateful *to* Stateless with PRF
- Sign on a long message $\rightarrow$ short message by using a hash function.

Exercice:

Given:

- a collision resistant hash function $H : \{0,1\}^* \rightarrow H : \{0,1\}^n$
- a EUF-CMA singature on message of n bits

Construct another EUF-CMA singature that can sign on messages of abitrary size.

FDH - RSA

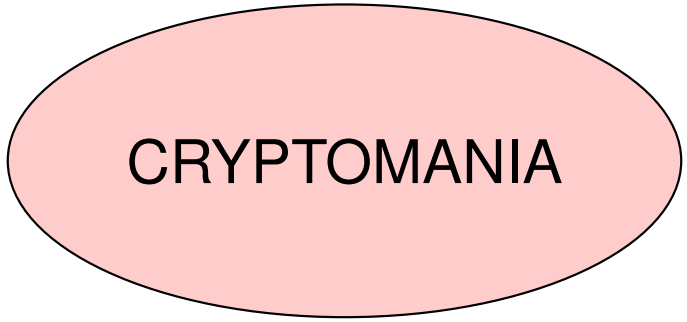
- $Gen(1^\lambda) \rightarrow (sk = d, pk = (N, e))$ as in RSA
- $Sign(sk, m) \rightarrow \sigma = H(m)^d$, where H is a **random oracle**.
- $Verfy(pk, m, \sigma)$ accept iff $\sigma^e = H(m)$

Security of FDH - RSA

If RSA problem is hard then FDH - RSA is EUF-CMA secure.
Proof: reading exercice.

Signature Schemes in Practice

		Key exchange			Signatures		
	Hosts	**RSA**	**DH**	**ECDH**	**RSA**	**DSA**	**ECDSA**
HTTPS	39M	39%	10%	51%	99%	≈ 0	1%
SSH	17M	≈ 0	52%	48%	93%	7%	0.3%
IKEv1	1.1M	-	97%	3%	-	-	-
IKEv2	1.2M	-	98%	2%	-	-	-


CRYPTOMANIA

Provable security: sufficient conditions for security

What we discussed

If factorization or DL problems are easy, then we can attack crypto systems (RSA, ElGamal) that based on these problems

Question

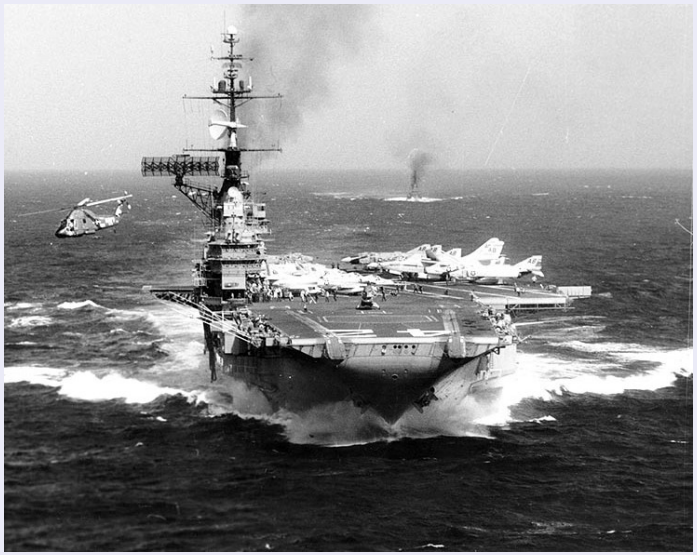
Suppose that factorization and DL problems are hard. Could we prove the security for proposed crypto systems?

One wayness is enough?

$$E'(m1||m_2) := E(m_1)||m_2$$

- If E is one-way, then E' is also one-way
- But the security of E' is clearly not enough: at least half the message leaks!

In many situation, one bit (attack or not) is important...



Semantic security [Goldwasser-Micali '82]

Perfect Security vs. Semantic security

- Perfect security: the distribution of the ciphertext is perfectly independent of the plaintext
- Semantic security (computational version of perfect security): the distribution of the ciphertext is computationally independent of the plaintext

Semantic security [Goldwasser-Micali '82]

Perfect Security vs. Semantic security

- Perfect security: the distribution of the ciphertext is perfectly independent of the plaintext
- Semantic security (computational version of perfect security): the distribution of the ciphertext is computationally independent of the plaintext

Semantic Security - IND

- Semantic Security is equivalent to the notion of Indistinguishability (IND): No adversary (modeled by a poly-time Turing machine) can distinguish a ciphertext of m_0 from a ciphertext of m_1.

IND Security Notion

- **IND:**
  - **Security Game:**
    1. The adversary $\mathcal{A}$ receive pk and chooses two plaintexts m_0 and m_1.
    2. A random bit $b \in \{0, 1\}$ is selected, and the challenger encrypts m_b to get the ciphertext $c = \mathsf{Enc}(pk, m_b)$.
    3. The ciphertext c is given to $\mathcal{A}$.
    4. $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$.
  - **Advantage:**
$$\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CPA}} = \left| \Pr[b' = b] - \frac{1}{2} \right|$$
  - **IND-CPA Security:**
$$\forall \text{polynomial-time } \mathcal{A}, \mathsf{Adv}_{\mathcal{A}}^{\text{IND-CPA}} \text{ is negligible}$$

Security of RSA & ElGamal PKE

Recall:

- k_e could be published $\rightarrow$ encryption can be publicly computed.
- **RSA scheme**

$$(m^e)^{(e^{-1} \mod \phi(N))} = m \mod N, \text{ where } N = pq$$

Security of RSA & ElGamal PKE

Recall:

- k_e could be published $\rightarrow$ encryption can be publicly computed.
- **RSA scheme**

$$(m^e)^{(e^{-1} \mod \phi(N))} = m \mod N, \text{ where } N = pq$$

- **ElGamal scheme**

$$\frac{m(g^d)^r}{(g^r)^d} = m, \text{ where } g \text{ is a generator of a cyclic group}$$

Exercices

- Is RSA IND? Is ElGamal IND?

Security of RSA & ElGamal PKE

Recall:

- k_e could be published $\rightarrow$ encryption can be publicly computed.
- **RSA scheme**

$$(m^e)^{(e^{-1} \mod \phi(N))} = m \mod N, \text{ where } N = pq$$

- **ElGamal scheme**

$$\frac{m(g^d)^r}{(g^r)^d} = m, \text{ where } g \text{ is a generator of a cyclic group}$$

Exercices

- Is RSA IND? Is ElGamal IND?
- For public-key encryption: Probabilistic encryption is required!
- For secret-key encryption: deterministic encryption could be semantically secure [Phan-Pointcheval '04]

Semantic security/IND is enough?

ElGamal Encryption

- Elgamal encryption can be proven to be IND, based on Decisional Diffie-Hellman assumption (given g^a, g^b, it is hard to distinguish between g^{ab} and a random element g^z).
- Elgamal encryption is homomorphic: $E(m_1 m_2) = E(m_1)E(m_2)$

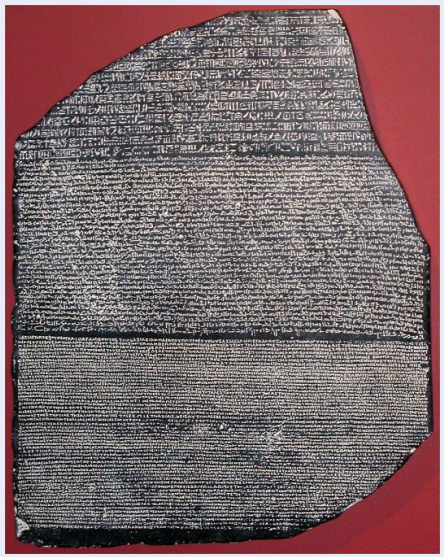
Private Auctions

The bids are encrypted. The authority then opens all the encrypted bids and the highest bid wins

- IND guarantees privacy of the bids
- Malleability: from $c = E(pk, b)$, without knowing b, one can generate $c' = E(pk, 2b)$: an unknown higher bid!
- Should consider adversaries with some more information.

Adversaries with additional information

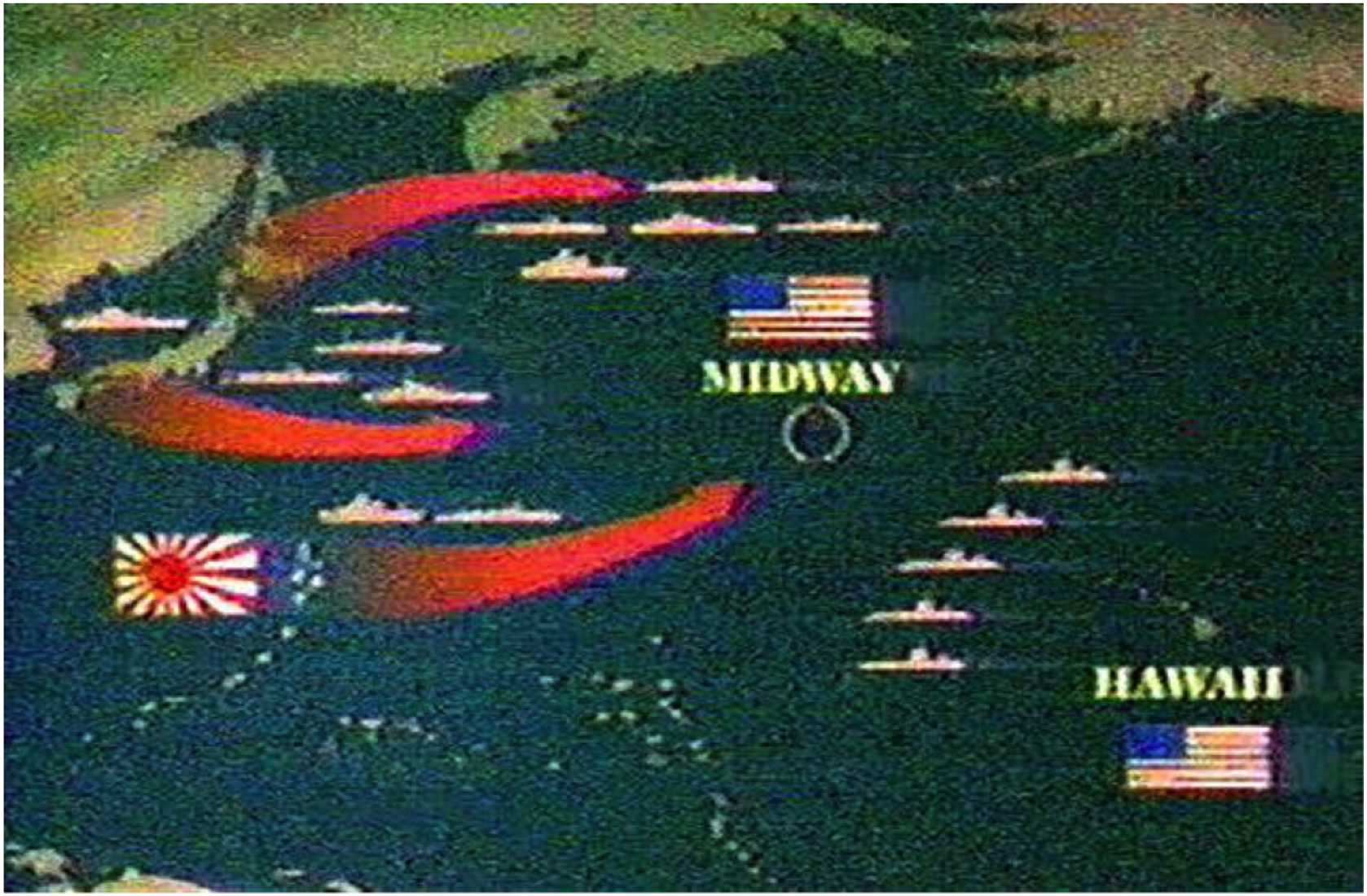
Rosetta Stone: A key element to decode Ancient Egyptian hieroglyphs



Chosen plaintext attacks (CPA)

The adversary can have access to encryption oracle (this only makes sense for symmetric encryption)

Interactive Adversaries: CCA attacks



IND-CCA Security Notion in Encryption

- **IND-CCA Security Game:**
  1. The adversary $\mathcal{A}$ is given pk and also given access to an oracle that decrypts ciphertexts.
  2. $\mathcal{A}$ chooses two plaintexts m_0 and m_1.
  3. A random bit $b \in \{0,1\}$ is selected, and the challenger encrypts m_b to get the challenge ciphertext $c^* = \mathsf{Enc}(pk, m_b)$.
  4. The ciphertext c^* is given to $\mathcal{A}$.
  5. $\mathcal{A}$ continues to have access to the decryption oracle, except for the challenge ciphertext (*i.e.,* cannot query c^*). (*)
  6. $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.
- **Advantage:**

$$\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CCA}} = \left| \Pr[b' = b] - \frac{1}{2} \right|$$

- **IND-CCA Security:**

$$\forall \text{polynomial-time } \mathcal{A}, \mathsf{Adv}_{\mathcal{A}}^{\text{IND-CCA}} \text{ is negligible}$$

CCA1: CCA without access to the decryption oracle in the second phase (*)

Chosen plaintext and chosen ciphertext attacks

IND-CCA Security

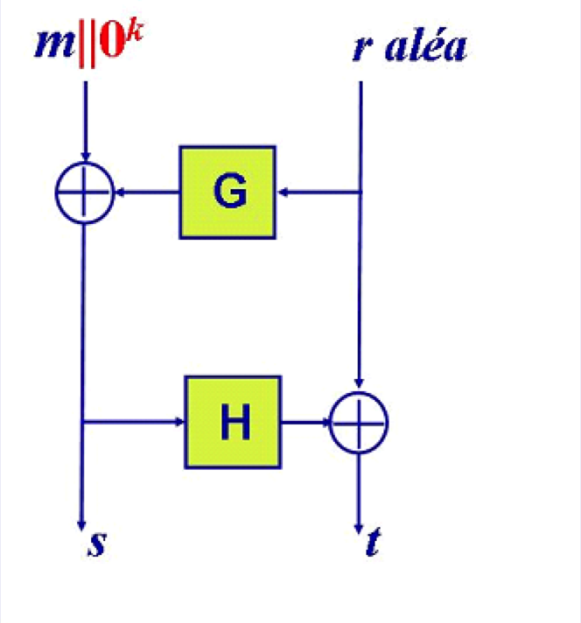
- IND-CCA also implies non-malleability (NM-CCA)
- This is the standard notion for public-key encryption
- Exercice: Is ElGamal IND-CCA?

Major problem in cryptography

Construction of IND-CCA encryption schemes.

OAEP (Bellare-Rogaway94)

Random oracle model



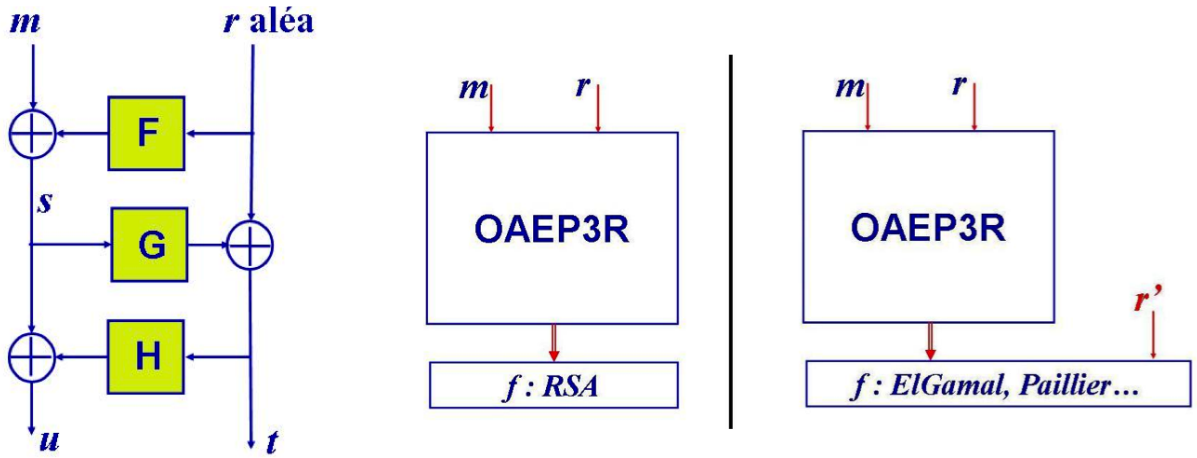
- It is believed that *f*-OAEP is IND-CCA for any trapdoor one-way permutation.
- In 2000, Shoup presented an attack on a very special trapdoor one-way permutation.

RSA-OAEP

RSA-OAEP is proven IND-CCA secure [Fujisaki-Okamoto-Pointcheval-Stern01]

- If f is partially one-way, then f-OAEP is secure
- RSA is partially one-way

3-round OAEP (among others varieties of OAEP)

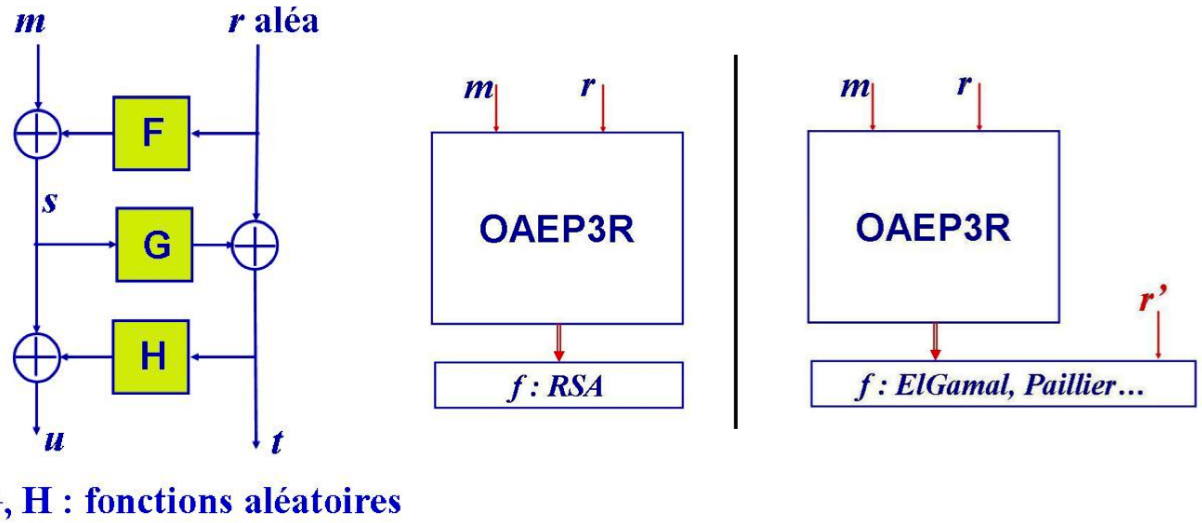


F, G, H : fonctions aléatoires

Advantages

- f does not need to be partially one-way
- f could also be one-way function (such as Elgamal, Paillier encryptions...)

3-round OAEP (among others varieties of OAEP)



Advantages

- f does not need to be partially one-way
- f could also be one-way function (such as Elgamal, Paillier encryptions...)

Current state

Many solutions in the standard model (without random oracle) but the practical implementations mostly rely on RSA-OAEP.

Security Proofs: Game Sequence technique

Proof of IND-CPA of ElGamal scheme, under DDH assumption

Let $\mathbb{G} = \langle g \rangle$ with generator g of order $|\mathbb{G}| = q$ where q is a prime.
Public key $pk = (g, h = g^x)$ and secret key $sk = x$.
Encryption:Enc$(pk, m) = (g^r, h^r \cdot m)$ where $r \leftarrow \mathbb{Z}_q$.

- **Game 0:** Real IND-CPA game, challenge ciphertext is $(g^r, h^r \cdot m_b)$
- **Game 1:** Replace (g, h, g^r, h^r) by $(g, h, g^r, h^{r'})$, for random r, r'
  The adversary cannot distinguish Game 0 and Game 1, otherwise we can solve DDH
- In Game 1: the adversary has no information about m_b.

Security Proofs: IND-CCA

Idea: Embed a ZK proof of knowledge for a DDH tuple in the ciphertext.

- Let $\mathbb{G} = \langle g \rangle$ with generator g of order $|\mathbb{G}| = q$ where q is a prime.
- Verifier chooses $\alpha, x_1, x_2 \leftarrow \mathbb{Z}_q$ and sets $g_1 = g, g_2 = g^\alpha, c = g_1^{x_1} g_2^{x_2}$ and sends g_1, g_2, c to prover.
- Prover chooses $r \leftarrow \mathbb{Z}_q$, sets $u_1 = g_1^r, u_2 = g_2^r$ and $v = c^r$
- Verifier checks whether $v = u_1^{x_1} u_2^{x_2}$.

Lemma

It is hard for the prover to return $u_1 = g_1^{r_1}, u_2 = g_2^{r_2}$ and v such that $r_1 \neq r_2$ and pass the check of the verifier:

$$v = u_1^{x_1} u_2^{x_2}$$

Proof: exercice

Security Proofs: IND-CCA

Idea: Embed a ZK proof of knowedge in the ciphertext.

- Let $\mathbb{G} = \langle g \rangle$ with generator g of order $|\mathbb{G}| = q$ where q is a prime.
- Verifier chooses $\alpha, x_1, x_2 \leftarrow \mathbb{Z}_q$ and sets $g_1 = g, g_2 = g^\alpha, c = g_1^{x_1} g_2^{x_2}$ and sends g_1, g_2, c to prover.
- Prover chooses $r \leftarrow \mathbb{Z}_q$, sets $u_1 = g_1^r, u_2 = g_2^r$ and $v = c^r$
- Verifier checks whether $v = u_1^{x_1} u_2^{x_2}$.

Cramer-Shoup Lite scheme (IND-CCA1)

Public key $pk = (c = g_1^{x_1} g_2^{x_2}, h = g_1^z)$ and secret key $sk = (x_1, x_2, z)$.
Encryption: $\mathrm{Enc}(pk, m) = (u_1 = g_1^r, u_2 = g_2^r, e = h^r \cdot m, v = c^r)$ where $r \leftarrow \mathbb{Z}_q$.
Decryption: Check if $v = u_1^{x_1} u_2^{x_2}$, return $\frac{e}{u_1^z}$, otherwise return $\perp$

Security Proofs: IND-CCA1 for Cramer-Shoup Lite

Cramer-Shoup Lite scheme (IND-CCA1)

Public key $pk = (c = g_1^{x_1} g_2^{x_2}, h = g_1^z)$ and secret key $sk = (x_1, x_2, z)$.
Encryption: $\mathrm{Enc}(pk, m) = (u_1 = g_1^r, u_2 = g_2^r, e = h^r \cdot m, v = c^r)$
Decryption: Check if $v = u_1^{x_1} u_2^{x_2}$, return $\frac{e}{u_1^z}$, otherwise return $\perp$

- Game 0:
  - Choose x_1, x_2, z, sets : $c = g_1^{x_1} g_2^{x_2}$, $h = g_1^z$
  - For a decryption query (u_1, u_2, e, v):
    check $v \stackrel{?}{=} u_1^{x_1} u_2^{x_2}$, if yes $m = \frac{e}{g_1^z}$
  - Adv chooses m_0, m_1, generates challenge $(g_1^r, g_2^r, e = h^r m_b, v = c^r)$
- Game 1:
  - Choose z_1, z_2 instead of z, sets : $h = g_1^{z_1} g_2^{z_2}$
  - check $v \stackrel{?}{=} u_1^{x_1} u_2^{x_2}$, if yes $m = \frac{e}{u_1^{z_1} u_2^{z_2}}$
  - generates challenge $\left(g_1^r, g_2^r, e = u_1^{z_1} u_2^{z_2} m_b, v = c^r\right)$
- Game 2: Challenge $:= \left(g_1^{r_1}, g_2^{r_2}, e = u_1^{r_1} u_2^{r_2} m_b, v = u_1^{r_1} u_2^{r_2}\right)$.

Exercice: Homomorphism of ElGamal encryption

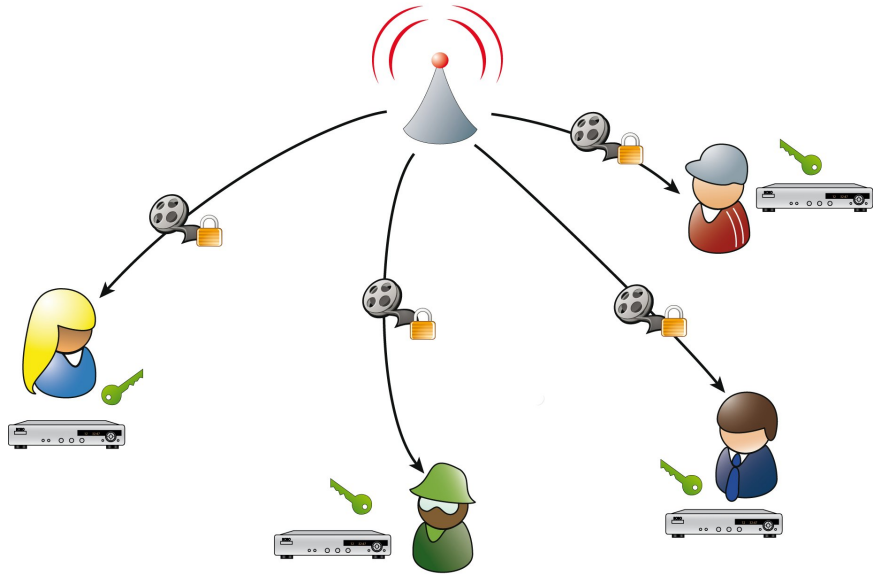
Let $\mathbb{G} = \langle g \rangle$ with generator g of order $|\mathbb{G}| = q$ where q is a prime.
Public key $pk = (g, h = g^x)$ and secret key $sk = x$.
Encryption: $\mathrm{Enc}(pk, m) = (g^r, h^r \cdot m)$ where $r \leftarrow \mathbb{Z}_q$.

- Given a public key pk and an ciphertext c, show how to create a ciphertext c' which encrypts the same message under pk but with independent randomness.
- Given a public key pk and any two independently generated ciphertexts c_1, c_2 encrypting some unknown messages $m_1, m_2 \in \mathbb{G}$ under pk, create a new ciphertext c^* encrypting $m^* = m_1 \cdot m_2$ under pk without needing to know sk, m_1, m_2.

Application: Voting system.

Multi-receiver Encryption

From “One-to-one” to ‘one-to-many” communications

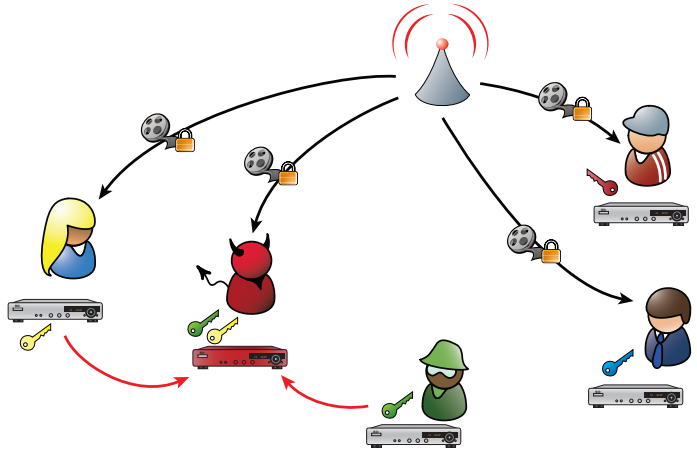


Provide all users with the same key $\rightarrow$ problems:

1. Impossibility to identify the source of the key leakage (traitor)
2. Impossibility to revoke a user, except by resetting the parameters

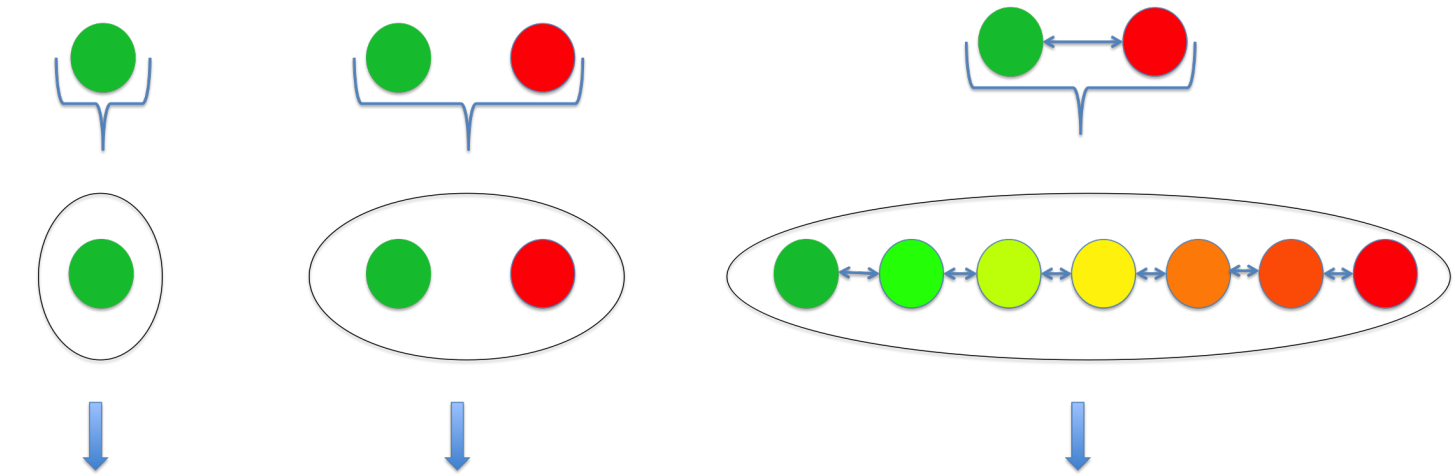
Broadcast Encryption

Revocation [Berkovist91, Fiat-Naor94] & Traitor Tracing [Chor-Fiat-Naor94]



1. Tracing traitors
   - From a pirate key $\rightarrow$ White-box tracing
   - From a pirate decoder (i.e., the pirate can obfuscate its own decryption algorithm and key)
     - ★ Black-box confirmation: tracer has a suspect list
     - ★ Black-box tracing: without any assumption
2. Revoke scheme: encrypt to all but revoked users

Pirate



No collusion

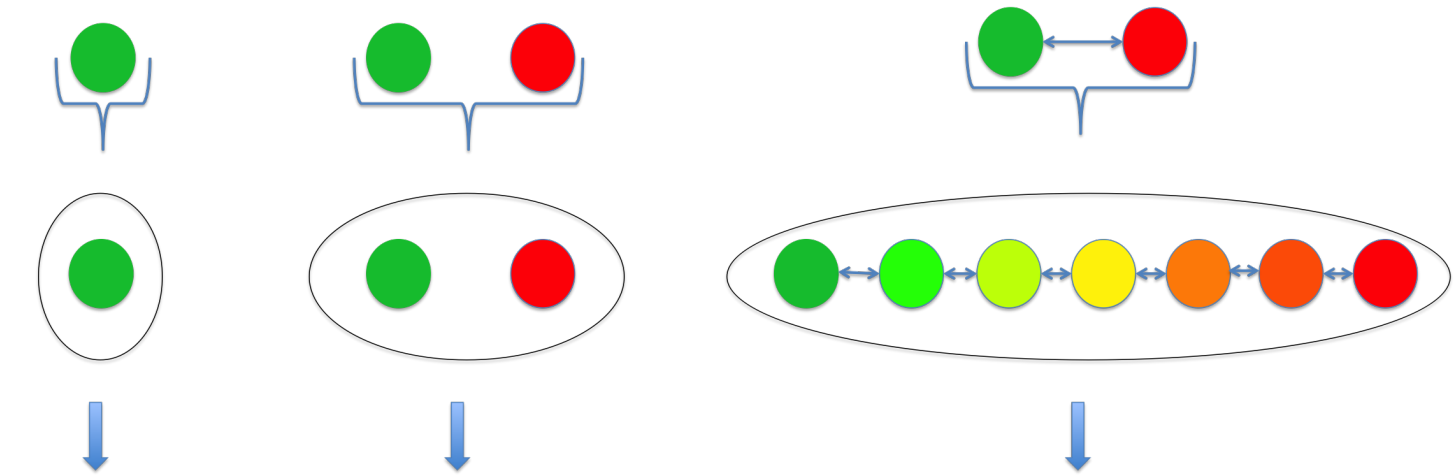
Independent keys
➔Trivial solution
➔ O(N) size CT

Dependent keys
➔ May achieve “short” size CT
➔ Challenge for trace & revoke

Collusion of users → Pirate

The users’ keys are not independent
→ A pirate (from only 2 keys) can produce many pirate keys

Pirate



No collusion

Independent keys
➔Trivial solution
➔ O(N) size CT

Dependent keys
➔ May achieve “short” size CT
➔ Challenge for trace & revoke

Collusion of users → Pirate

The users’ keys are not independent
→ A pirate (from only 2 keys) can produce many pirate keys
→ Tracing and revocation are non trivial, even for small collusions

From Encryption to Multi-receiver Encryption

Dependence between the keys: sharing some algebraic properties

ElGamal Encryption Scheme

- $G =< g >$ of order q
- Secret key: $\alpha \leftarrow \mathbb{Z}_q$
- Public key: $y = g^{\alpha}$
- Ciphertext: $(g^r, y^r m)$, where $r \leftarrow \mathbb{Z}_q$
- Decryption: from α, compute $y^r = (g^r)^{\alpha}$ and recover m

From Encryption to Multi-receiver Encryption

Dependence between the keys: sharing some algebraic properties

ElGamal Encryption Scheme

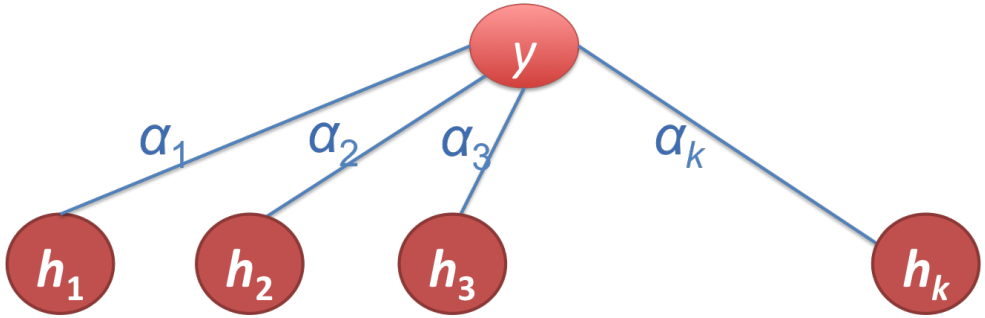
- $G =< g >$ of order q
- Secret key: $\alpha \leftarrow \mathbb{Z}_q$
- Public key: $y = g^{\alpha}$
- Ciphertext: $(g^r, y^r m)$, where $r \leftarrow \mathbb{Z}_q$
- Decryption: from α, compute $y^r = (g^r)^{\alpha}$ and recover m

Multi-receiver Encryption

Main problem: how to extend the same y to support many users?

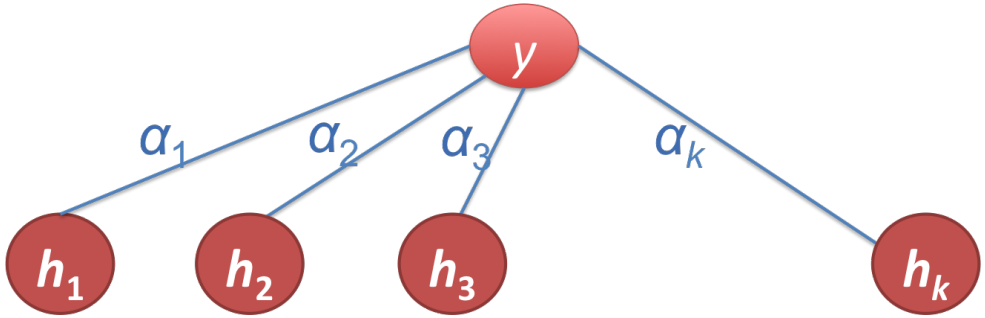
From Encryption to Multi-receiver Encryption

Dependent keys: sharing some algebraic properties [Boneh-Franklin99]



From Encryption to Multi-receiver Encryption

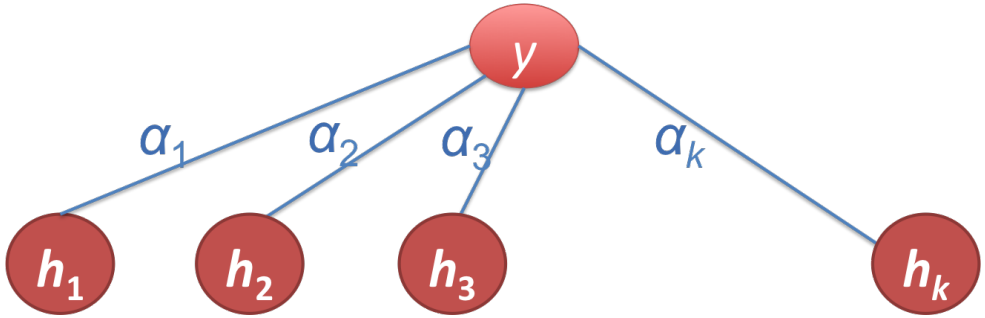
Dependent keys: sharing some algebraic properties [Boneh-Franklin99]



- $G =< g >$ of order q; Public key: $(y, h_1, \ldots, h_k) \in G^{k+1}$
- User key: a representation $(\alpha_1, \ldots, \alpha_k)$ of y in the basis $(h_1, \ldots, h_k)$: $(y = h_1^{\alpha_1} \ldots h_k^{\alpha_k})$

From Encryption to Multi-receiver Encryption

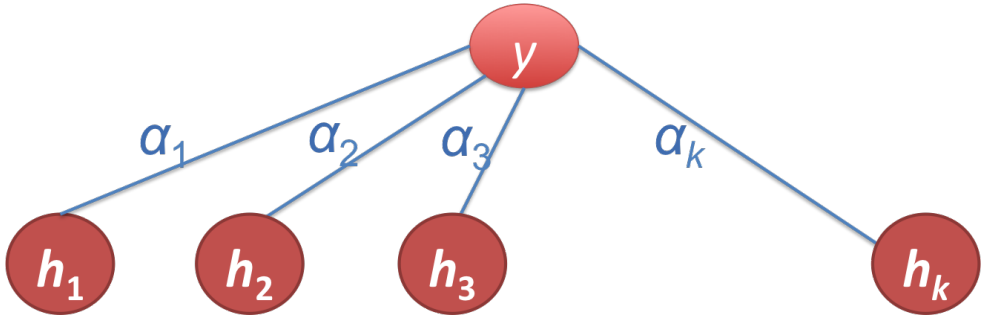
Dependent keys: sharing some algebraic properties [Boneh-Franklin99]



- $G =< g >$ of order q; Public key: $(y, h_1, \ldots, h_k) \in G^{k+1}$
- User key: a representation $(\alpha_1, \ldots, \alpha_k)$ of y in the basis $(h_1, \ldots, h_k)$: $(y = h_1^{\alpha_1} \ldots h_k^{\alpha_k})$
- Ciphertext: $(y^r m, h_1^r, \ldots, h_k^r)$, where $r \leftarrow \mathbb{Z}_q$
- Each user can compute y^r from $(h_1^r, \ldots, h_k^r)$ and recover m

From Encryption to Multi-receiver Encryption

Dependent keys: sharing some algebraic properties [Boneh-Franklin99]



- $G =< g >$ of order q; Public key: $(y, h_1, \ldots, h_k) \in G^{k+1}$
- User key: a representation $(\alpha_1, \ldots, \alpha_k)$ of y in the basis $(h_1, \ldots, h_k)$: $(y = h_1^{\alpha_1} \ldots h_k^{\alpha_k})$
- Ciphertext: $(y^r m, h_1^r, \ldots, h_k^r)$, where $r \leftarrow \mathbb{Z}_q$
- Each user can compute y^r from $(h_1^r, \ldots, h_k^r)$ and recover m

Collusion of 2 users

convex combination $\rightarrow$ q new pirate keys

From Encryption to Multi-receiver Encryption

ElGamal Encryption Scheme

- $G =< g >$ of order q
- Secret key: $\alpha \leftarrow \mathbb{Z}_q$
- Public key: $y = g^{\alpha}$
- Ciphertext: $(g^r, y^r m)$, where $r \leftarrow \mathbb{Z}_q$
- Decryption: from α, compute $y^r = (g^r)^{\alpha}$ and recover m

Boneh-Franklin Multi-receiver Encryption

- Each user receive a representation $(\alpha_1, \ldots, \alpha_k)$ of y in a public basis $(h_1, \ldots, h_k)$: $(y = h_1^{\alpha_1} \ldots h_k^{\alpha_k})$
- Each user can compute y^r from $(h_1^r, \ldots, h_k^r)$
- Public key: $(y, h_1, \ldots, h_k)$
- Ciphertext: $(y^r m, h_1^r, \ldots, h_k^r)$

Boneh-Franklin Scheme

Boneh-Franklin Traitor Tracing

- Transformation from Elgamal Encryption to Traitor Tracing: linear loss in the number of traitors
- Achieve black-box confirmation

Boneh-Franklin Scheme

Boneh-Franklin Traitor Tracing

- Transformation from Elgamal Encryption to Traitor Tracing: linear loss in the number of traitors
- Achieve black-box confirmation

Our Work [Ling-Phan-Stehlé-Steinfeld, Crypto14]

- Study a variant of the Learning With Errors problem [Regev 05], namely k-LWE
- Get a more efficient transformation:

$$\text{LWE-based Encryption} \approx \text{LWE traitor tracing}$$

- Achieve black-box confirmation as in Boneh-Franklin scheme
- Resist quantum attacks

Secret Sharing $\rightarrow$ Threshold BLS Signature

Secret Sharing

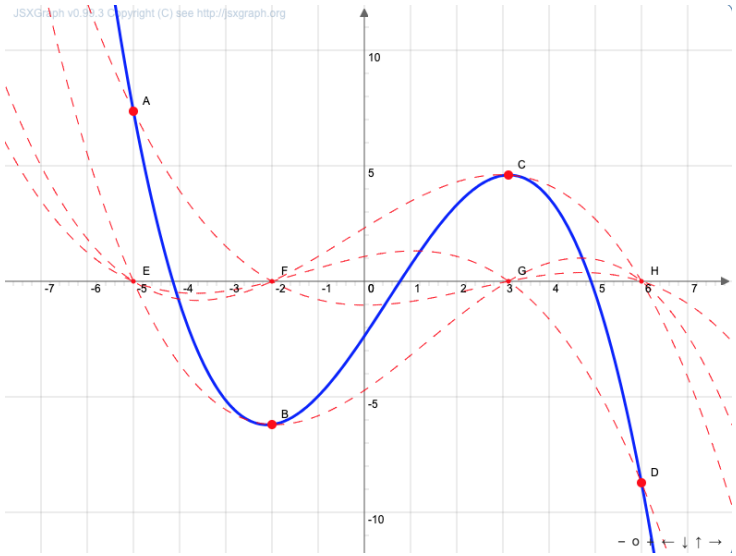
Dealer:

- On input a secret s, choose a polynomial P of degree d such that $P(0) = s$.
- Give to each user i a random point $(x_i, P(x_i))$

Goal:

- any $t = d + 1$ users can do a joint computation to get s
- any $k \leq d$ users get no information abour s.

Secret Sharing → Threshold BLS Signature



Simulation source: `https://inst.eecs.berkeley.edu/~cs70/sp15/hw/vlab7.html`

Tool: Lagrange Polynomial Interpolation

- Given a set of $t = d + 1$ points $(x_0, y_0), \ldots, (x_j, y_j), \ldots, (x_t, y_t)$
- The interpolation polynomial is a linear combination $L(x) := \sum_{j=0}^{k} y_j \ell_j(x)$ of Lagrange basis polynomials

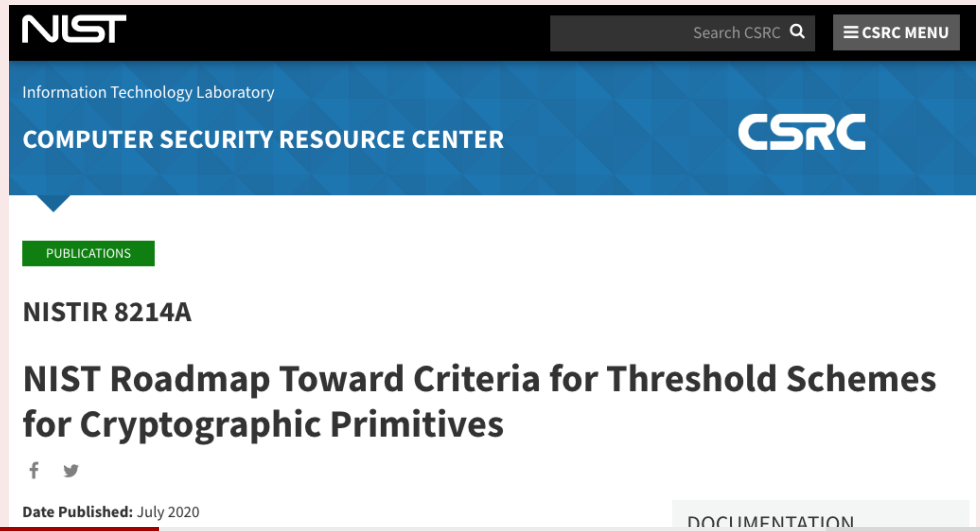
$$\ell_j(x) := \prod_{\substack{0 \le m \le k \\ m \ne j}} \frac{x - x_m}{x_j - x_m} = \frac{(x-x_0)}{(x_j-x_0)} \cdots \frac{(x-x_{j-1})}{(x_j-x_{j-1})} \frac{(x-x_{j+1})}{(x_j-x_{j+1})} \cdots \frac{(x-x_k)}{(x_j-x_k)}$$

Threshold BLS Signature

Exercice: Given a secret sharing scheme, propose a Threshold BLS Signature:

- Each signer receives from the Authority a secret key.
- Each signer signs the message m on its own.
- Any t signers can jointly produce a BLS signatures (Tool: Interpolation on exponents)
- No group of less than t signers can produce a valid BLS signature.

Threshold Cryptography (will see in Advanced Primitives)



Identity-based Encryption

Public key Encryption

- each user generates a couple of public-key/secret-key
- public-key is associated to the identity of the user via a certification $\rightarrow$ complicated public key infrastructure (PKI)

Identity-based Encryption

Shamir 1984 introduced the idea of using the identity of the user as the public-key $\rightarrow$ avoid the PKI.

- extract the secret-key from the public-key
- the extraction is done by an authority, from a trapdoor (master secret key)

Only at the begining of 2000, the first constructions of IBE were introduced.

PKE vs. IBE?

1. CCA PKE from CPA IBE [Boneh-Canetti-Halevi-Katz 2006]
2. No black-box construction of IBE from CCA-PKE [Dan Boneh-Papakonstantinou-Rackoff-Vahlis-Waters 2008]

Why is it difficult to construct an IBE?

1. Design:
   - In a PKE, one often generates a public key from a secret key. Well-formed public keys might be exponentially sparse.
   - In an IBE scheme:
     - ★ any identity should be publicly mapped to a public key
     - ★ extract secret key from public-key via a trapdoor.

Why is it difficult to construct an IBE?

1. Design:
   - In a PKE, one often generates a public key from a secret key. Well-formed public keys might be exponentially sparse.
   - In an IBE scheme:
     - ⋆ any identity should be publicly mapped to a public key
     - ⋆ extract secret key from public-key via a trapdoor.
2. Security: in IBE, the adversary can corrupt secret keys $\rightarrow$ the simulator should be able to simulate all key queries except the challenge identity.

Brief History of IBE

First idea by Shamir in 84.
There are five families of IBE schemes from:

- elliptic curves pairing: Sakai Ohgishi Kasahara in 2000, Boneh Franklin in 2001.
- quadratic residues: Cocks in 2001.
- lattice: Gentry Peikert Vaikuntanathan in 2008.
- computational Diffie-Hellman: Dottling-Garg in 2017.
- coding: Gabotit-Hauteville-Phan-Tillich in 2017

Brief History of IBE

First idea by Shamir in 84.
There are five families of IBE schemes from:

- elliptic curves pairing: Sakai Ohgishi Kasahara in 2000, Boneh Franklin in 2001.
- quadratic residues: Cocks in 2001.
- lattice: Gentry Peikert Vaikuntanathan in 2008.
- computational Diffie-Hellman: Dottling-Garg in 2017.
- coding: Gabotit-Hauteville-Phan-Tillich in 2017

Elgamal Encryption $\rightarrow$ IBE?

- $G =< g >$ of order q
- Secret key: $s \leftarrow \mathbb{Z}_q$
- Public key: $y = g^s$
- Ciphertext: $(g^r, y^r m)$, where $r \leftarrow \mathbb{Z}_q$
- Decryption: from s, compute $y^r = (g^r)^s$ and recover m

Transform to IBE:

1. Public key: define $y = H(id) = g^s \rightarrow$ can we extract s?
2. Possible in bilinear groups $\rightarrow$ Boneh-Franklin scheme

Elgamal Encryption $\rightarrow$ IBE? (with Pairings)

ElGamal:

- Secret key: random s
- Public key: $y = g^s$
- Ciphertext: $(g^r, y^r m)$, for a random r
- Decryption: from s, compute $y^r = (g^r)^s$ and recover m

Boneh-Franklin IBE [2001]

$$y_{id} = e(g, H(id))^s = e(g, H(id)^s) = e(g^s, H(id))$$

Elgamal Encryption $\rightarrow$ IBE? (with Pairings)

ElGamal:

- Secret key: random s
- Public key: $y = g^s$
- Ciphertext: $(g^r, y^r m)$, for a random r
- Decryption: from s, compute $y^r = (g^r)^s$ and recover m

Boneh-Franklin IBE [2001]

$$y_{id} = e(g, H(id))^s = e(g, H(id)^s) = e(g^s, H(id))$$

Considering s as trapdoor (master secret key), g^s as a public then:

- "Public key" $y_{id} = e(g^s, H(id)$ is computable from id
- Secret key can be extracted as $sk_{id} = H(id)^s$.
- Ciphertext: $(g^r, y_{id}^r m)$
- Decryption: from $H(id)^s$, compute $y_{id}^r = e(g^r, H(id)^s)$ and recover m

Computing on Encrypted Data

Computing on Encrypted Data: FHE/ Functional Encryption

Fully homomorphic encryption

- RSA is additionally homomorphic
- ElGamal is multiplicatively homomorphic

It was an long standing open question to construct a fully homomorphic encryption until the breakthrough of Gentry 09.

Computing on Encrypted Data: FHE/ Functional Encryption

Fully homomorphic encryption

- RSA is additionally homomorphic
- ElGamal is multiplicatively homomorphic

It was an long standing open question to construct a fully homomorphic encryption until the breakthrough of Gentry 09.

Functional Encryption

- Classical encryption: $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(m)) = m$
- Functional encryption: $\mathcal{FE}.\mathsf{Dec}(\mathsf{sk}_f, \mathcal{FE}.\mathsf{Enc}(\mathbf{m})) = f(m)$

Functional Encryption / Inner-Product FE

Functional Encryption

- Classical encryption: $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(m)) = m$
- Functional encryption: $\mathcal{FE}.\mathsf{Dec}(\mathsf{sk}_f, \mathcal{FE}.\mathsf{Enc}(\mathbf{m})) = f(m)$

Functional Encryption / Inner-Product FE

Functional Encryption

- Classical encryption: $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(m)) = m$
- Functional encryption: $\mathcal{FE}.\mathsf{Dec}(\mathsf{sk}_f, \mathcal{FE}.\mathsf{Enc}(\mathbf{m})) = f(m)$

Inner-Product Functional Encryption over $\mathbb{Z}_p^\ell$

- secret key encodes a vector $\mathbf{x} \in \mathbb{Z}_p^\ell$: $\mathcal{FE}.\mathsf{KeyGen}(\mathbf{x}) \to \mathsf{sk}_\mathbf{x}$
- ciphertext encodes a vector $\mathbf{v} \in \mathbb{Z}_p^\ell$: $\mathcal{FE}.\mathsf{Enc}(\mathsf{pk}, \mathbf{v}) \to C$
- decryption recovers the inner product

$$\mathcal{FE}.\mathsf{Dec}(\mathsf{sk}_\mathbf{x}, C) \to \langle \mathbf{x}, \mathbf{v} \rangle \bmod p$$

Functional Encryption / Inner-Product FE

Functional Encryption

- Classical encryption: $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(m)) = m$
- Functional encryption: $\mathcal{FE}.\mathsf{Dec}(\mathsf{sk}_f, \mathcal{FE}.\mathsf{Enc}(\mathbf{m})) = f(m)$

Inner-Product Functional Encryption over $\mathbb{Z}_p^\ell$

- secret key encodes a vector $\mathbf{x} \in \mathbb{Z}_p^\ell$: $\mathcal{FE}.\mathsf{KeyGen}(\mathbf{x}) \to \mathsf{sk}_\mathbf{x}$
- ciphertext encodes a vector $\mathbf{v} \in \mathbb{Z}_p^\ell$: $\mathcal{FE}.\mathsf{Enc}(\mathsf{pk}, \mathbf{v}) \to C$
- decryption recovers the inner product

$$\mathcal{FE}.\mathsf{Dec}(\mathsf{sk}_\mathbf{x}, C) \to \langle \mathbf{x}, \mathbf{v} \rangle \bmod p$$

- Efficient solutions [ADBP15, ALS15...]
- Our new result: Decentralized multi-client IPFE (Asiacrypt '18)

Different Tools for the Design of Advanced Primitives

- **Group, Pairings:** IBE, BE, TT [1], ABE, zk-SNARK, Voting, Inner-Product FE, Decentralized IPFE [2], 2-DNF FHE.
- **Lattice:** IBE, BE&TT [3,4], ABE, Inner-Product FE, FHE.
- **Coding:** IBE [5]
- **Combinatorics**: Group testing, Collusion secure code, IPP code, BE, Trace & Revoke code [6].

$\rightarrow$ A large number of open problems!

Concluding Discussions

- Standard primitives:
  - Encryption for confidentiality
  - Hash functions for integrity
  - MAC, digital signature for authentification
  - Interactive, zero-knowledge proofs (used in IND-CCA PKE, multi-party computation,...)

Concluding Discussions

- Standard primitives:
  - Encryption for confidentiality
  - Hash functions for integrity
  - MAC, digital signature for authentification
  - Interactive, zero-knowledge proofs (used in IND-CCA PKE, multi-party computation,...)
- Advanced primitives:
  - Multi-user cryptography (BE, TT, ABE, GS...)
  - Computing in encrypted data (FHE, FE, machine learning/AI on encrypted data...)

Concluding Discussions

- Standard primitives:
  - Encryption for confidentiality
  - Hash functions for integrity
  - MAC, digital signature for authentification
  - Interactive, zero-knowledge proofs (used in IND-CCA PKE, multi-party computation,...)
- Advanced primitives:
  - Multi-user cryptography (BE, TT, ABE, GS...)
  - Computing in encrypted data (FHE, FE, machine learning/AI on encrypted data...)
- In these revolutionary years of technology:
  - Everyone should care about the privacy and the confidentiality
  - No abuse of data access, from the companies or from the governments
  - Should deal with powerful adversaries (quantum, collaborative attacks,...)