Modern Cryptography

Phan Duong Hieu

Telecom Paris, IPP

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

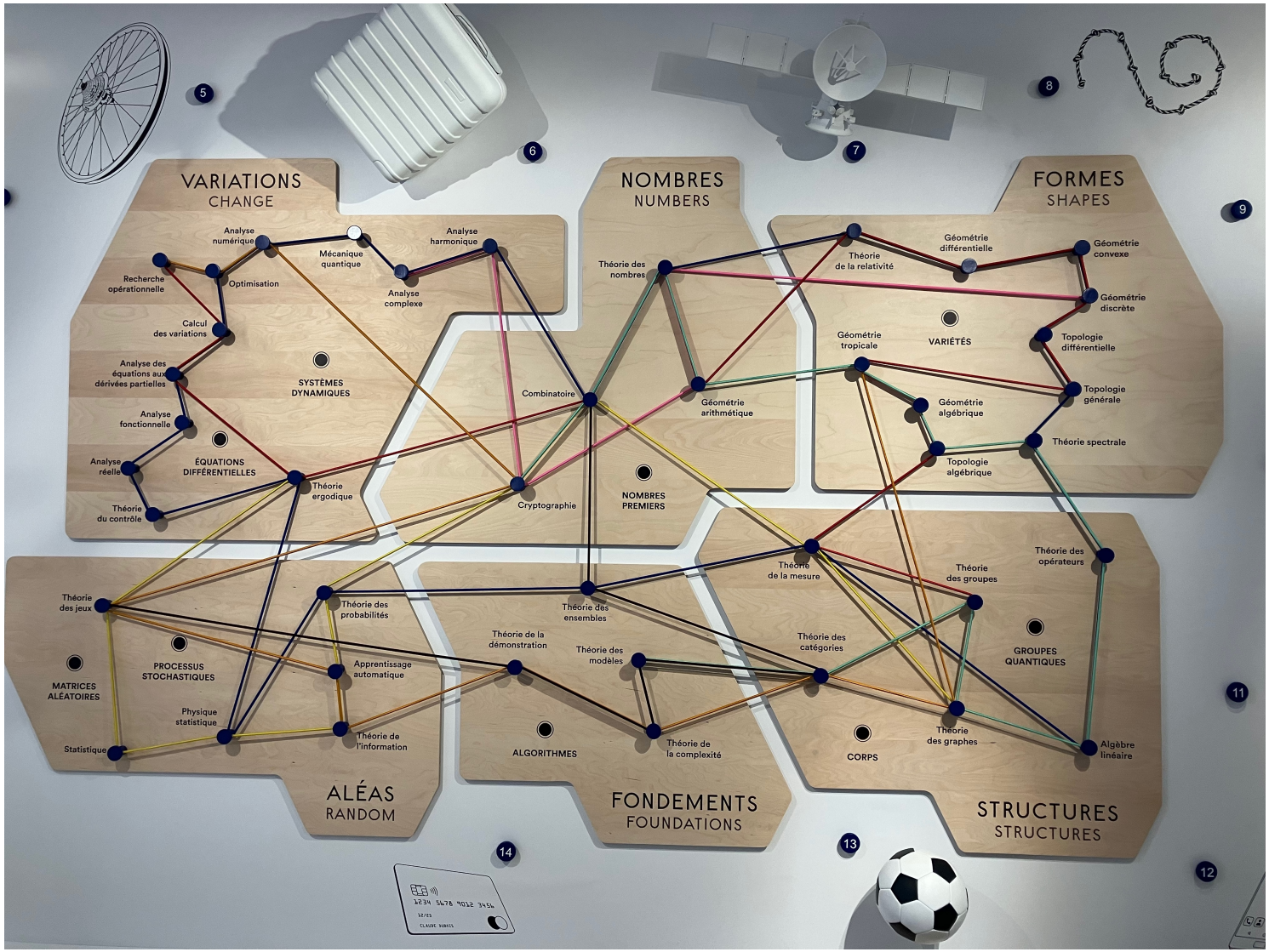
- Big Data, Cloud Computing $\rightarrow$ widespread real-life applications
- Privacy: protect personal information.
  - Security
  - Trust on Authorities

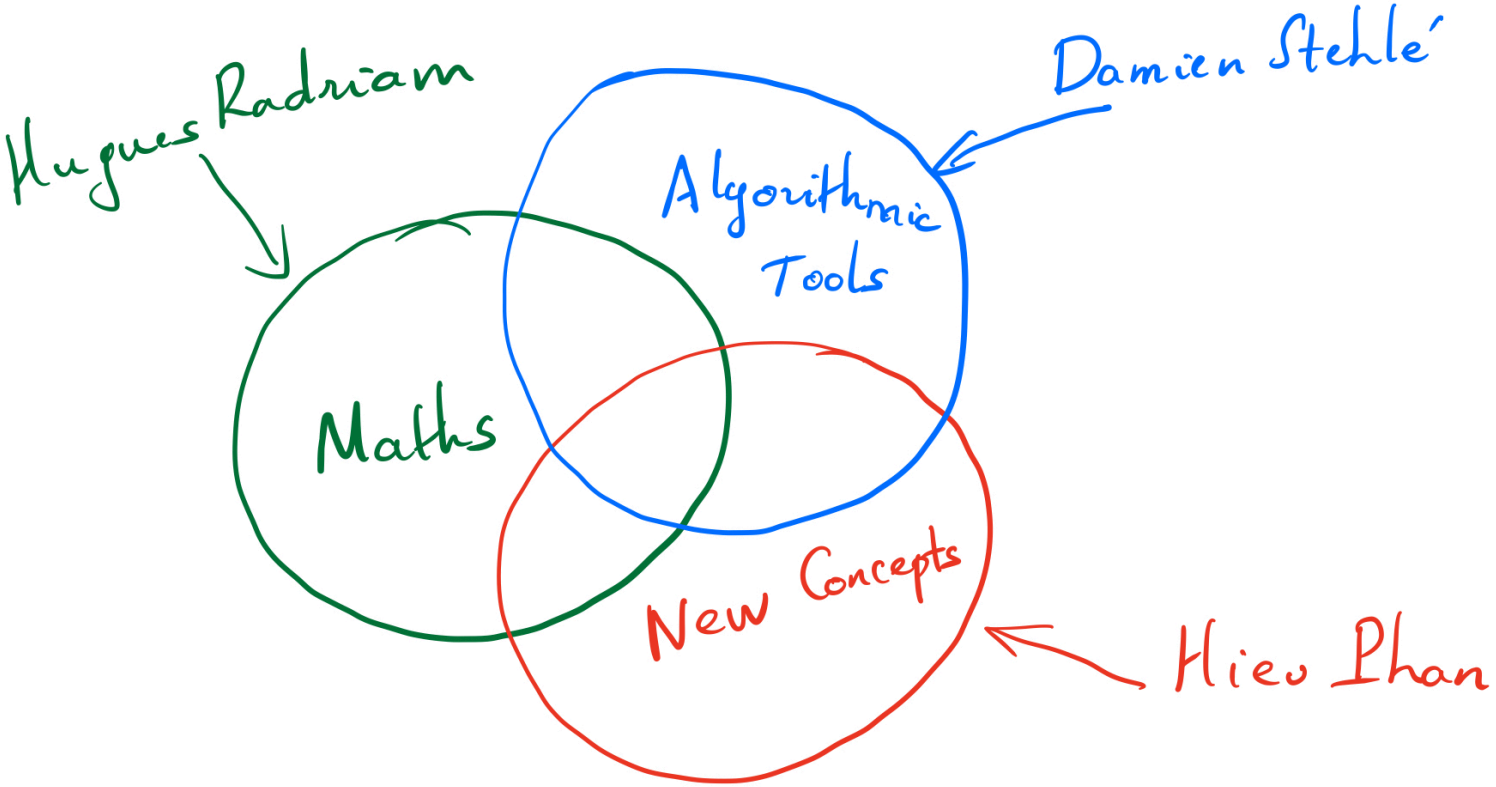
  $\rightarrow$ **Security of Computation on Untrusted Machines.**

Documentation:

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

Cryptography in Museum of Mathematics





Anamorphic View of Cryptography

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

Public-key Encryption (Diffie-Helmann 1976)

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

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

Modern Cryptography

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?

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.)

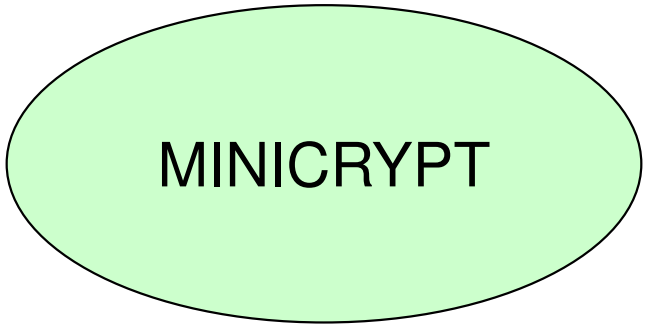
Minicrypt: One-way functions exist but public-key cryptography does not exist.

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

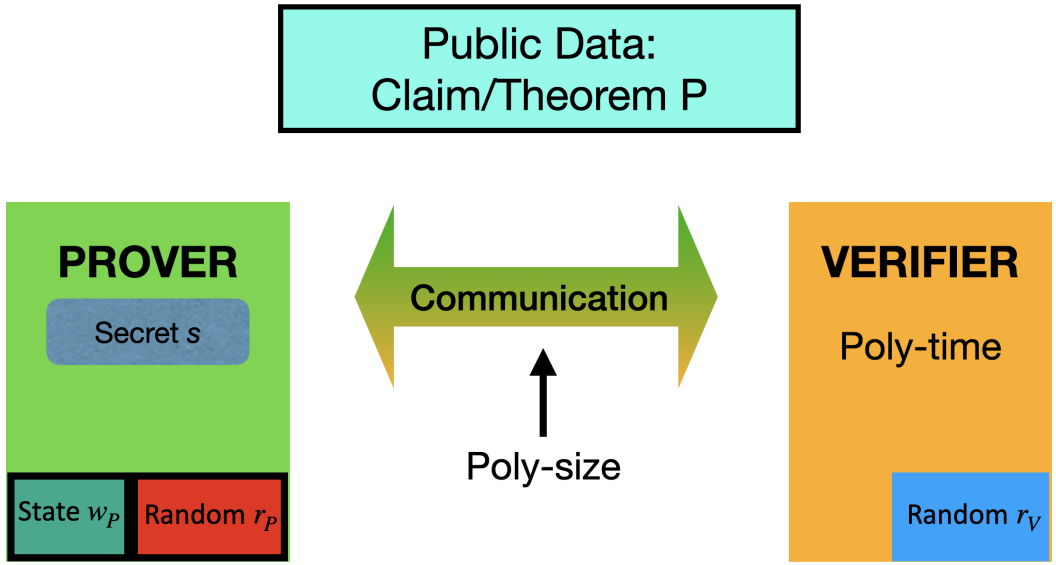
Interactive proofs [Goldwasser, Micali, Rackoff '85]

"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:

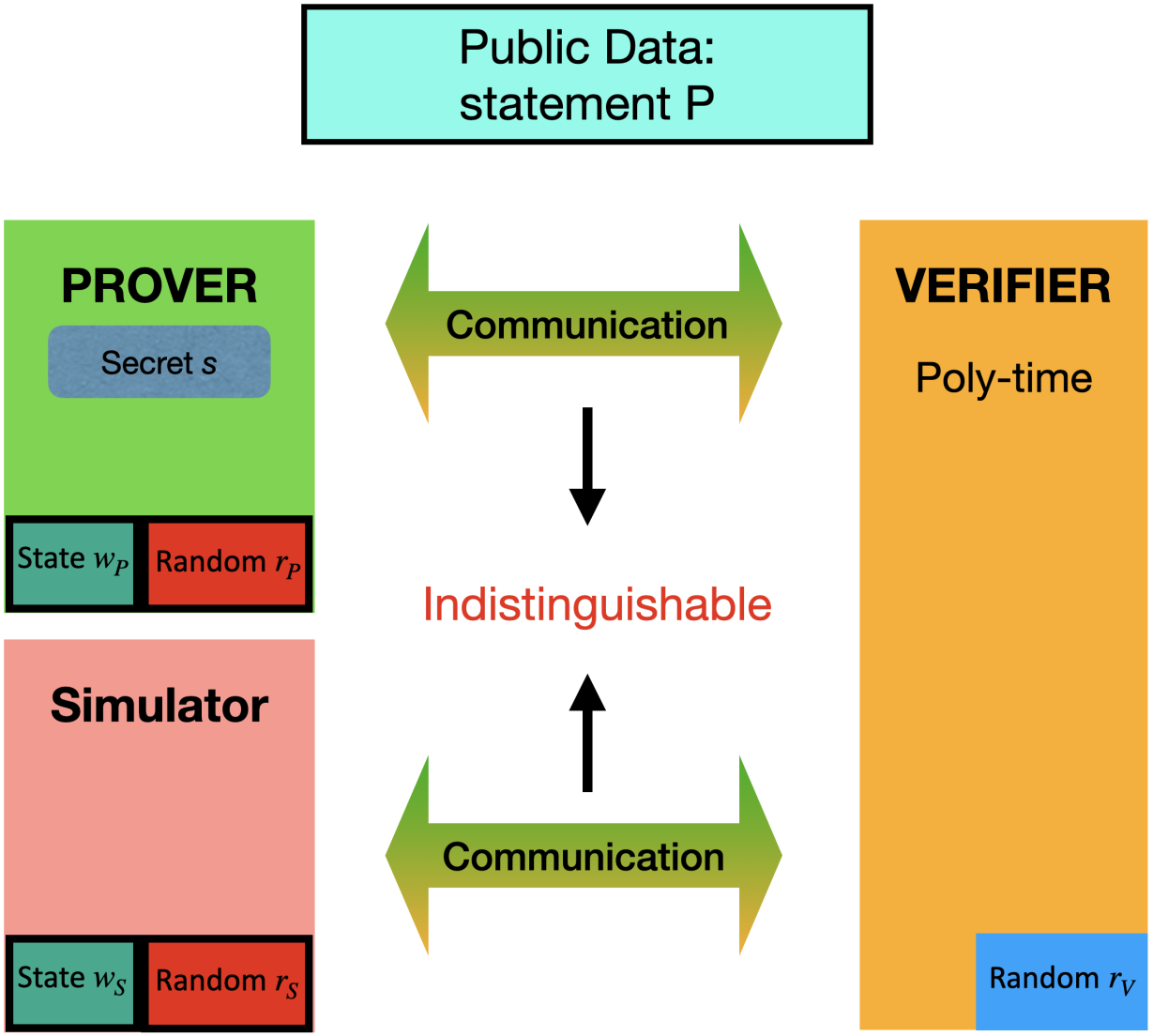
- **Completeness:** If $x \in \mathcal{L}$,

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

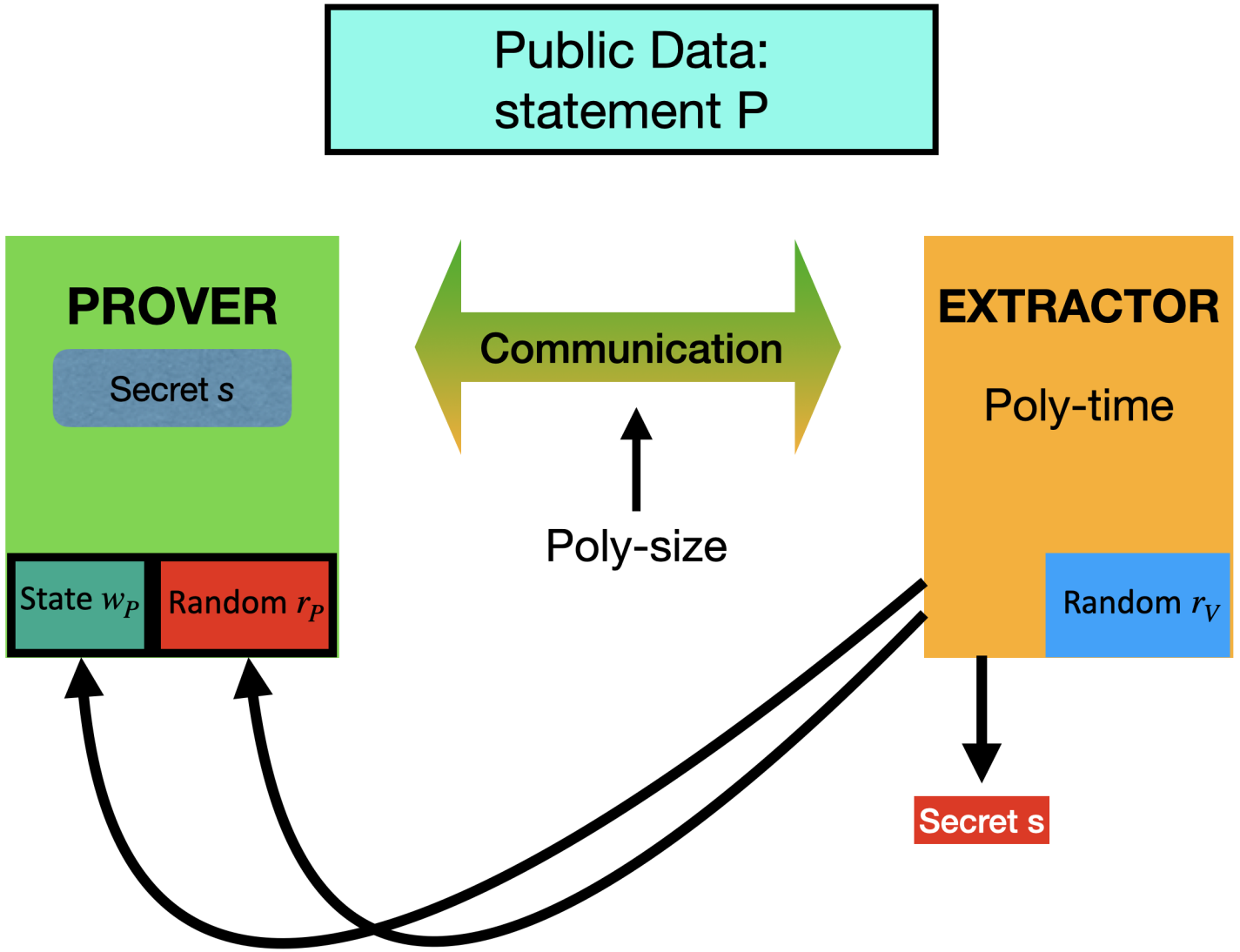
- **Soundness:** If $x \notin \mathcal{L}$, for every P^*,

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

Zero-knowledge Proof: Simulator



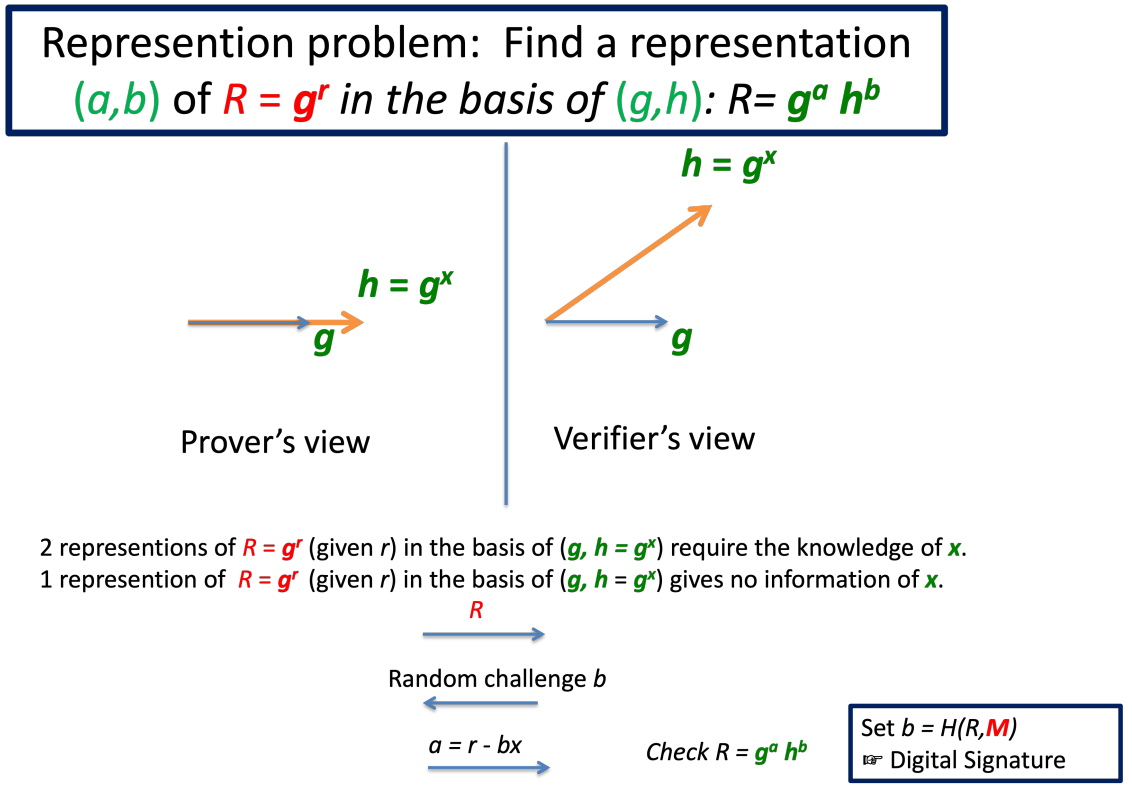
Zero-knowledge Proof of Knowledge: Extractor



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

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

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 $(u = g^x, v = h^x)$ and wants to prove it to $\mathcal{V}$
- $\mathcal{P}$ chooses $r \xleftarrow{R} \mathbb{Z}_q^*$, sets and sends $U = g^r$ and $V = h^r$
- $\mathcal{V}$ chooses $b \xleftarrow{R} \{0,1\}^t$ and sends it to $\mathcal{P}$
- $\mathcal{P}$ computes and sends $a = r - bx \mod q$
- $\mathcal{V}$ checks whether $U \stackrel{?}{=} g^a u^b$ and $V \stackrel{?}{=} h^a v^b$

Extractor: for a fixed (U, V), two valid answers s and s' satisfy

$$g^s u^h = U = g^{s'} u^{h'} \quad \text{and} \quad h^s v^h = V = h^{s'} v^{h'}$$

- if one sets $y = (s - s')(h' - h)^{-1} \mod q \implies u = g^y$ and $v = h^y$

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:

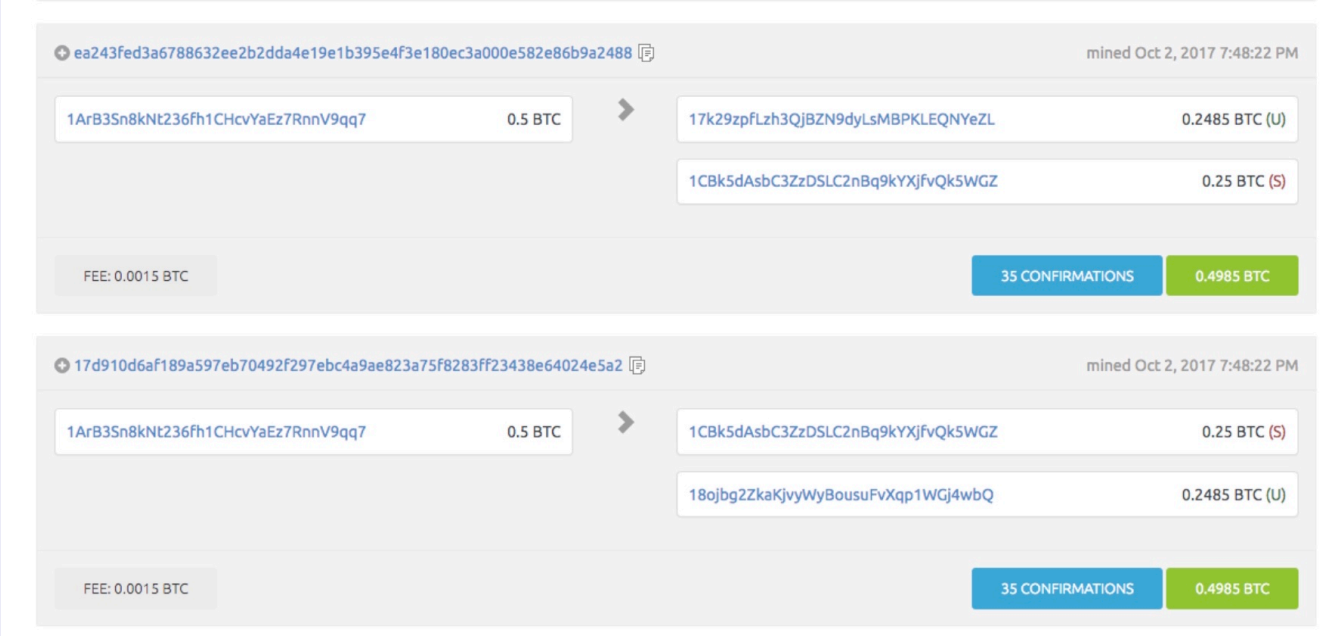
- **Hiding** Commitment c hides information on m
- **Binding** Alice cannot open c to $(m', r') \neq (m, r)$

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



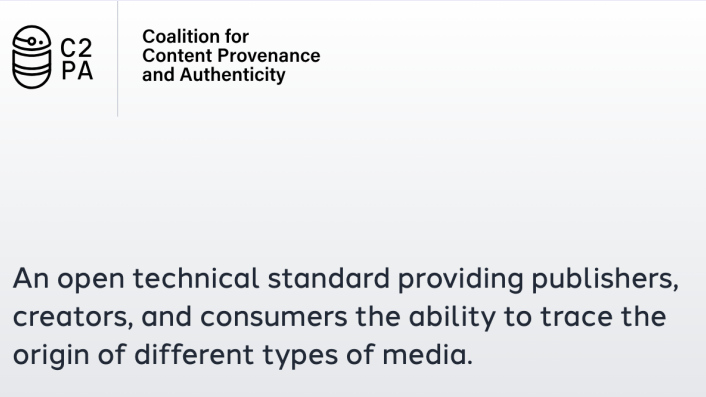
Privacy

- 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