A Guided Tour through the Jungle of Arithmetization-Oriented Primitives

PART 1

Clémence Bouvier

Université de Lorraine, CNRS, Inria, LORIA

SAC Summer School, Toronto, Canada
August 12th, 2025

Emerging uses

Decentralized identity

Blockchain

Internet of Things

Electronic Voting

and many more...

New symmetric primitives

How to build symmetric primitives ?

- **Efficiency** - What does it mean ?
  - Allowed operations ?
    - logical gates
    - CPU instructions
    - AES round
    - arithmetic operations
    - ...
  - Cost metric ?
    - throughput
    - hardware
    - RAM consumption
    - number of multiplications
    - ...
- **Security** – How do we defined it ?
- **Context** - Single or multiple use cases ?

Outline

PART 1

- ⋆ General Introduction

- ⋆ Advanced Protocols

- ⋆ New AOPs

- ⋆ Computing constraints

PART 2

- ⋆ Design of AOPs

- ⋆ Algebraic Cryptanalysis

- ⋆ Other attacks

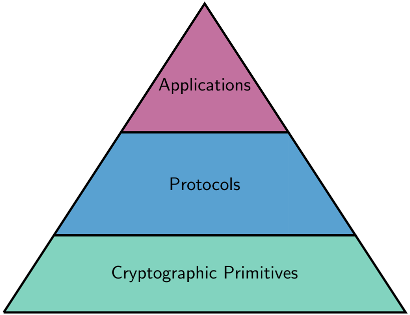
General Introduction

New context

Block Ciphers

Hash functions

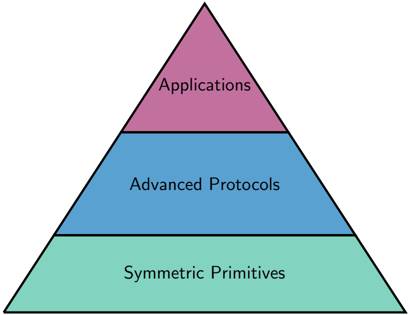
A need for new primitives



A need for new primitives

Protocols requiring new primitives :

- ⋆ **FHE** : Fully Homomorphic Encryption
- ⋆ **MPC** : Multiparty Computation
- ⋆ **ZK** : Systems of Zero-Knowledge proofs
  Example : SNARKs, STARKs, Bulletproofs



Problem : Designing new symmetric primitives

Secure a computation

Exchange secure messages

- **Confidentiality**

  No external party can read the message.

- **Integrity**

  No external party can modify it.

- **Authentication**

  The message was written by the right person.

Secure a computation

Exchange secure messages

⋆ **Confidentiality**

No external party can read the message.

⋆ **Integrity**

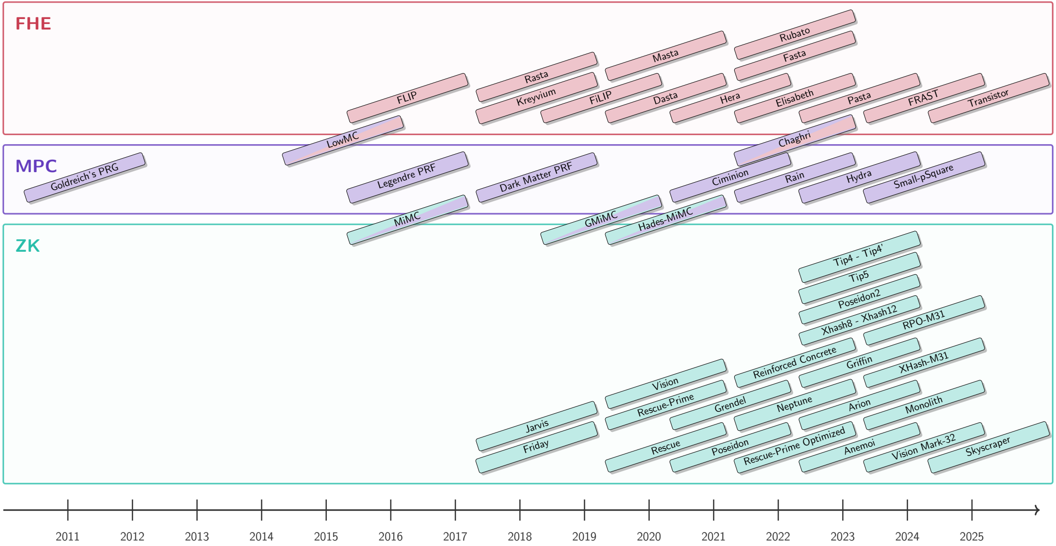
No external party can modify it.

⋆ **Authentication**

The message was written by the right person.

Secure a computation

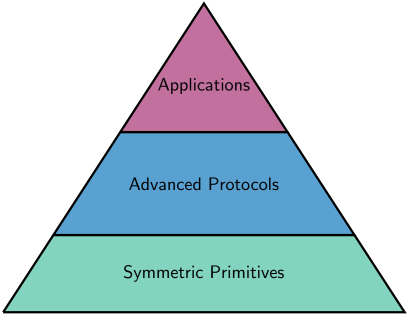
Primitives


FHE
MPC
ZK
Goldreich's PRG
LowMC
FLIP
Legendre PRF
MiMC
Rasta
Kreyvium
Dark Matter PRF
Jarvis
Friday
FiLIP
GMiMC
Masta
Dasta
Hades-MiMC
Vision
Rescue-Prime
Rescue
Hera
Ciminion
Grendel
Poseidon
Rubato
Fasta
Elisabeth
Chaghri
Rain
Reinforced Concrete
Neptune
Rescue-Prime Optimized
Pasta
Hydra
Tip4 - Tip4'
Tip5
Poseidon2
Xhash8 - Xhash12
Griffin
Arion
Anemoi
FRAST
Small-pSquare
RPO-M31
XHash-M31
Monolith
Vision Mark-32
Transistor
Skyscraper
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025

A need for new primitives

Protocols requiring new primitives :

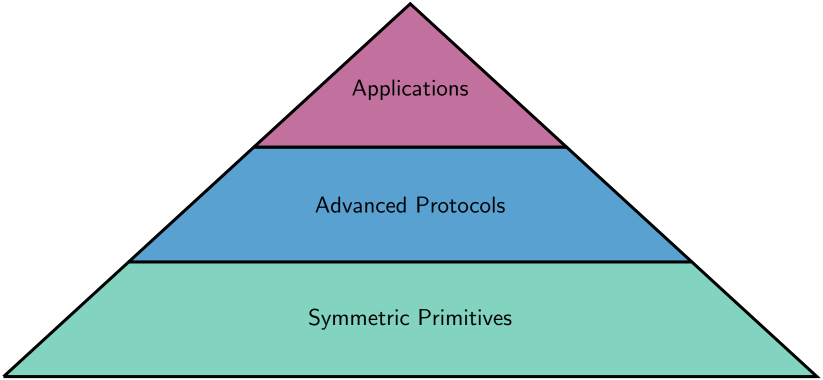
- ⋆ **FHE** : Fully Homomorphic Encryption
- ⋆ **MPC** : Multiparty Computation
- ⋆ **ZK** : Systems of Zero-Knowledge proofs
  Example : SNARKs, STARKs, Bulletproofs



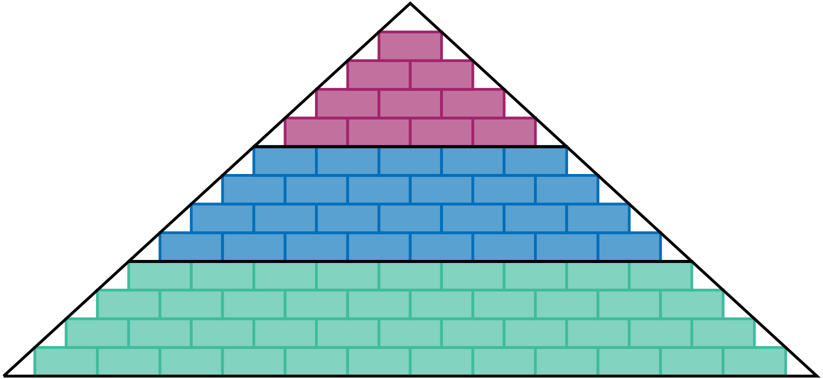
Problem : Designing new symmetric primitives

And analyse their security !

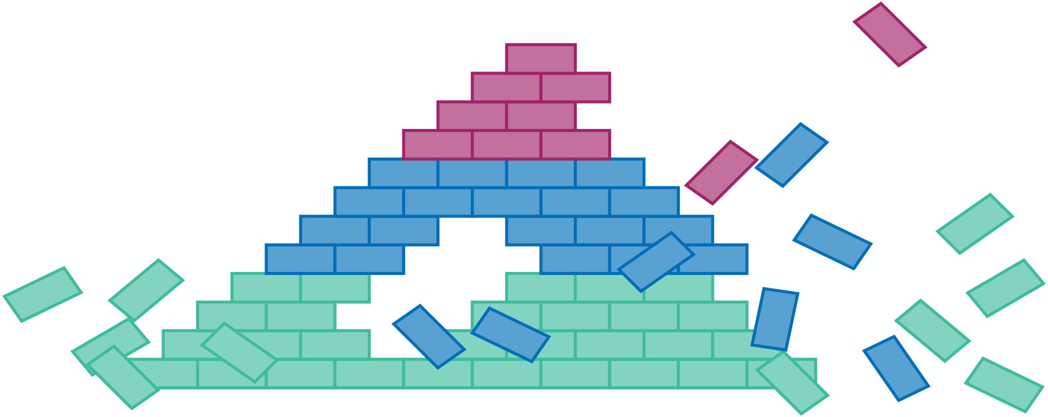
Building blocks of security



Building blocks of security



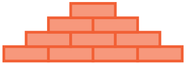
Cycle primitive



Primitive life cycle

Conception

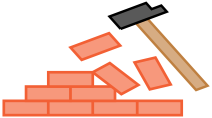
- ⋆ Specification of the algorithm
- ⋆ Justification of design choices
- ⋆ First security analysis



Publication

Analysis

- ⋆ Trying to break algorithms

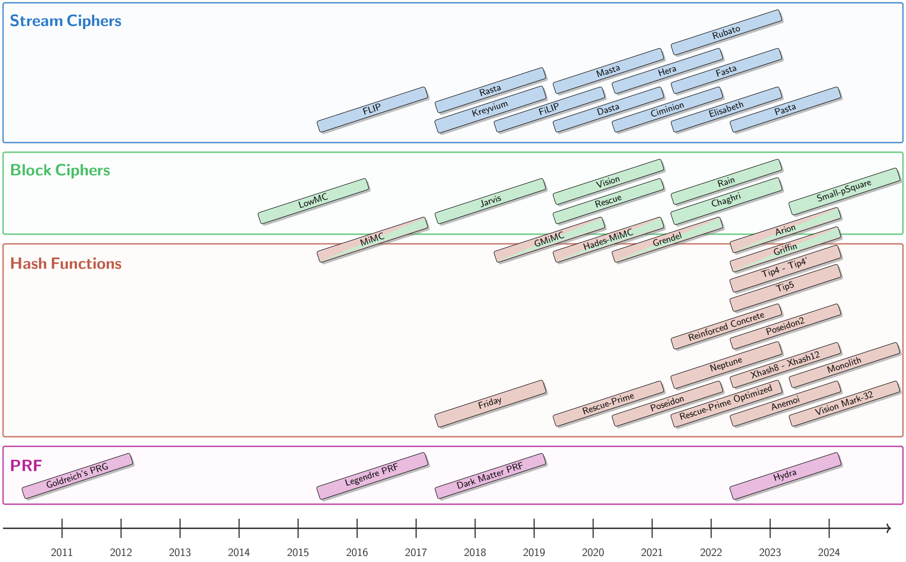


Standardization

Deployment

- ⋆ Implementation of algorithms

Primitives


Stream Ciphers
FLIP
Rasta
Kreyvium
FiLIP
Masta
Dasta
Hera
Ciminion
Rubato
Fasta
Elisabeth
Pasta
Block Ciphers
LowMC
MiMC
Jarvis
GMiMC
Vision
Rescue
Hades-MiMC
Grendel
Rain
Chaghri
Small-pSquare
Arion
Griffin
Hash Functions
Tip4 - Tip4'
Tip5
Reinforced Concrete
Poseidon2
Neptune
Xhash8 - Xhash12
Monolith
Friday
Rescue-Prime
Poseidon
Rescue-Prime Optimized
Anemoi
Vision Mark-32
PRF
Goldreich's PRG
Legendre PRF
Dark Matter PRF
Hydra
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024

Block ciphers

- input :

  n-bit bloc

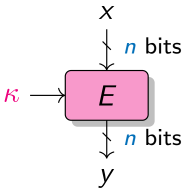
- parameter :

  k-bit key

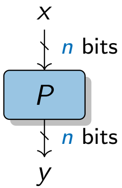
- output :

  n-bit bloc

- symmetry : E and E^{-1} use the same κ



(a) *Block cipher*



(b) *Random permutation*

Block ciphers

⋆ input :

n-bit bloc

⋆ parameter :

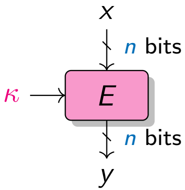
k-bit key

⋆ output :

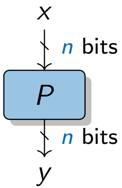
n-bit bloc

⋆ symmetry : E and E^{-1} use the same κ

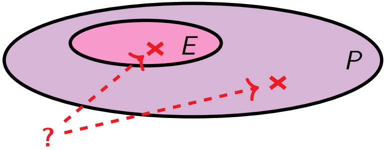
A block cipher is a family of permutations of n-bit blocs.



(a) *Block cipher*



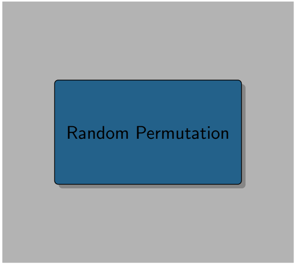
(b) *Random permutation*



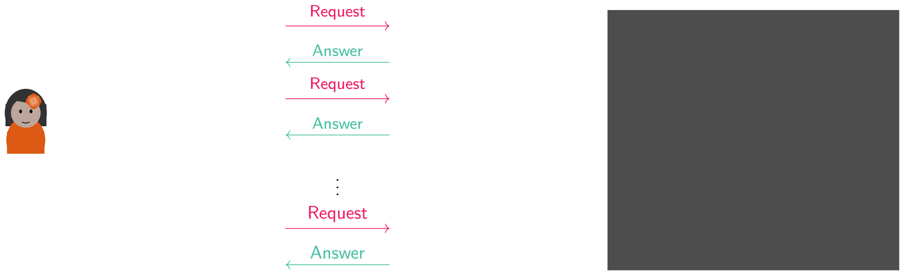
Indistinguishability

Indistinguishability

Indistinguishability

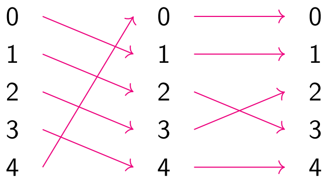


Indistinguishability



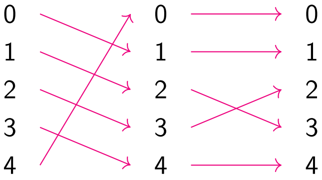
Indistinguishability

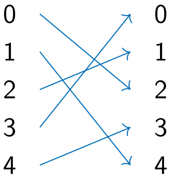
Example $E : x \mapsto (x+k)^3$ with $x \in \{0,1,2,3,4\}$ and $k = 1$.



Indistinguishability

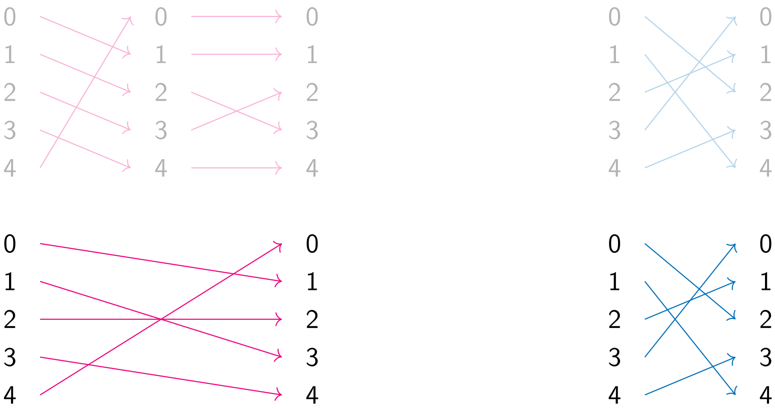
Example $E : x \mapsto (x+k)^3$ with $x \in \{0,1,2,3,4\}$ and $k = 1$.





Indistinguishability

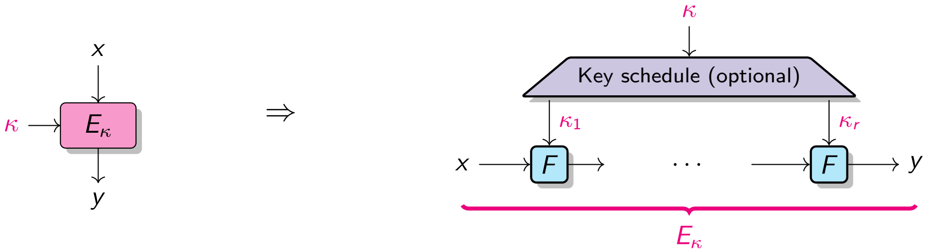
Example $E : x \mapsto (x+k)^3$ with $x \in \{0,1,2,3,4\}$ and $k = 1$.



Iterated constructions

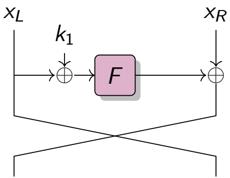
How to build an efficient block cipher ?

By iterating a round function.

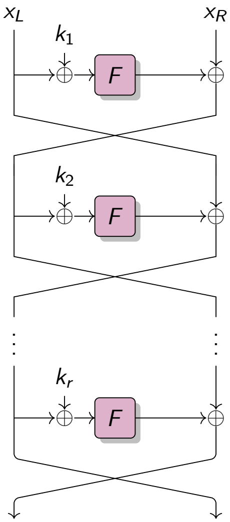


Performance constraints ! The primitive must be fast.

Feistel construction

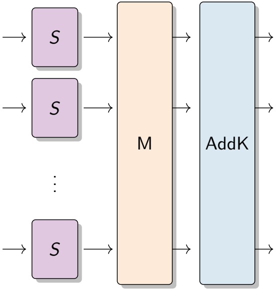


Feistel construction



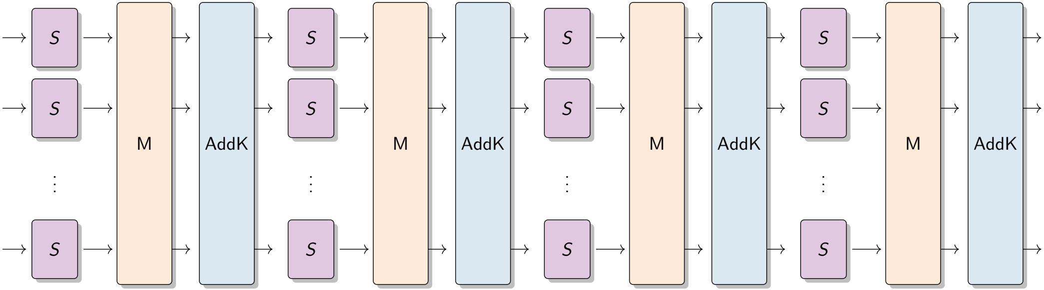
SPN construction

SPN = Substitution Permutation Networks



SPN construction

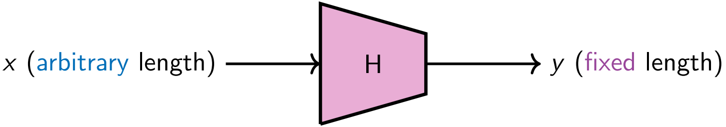
SPN = Substitution Permutation Networks



Hash functions

Definition

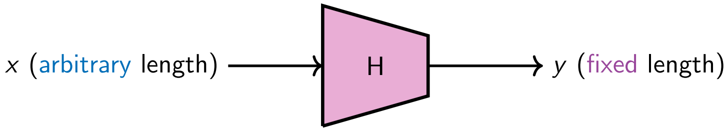
Hash function : $H : \mathbb{F}_q^{\ell} \rightarrow \mathbb{F}_q^{h}, x \mapsto y = H(x)$ where ℓ is arbitrary and h is fixed.



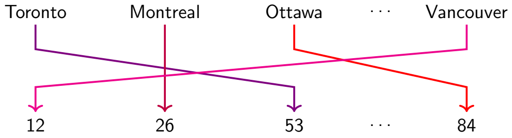
Hash functions

Definition

Hash function : $H : \mathbb{F}_q^{\ell} \rightarrow \mathbb{F}_q^{h}, x \mapsto y = H(x)$ where ℓ is arbitrary and h is fixed.



Example : Hash table of Canadian cities



Collisions and Preimages

Preimage resistance

Given y it must be *infeasible* to find :

$$\boxed{x \quad \text{such that} \quad H(x) = y}$$

Collision resistance

It must be *infeasible* to find :

$$\boxed{x \neq x' \quad \text{such that} \quad H(x) = H(x')}$$

Collisions and Preimages

Preimage resistance

Given y it must be *infeasible* to find :

$$x \quad \text{such that} \quad H(x) = y$$

Collision resistance

It must be *infeasible* to find :

$$x \neq x' \quad \text{such that} \quad H(x) = H(x')$$

Birthday paradox

In a group of **23** *people, there is a* **51%** *chance that 2 people will have their birthday on the same day.*

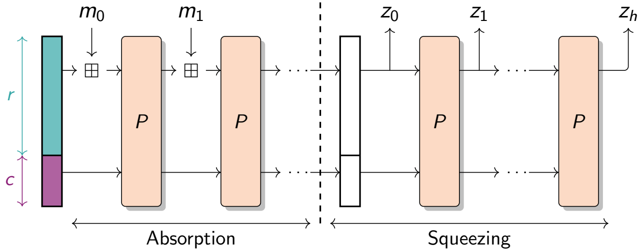
After computing $2^{n/2}$ hashes, there is a very good chance of obtaining a collision.

Sponge construction

Sponge construction

Parameters :

- ⋆ rate $r > 0$
- ⋆ capacity $c > 0$
- ⋆ permutation of $\mathbb{F}_q^n$ $(n = r + c)$

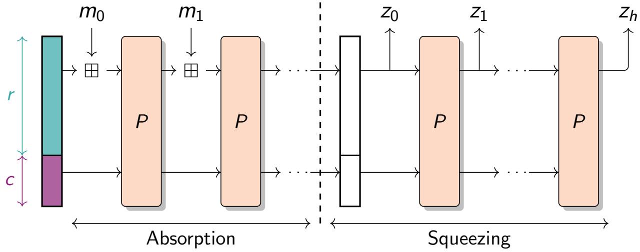


Sponge construction

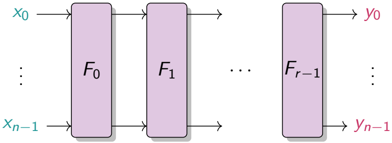
Sponge construction

Parameters :

- ⋆ rate $r > 0$
- ⋆ capacity $c > 0$
- ⋆ permutation of $\mathbb{F}_q^n$ $(n = r + c)$



P is an iterated construction



CICO problem

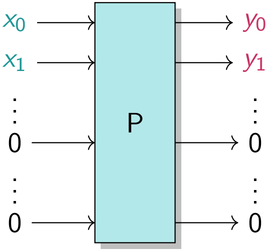
CICO : Constrained Input Constrained Output

Definition

Let $P : \mathbb{F}_q^{r+c} \rightarrow \mathbb{F}_q^{r+c}$. The **CICO** problem is :

Finding $X, Y \in \mathbb{F}_q^r$ s.t.

$$P(X, 0^c) = (Y, 0^c)$$



QUIZ ! !

- ⋆ Is a hash function a symmetric primitive ? True or False ?
- ⋆ Are AOPs part of a revolution in symmetric crypto ?
- ⋆ Does Patrick Derbez have some of the best attack on AES ?

Take-away

Symmetric techniques for advanced protocols

- ⋆ Need to secure computation
- ⋆ Block Ciphers and indistinguishability
- ⋆ Hash Functions and CICO problem

Advanced Protocols

Fully Homomorphic Encryption

Multi-Party Computation

Zero-Knowledge Proofs

Homomorphic Encryption

Definition

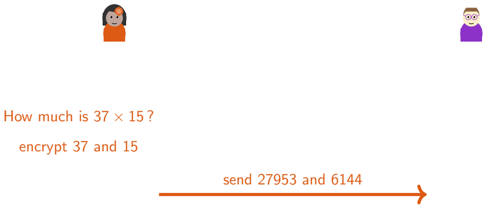
FHE (Fully Homomorphic Encryption) is an encryption scheme with algebraic properties that allow it to perform certain operations on encrypted data without first needing to decrypt it.

How much is 37×15 ?

Homomorphic Encryption

Definition

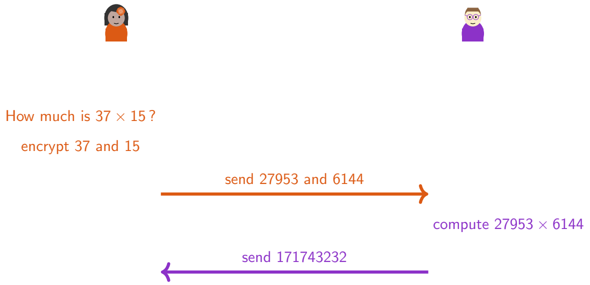
FHE (Fully Homomorphic Encryption) is an encryption scheme with algebraic properties that allow it to perform certain operations on encrypted data without first needing to decrypt it.



Homomorphic Encryption

Definition

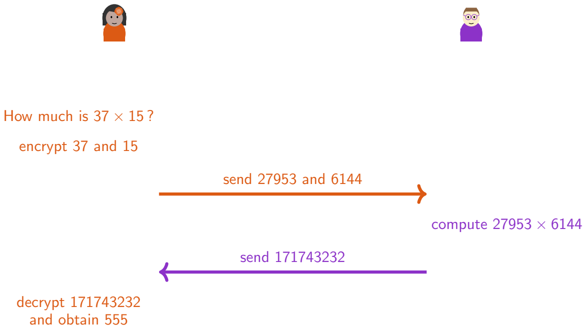
FHE (Fully Homomorphic Encryption) is an encryption scheme with algebraic properties that allow it to perform certain operations on encrypted data without first needing to decrypt it.



Homomorphic Encryption

Definition

FHE (Fully Homomorphic Encryption) is an encryption scheme with algebraic properties that allow it to perform certain operations on encrypted data without first needing to decrypt it.



Multi Party Computation

Definition

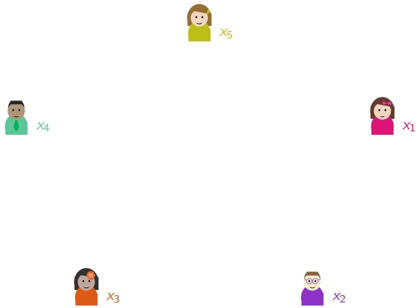
MPC (Multi Party Computation) allows multiple parties to jointly calculate a function on their respective data without revealing it, while ensuring that the result is exact.

Multi Party Computation

Definition

MPC (Multi Party Computation) allows multiple parties to jointly calculate a function on their respective data without revealing it, while ensuring that the result is exact.

How to compute $x_1 + x_2 + x_3 + x_4 + x_5$?

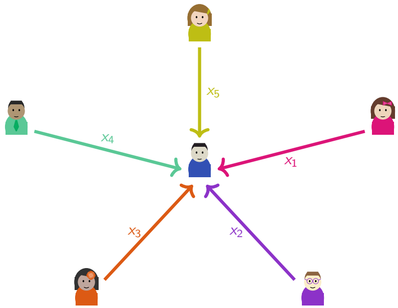


Multi Party Computation

Definition

MPC (Multi Party Computation) allows multiple parties to jointly calculate a function on their respective data without revealing it, while ensuring that the result is exact.

How to compute $x_1 + x_2 + x_3 + x_4 + x_5$?

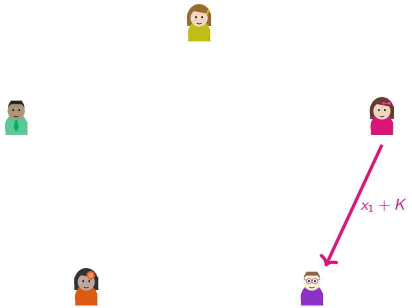


Multi Party Computation

Definition

MPC (Multi Party Computation) allows multiple parties to jointly calculate a function on their respective data without revealing it, while ensuring that the result is exact.

How to compute $x_1 + x_2 + x_3 + x_4 + x_5$?

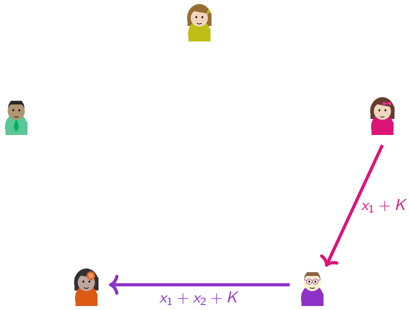


Multi Party Computation

Definition

MPC (Multi Party Computation) allows multiple parties to jointly calculate a function on their respective data without revealing it, while ensuring that the result is exact.

How to compute $x_1 + x_2 + x_3 + x_4 + x_5$?

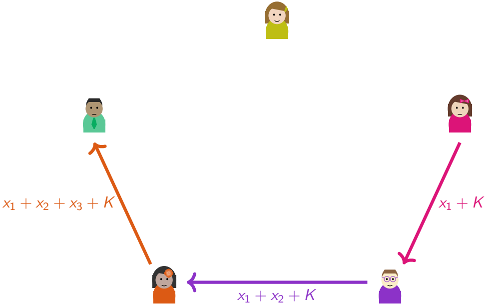


Multi Party Computation

Definition

MPC (Multi Party Computation) allows multiple parties to jointly calculate a function on their respective data without revealing it, while ensuring that the result is exact.

How to compute $x_1 + x_2 + x_3 + x_4 + x_5$?

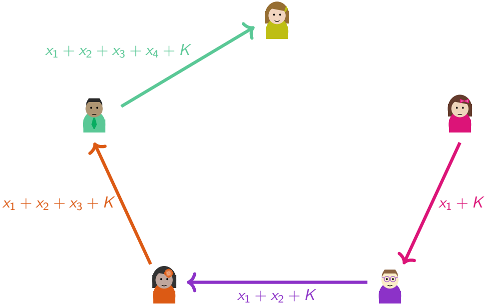


Multi Party Computation

Definition

MPC (Multi Party Computation) allows multiple parties to jointly calculate a function on their respective data without revealing it, while ensuring that the result is exact.

How to compute $x_1 + x_2 + x_3 + x_4 + x_5$?

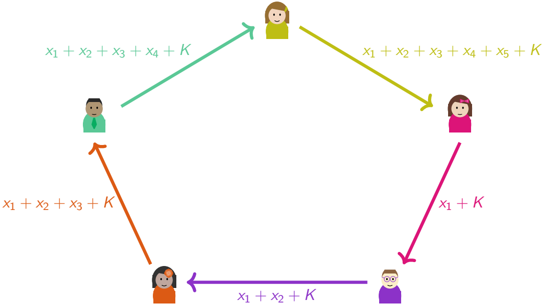


Multi Party Computation

Definition

MPC (Multi Party Computation) allows multiple parties to jointly calculate a function on their respective data without revealing it, while ensuring that the result is exact.

How to compute $x_1 + x_2 + x_3 + x_4 + x_5$?

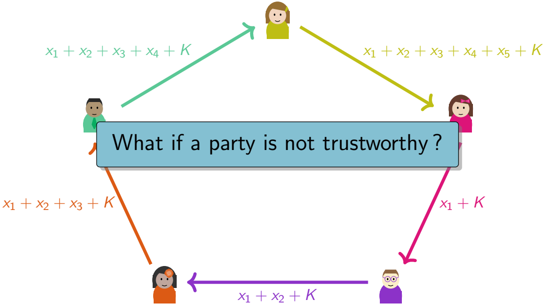


Multi Party Computation

Definition

MPC (Multi Party Computation) allows multiple parties to jointly calculate a function on their respective data without revealing it, while ensuring that the result is exact.

How to compute $x_1 + x_2 + x_3 + x_4 + x_5$?



What if a party is not trustworthy ?

Zero-Knowledge Proofs

Definition

ZK (Zero Knowledge) proofs allow a prover to convince a verifier that a proposition is true without revealing it.

Zero-Knowledge Proofs

Definition

ZK (Zero Knowledge) proofs allow a prover to convince a verifier that a proposition is true without revealing it.

How to proceed ?

- ⋆ A prover $\mathcal{P}$ pretends that some statement **s** is true.
- ⋆ A verifier $\mathcal{V}$ asks questions to the prover $\mathcal{P}$ in order to challenge him.
- ⋆ If the prover $\mathcal{P}$ gives correct answers to any questions, then the verifier $\mathcal{V}$ gains confidence.

Zero-Knowledge Proofs

Definition

ZK (Zero Knowledge) proofs allow a prover to convince a verifier that a proposition is true without revealing it.

How to proceed ?

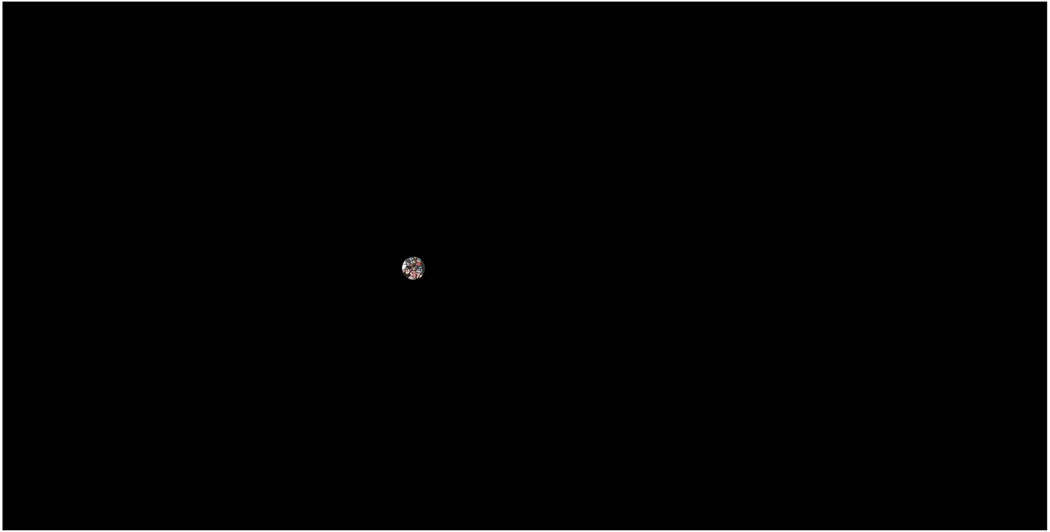
- ⋆ A prover $\mathcal{P}$ pretends that some statement **s** is true.
- ⋆ A verifier $\mathcal{V}$ asks questions to the prover $\mathcal{P}$ in order to challenge him.
- ⋆ If the prover $\mathcal{P}$ gives correct answers to any questions, then the verifier $\mathcal{V}$ gains confidence.

Definition

- ⋆ **Completeness**. If **s** is true, it must exist a way to convince $\mathcal{V}$.
- ⋆ **Soundness**. If **s** is false, there is no way $\mathcal{P}$ can convince $\mathcal{V}$ (except with small probability).
- ⋆ **Zero-knowledge**. If **s** is true, $\mathcal{V}$ learns nothing more than this fact.

Where's Wally ?

Where's Wally ?



Where's Wally ?

Where's Wally ?

Where's Wally ?

Where's Wally ?

Where's Wally ?

Statement **s** :

I know where is Wally !

Properties

⋆ **Completeness**
There is a way to convince $\mathcal{V}$.

⋆ **Soundness**
If **s** is false, $\mathcal{P}$ can convince $\mathcal{V}$ with probability

0 .

⋆ **Zero-knowledge**
$\mathcal{V}$ learns nothing more than **s**.
He doesn't know the exact position of Wally.

Sudoku

	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

Unsolved Sudoku

Sudoku

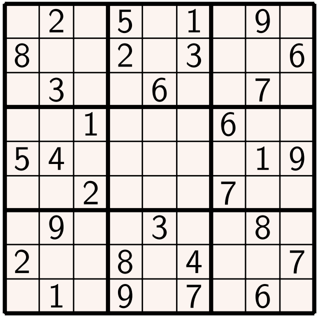
	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

Unsolved Sudoku

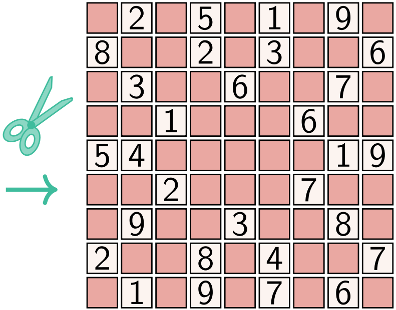
4	2	*6*	5	*7*	1	*3*	9	*8*
8	*5*	*7*	2	*9*	3	*1*	*4*	6
1	3	*9*	*4*	6	*8*	*2*	7	*5*
9	*7*	1	*3*	*8*	*5*	6	*2*	*4*
5	4	*3*	*7*	*2*	*6*	*8*	1	9
6	*8*	2	*1*	*4*	*9*	7	*5*	*3*
7	9	*4*	*6*	3	*2*	*5*	8	*1*
2	*6*	*5*	8	*1*	4	*9*	*3*	7
3	1	*8*	9	*5*	7	*4*	6	*2*

Solved Sudoku

Sudoku



Unsolved Sudoku

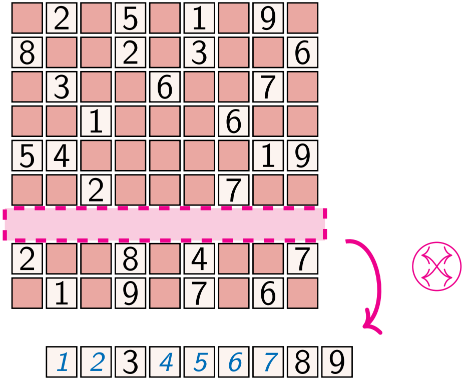


Grid cutting

Sudoku

	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

Unsolved Sudoku

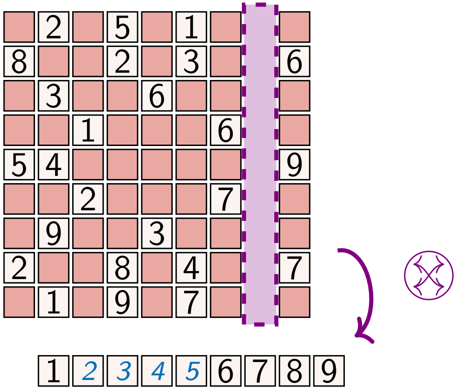


Rows checking

Sudoku

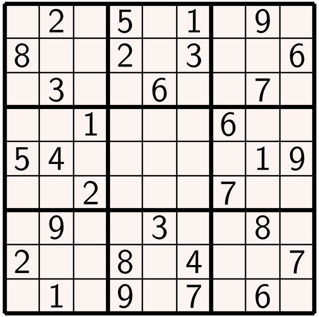
	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

Unsolved Sudoku

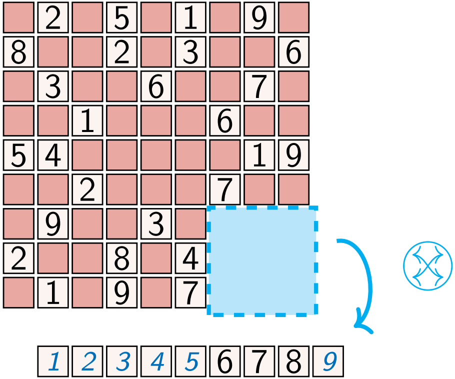


Columns checking

Sudoku



Unsolved Sudoku



Squares checking

Sudoku

Statement **s** :

I know the solution of the grid !

Properties

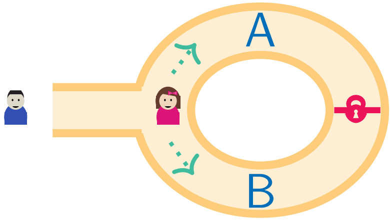
⋆ **Completeness**
There is a way to convince $\mathcal{V}$.

⋆ **Soundness**
If **s** is false, $\mathcal{P}$ can convince $\mathcal{V}$ with probability

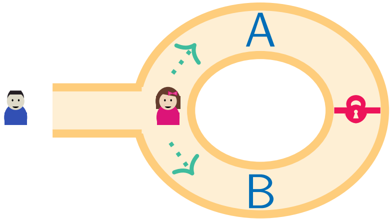
$$\frac{1}{(9-n)!} \quad \text{for } n \text{ known squares.}$$

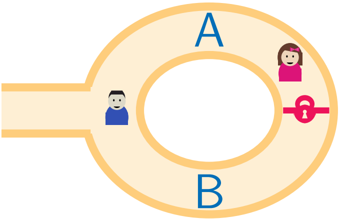
⋆ **Zero-knowledge**
$\mathcal{V}$ learns nothing more than **s**.
He doesn't know the exact position of each number in the grid.

Ali-Baba cave

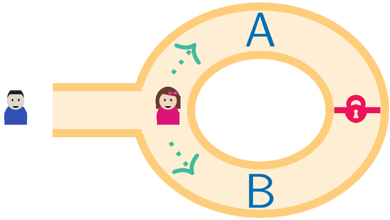


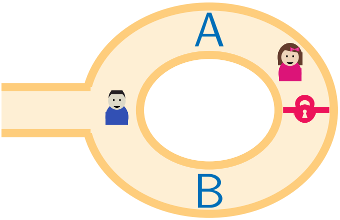
Ali-Baba cave

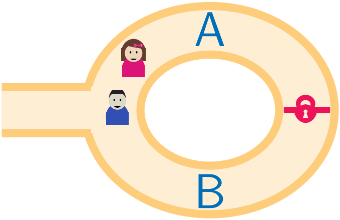




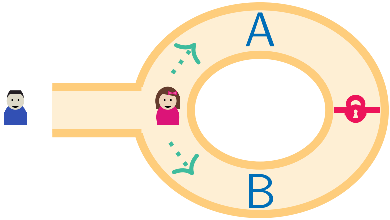
Ali-Baba cave



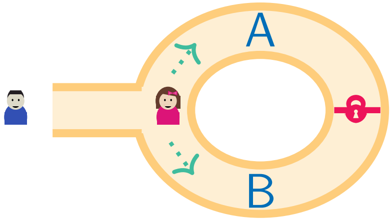


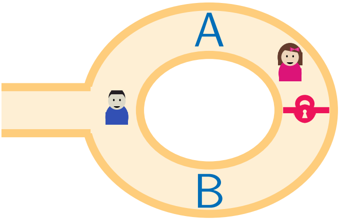


Ali-Baba cave

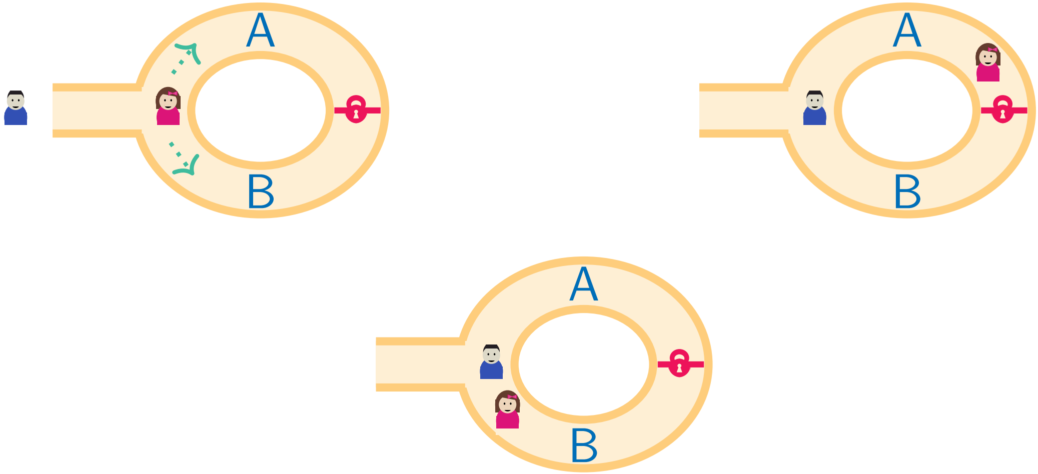


Ali-Baba cave





Ali-Baba cave



Ali-Baba cave

Statement **s** :

I know the password of the door !

Properties

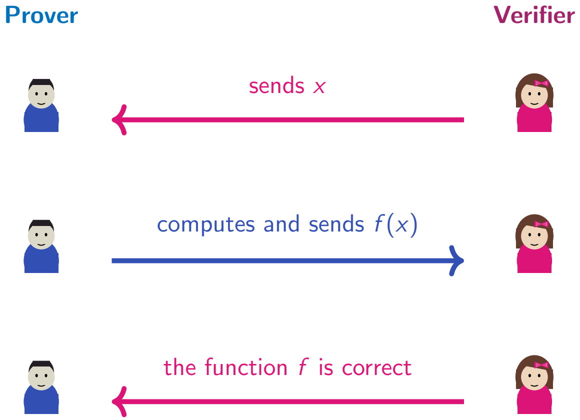
- ⋆ **Completeness**
  There is a way to convince $\mathcal{V}$.
- ⋆ **Soundness**
  If **s** is false, $\mathcal{P}$ can convince $\mathcal{V}$ with probability

$$\frac{1}{2^n} \quad \text{after } n \text{ runs.}$$

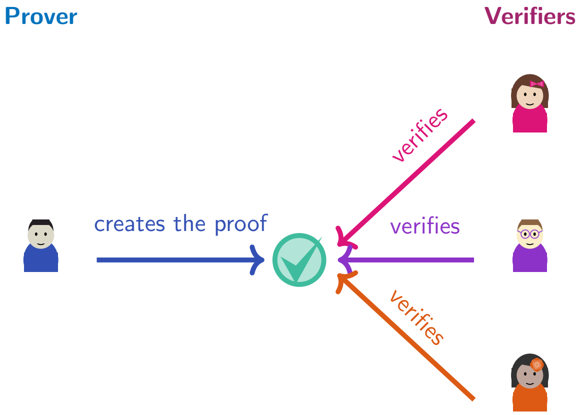
- ⋆ **Zero-knowledge**
  $\mathcal{V}$ learns nothing more than **s**.

He doesn't know the password.

Interactive Proofs (IZKP)



Non-interactive Proofs (NIZKP)



QUIZ ! !

- ⋆ Does MPC work if some parties are not trustworthy ?
- ⋆ What does the acronym ZKP stand for ?
- ⋆ Is the Sudoku an example of an IZKP ? A NIZKP ? Both ?
- ⋆ Ali Baba's cave is an example of a NIZKP. True or False ?

Take-away

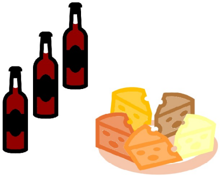
What do we mean by Advanced Protocols ?

- ⋆ FHE : performing operations on cipher text directly
- ⋆ MPC : jointly evaluating a function
- ⋆ ZKP : proving what we can not reveal

New AOPs

Differences with traditional primitives

New challenges



AOP

« Appellation d'origine protégée »

Camembert de Normandie

AOP

≪ **Arithmetization-Oriented Primitives** ≫

Camembert de Normandie

A new environment

Traditional case

Operations based on logical gates or CPU instructions.

$$\mathbb{F}_2^n, \text{ with } n \simeq 4, 8$$

Example

Field of AES

$$\mathbb{F}_2^n, \text{ where } n = 8$$

$$(0, 0, 0, 0, 0, 0, 0, 0),$$
$$(0, 0, 0, 0, 0, 0, 0, 1),$$
$$\ldots$$
$$(1, 1, 1, 1, 1, 1, 1, 1)$$

A new environment

Traditional case

Operations based on logical gates or CPU instructions.

$$\mathbb{F}_2^n, \text{ with } n \simeq 4, 8$$

Arithmetization-Oriented

Operations based on large finite-field arithmetic.

$$\mathbb{F}_q, \text{ with } q \in \{2^n, p\}, p \simeq 2^n, n \geq 32$$

Example

Field of AES

$$\mathbb{F}_2^n, \text{ where } n = 8$$

$$(0,0,0,0,0,0,0,0),$$
$$(0,0,0,0,0,0,0,1),$$
$$\ldots$$
$$(1,1,1,1,1,1,1,1)$$

Example

Scalar Field of Curve BLS12-381

$$\mathbb{F}_p, \text{ where}$$

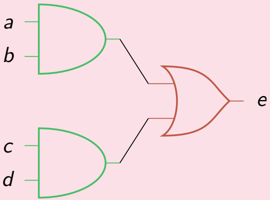
$$p = \texttt{0x73eda753299d7d483339d80809a1d805}$$
$$\texttt{53bda402fffe5bfeffffffff00000001}$$

$$0, 1, 2, \ldots, p-1$$

New operations

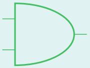
Traditional case

Use of logical gates and CPU instructions.



Example

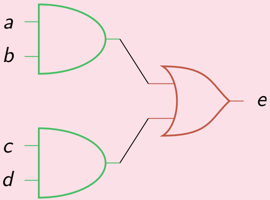
$$(0,1,0)\&(1,1,0) = (0,1,0)$$



New operations

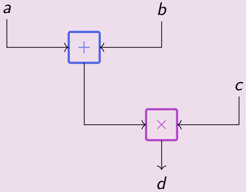
Traditional case

Use of logical gates and CPU instructions.



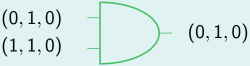
Arithmetization-Oriented

Use of Arithmetic circuit.



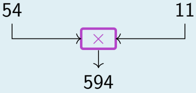
Example

$$(0,1,0)\&(1,1,0) = (0,1,0)$$



Example

$$54 \times 11 = 594$$

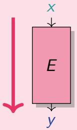


A new metric

Traditional case

Minimize time and memory.

$$y \leftarrow E(x)$$

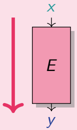


A new metric

Traditional case

Minimize time and memory.

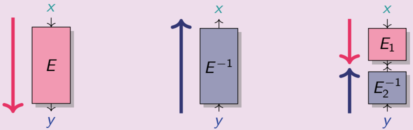
$$y \leftarrow E(x)$$



Arithmetization-Oriented

Minimize the number of multiplications.

$$y \leftarrow E(x) \quad \text{and} \quad y == E(x)$$

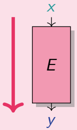


A new metric

Traditional case

Minimize time and memory.

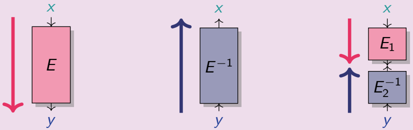
$$y \leftarrow E(x)$$



Arithmetization-Oriented

Minimize the number of multiplications.

$$y \leftarrow E(x) \quad \text{and} \quad y == E(x)$$



Example

Let $E : \mathbb{F}_{11} \rightarrow \mathbb{F}_{11}, x \mapsto x^3$. We have $E^{-1} : \mathbb{F}_{11} \rightarrow \mathbb{F}_{11}, x \mapsto x^7$.

Evaluation : Given $x = 5$, compute $y = E(x)$.

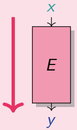
$$y = 4 \text{ (applying } E\text{)}$$

A new metric

Traditional case

Minimize time and memory.

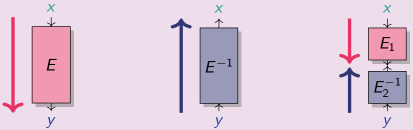
$$y \leftarrow E(x)$$



Arithmetization-Oriented

Minimize the number of multiplications.

$$y \leftarrow E(x) \quad \text{and} \quad y == E(x)$$



Example

Let $E : \mathbb{F}_{11} \to \mathbb{F}_{11}, x \mapsto x^3$. We have $E^{-1} : \mathbb{F}_{11} \to \mathbb{F}_{11}, x \mapsto x^7$.

Verification : Given $x = 5$ and $y = 4$, check if $y = E(x)$.

$$5^3 = 4 \text{ (applying } E) \qquad \text{or} \qquad 4^7 = 5 \text{ (applying } E^{-1})$$

Take-away

Traditional case

- ⋆ Alphabet :

$$\mathbb{F}_2^n, \text{ with } n \simeq 4, 8$$

- ⋆ Operations :
  Logical gates/CPU instructions
- ⋆ Metric : minimize time and memory for the evaluation
- ⋆ **Decades of Cryptanalysis**

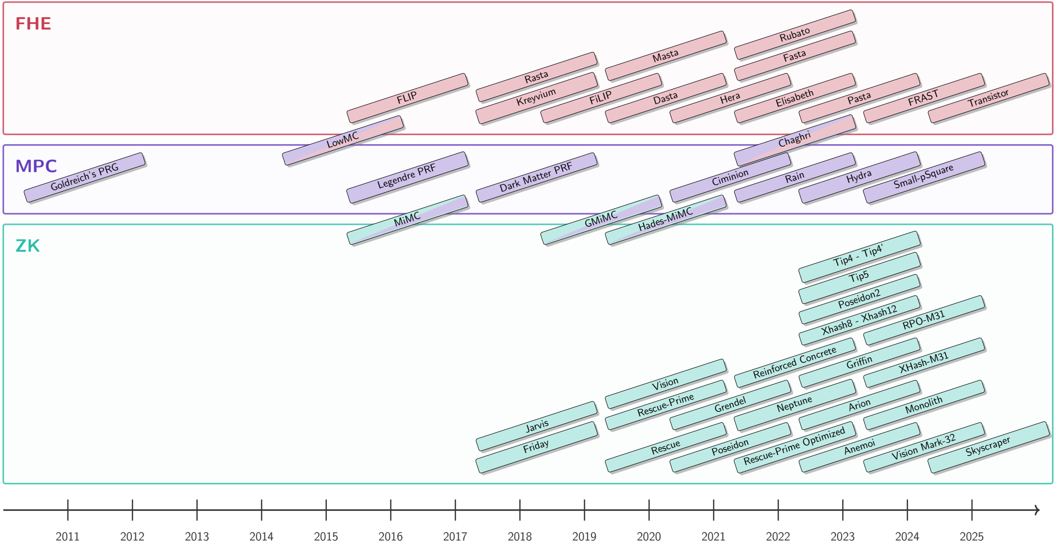
Arithmetization-Oriented

- ⋆ Alphabet :

$$\mathbb{F}_q, \text{ with } q \in \{2^n, p\}, p \simeq 2^n, n \geq 32$$

- ⋆ Operations :
  Large finite-field arithmetic
- ⋆ Metric : minimize the number of multiplications for the verification
- ⋆ ≤ 8 **years of Cryptanalysis**

Primitives


FHE
Rubato
Masta
Fasta
Rasta
FLIP
Kreyvium
FiLIP
Dasta
Hera
Elisabeth
Pasta
FRAST
Transistor
LowMC
Chaghri
MPC
Goldreich's PRG
Legendre PRF
Dark Matter PRF
Ciminion
Rain
Hydra
Small-pSquare
MiMC
GMiMC
Hades-MiMC
ZK
Tip4 - Tip4'
Tip5
Poseidon2
Xhash8 - Xhash12
RPO-M31
Reinforced Concrete
Griffin
XHash-M31
Vision
Rescue-Prime
Grendel
Neptune
Arion
Monolith
Jarvis
Friday
Rescue
Poseidon
Rescue-Prime Optimized
Anemoi
Vision Mark-32
Skyscraper
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025

QUIZ ! !

- ⋆ In French, what is the meaning of AOP ?
- ⋆ For this talk, what is the meaning of AOPs ?
- ⋆ AOPs are symmetric or asymmetric primitives ?
- ⋆ Can AES be used for advanced protocols ?


Take-away

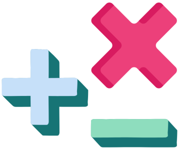
Challenges for AOPs

- ⋆ New context, new operations, new environment
- ⋆ Adapting traditional techniques...
- ⋆ ... or creating new ones ?

Computing constraints

R1CS : an easy example

Other proof systems : Plonk, AIR, ...



Computing constraints

What does "efficient" mean for Zero-Knowledge Proofs ?

Computing constraints

What does "efficient" mean for Zero-Knowledge Proofs ?

"It depends"

Computing constraints

What does "efficient" mean for Zero-Knowledge Proofs ?

"It depends"

Example

R1CS (Rank-1 Constraint System) : minimizing the number of multiplications

$$y = (ax + b)^3(cx + d) + ex$$

$t_0 = a \cdot x$
$t_1 = t_0 + b$
$t_2 = t_1 \times t_1$

$t_3 = t_2 \times t_1$
$t_4 = c \cdot x$
$t_5 = t_4 + d$

$t_6 = t_3 \times t_5$
$t_7 = e \cdot x$
$t_8 = t_6 + t_7$

Computing constraints

What does "efficient" mean for Zero-Knowledge Proofs ?

"It depends"

Example

R1CS (Rank-1 Constraint System) : minimizing the number of multiplications

$$y = (ax + b)^3(cx + d) + ex$$

$t_0 = a \cdot x$

$t_1 = t_0 + b$

$t_2 = t_1 \times t_1$

$t_3 = t_2 \times t_1$

$t_4 = c \cdot x$

$t_5 = t_4 + d$

$t_6 = t_3 \times t_5$

$t_7 = e \cdot x$

$t_8 = t_6 + t_7$

3 constraints

Computing constraints

What does "efficient" mean for Zero-Knowledge Proofs ?

"It depends"

Example

R1CS (Rank-1 Constraint System) : minimizing the number of multiplications

$$y = x^7$$

Computing constraints

What does "efficient" mean for Zero-Knowledge Proofs ?

"It depends"

Example

R1CS (Rank-1 Constraint System) : minimizing the number of multiplications

$$y = x^7$$

$t_0 = x \times x$

$t_1 = t_0 \times t_0$

$t_2 = t_1 \times t_0$

$t_3 = t_2 \times x$

Computing constraints

What does "efficient" mean for Zero-Knowledge Proofs ?

"It depends"

Example

R1CS (Rank-1 Constraint System) : minimizing the number of multiplications

$$y = x^7$$

$$t_0 = x \times x \qquad t_2 = t_1 \times t_0$$

$$t_1 = t_0 \times t_0 \qquad t_3 = t_2 \times x$$

4 constraints

Expand or factorise ?

Factorised expression

$$z = (x+y)^3 + 1$$

$t_0 = x + y$ $t_1 = t_0 \times t_0$ $t_2 = t_1 \times t_0$ $t_3 = t_2 + 1$

Expand or factorise ?

Factorised expression

$$z = (x + y)^3 + 1$$

$t_0 = x + y$ $\quad$ $t_1 = t_0 \times t_0$ $\quad$ $t_2 = t_1 \times t_0$ $\quad$ $t_3 = t_2 + 1$

2 constraints

Expand or factorise ?

Factorised expression

$$z = (x + y)^3 + 1$$

$t_0 = x + y$ $\quad t_1 = t_0 \times t_0$ $\quad t_2 = t_1 \times t_0$ $\quad t_3 = t_2 + 1$

2 constraints

Expanded expression

$$z = x^3 + 3x^2y + 3xy^2 + y^3$$

$t_0 = x \times x$	$t_3 = 3 \cdot t_2$	$t_6 = t_4 \times x$	$t_9 = t_5 + t_7$
$t_1 = t_0 \times x$	$t_4 = y \times y$	$t_7 = 3 \cdot t_6$	$t_{10} = t_8 + t_9$
$t_2 = t_0 \times y$	$t_5 = t_4 \times y$	$t_8 = t_1 + t_3$	$t_{11} = t_{10} + 1$

Expand or factorise ?

Factorised expression

$$z = (x + y)^3 + 1$$

$t_0 = x + y$ $\quad$ $t_1 = t_0 \times t_0$ $\quad$ $t_2 = t_1 \times t_0$ $\quad$ $t_3 = t_2 + 1$

2 constraints

Expanded expression

$$z = x^3 + 3x^2y + 3xy^2 + y^3$$

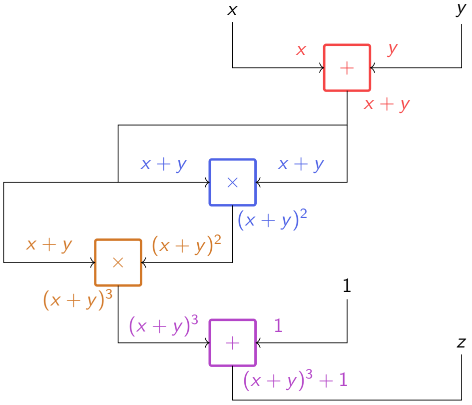
$t_0 = x \times x$	$t_3 = 3 \cdot t_2$	$t_6 = t_4 \times x$	$t_9 = t_5 + t_7$
$t_1 = t_0 \times x$	$t_4 = y \times y$	$t_7 = 3 \cdot t_6$	$t_{10} = t_8 + t_9$
$t_2 = t_0 \times y$	$t_5 = t_4 \times y$	$t_8 = t_1 + t_3$	$t_{11} = t_{10} + 1$

6 constraints

A circuit representation

Factorised expression

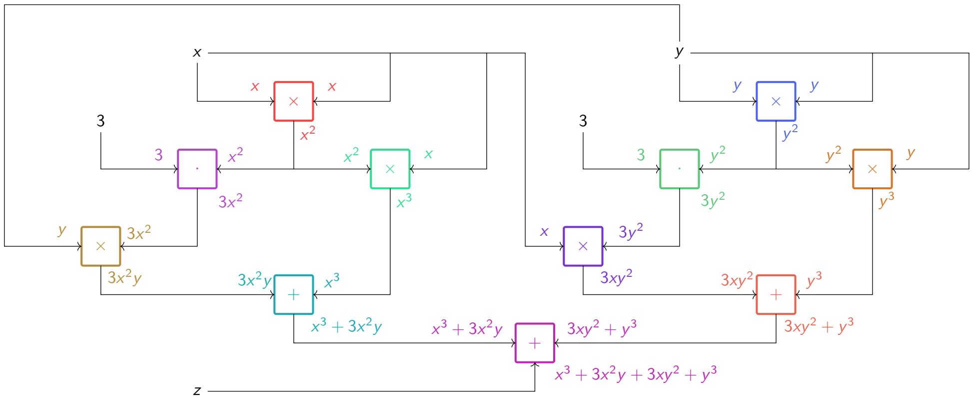
$$z = (x + y)^3 + 1$$



A circuit representation

Expanded expression

$$z = x^3 + 3x^2y + 3xy^2 + y^3$$



Expand or factorise ?

Factorised expression

$$\begin{cases} w &= (2x+y)^3 + x^3 \\ z &= (x+2y)^3 + y^3 \end{cases}$$

$t_0 = 2 \cdot x$
$t_1 = t_0 + y$
$t_2 = t_1 \times t_1$

$t_3 = t_2 \times t_1$
$t_4 = 2 \cdot y$
$t_5 = t_4 + x$

$t_6 = t_5 \times t_5$
$t_7 = t_6 \times t_5$
$t_8 = x \times x$

$t_9 = t_8 \times x$
$t_{10} = y \times y$
$t_{11} = t_{10} \times y$

$t_{12} = t_3 + t_9$
$t_{13} = t_7 + t_{11}$

Expand or factorise ?

Factorised expression

$$\begin{cases} w &= (2x+y)^3 + x^3 \\ z &= (x+2y)^3 + y^3 \end{cases}$$

$t_0 = 2 \cdot x$
$t_1 = t_0 + y$
$t_2 = t_1 \times t_1$

$t_3 = t_2 \times t_1$
$t_4 = 2 \cdot y$
$t_5 = t_4 + x$

$t_6 = t_5 \times t_5$
$t_7 = t_6 \times t_5$
$t_8 = x \times x$

$t_9 = t_8 \times x$
$t_{10} = y \times y$
$t_{11} = t_{10} \times y$

$t_{12} = t_3 + t_9$
$t_{13} = t_7 + t_{11}$

8 constraints

Expand or factorise ?

Factorised expression

$$\begin{cases} w &= (2x+y)^3 + x^3 \\ z &= (x+2y)^3 + y^3 \end{cases}$$

$t_0 = 2 \cdot x$ | $t_1 = t_0 + y$ | $t_2 = t_1 \times t_1$

$t_3 = t_2 \times t_1$ | $t_4 = 2 \cdot y$ | $t_5 = t_4 + x$

$t_6 = t_5 \times t_5$ | $t_7 = t_6 \times t_5$ | $t_8 = x \times x$

$t_9 = t_8 \times x$ | $t_{10} = y \times y$ | $t_{11} = t_{10} \times y$

$t_{12} = t_3 + t_9$ | $t_{13} = t_7 + t_{11}$

8 constraints

Expanded expression

$$\begin{cases} w &= 9x^3 + 12x^2y + 6xy^2 + y^3 \\ z &= x^3 + 6x^2y + 12xy^2 + 9y^3 \end{cases}$$

$t_0 = x \times x$ | $t_1 = t_0 \times x$ | $t_2 = y \times y$

$t_3 = t_2 \times y$ | $t_4 = t_0 \times y$ | $t_5 = t_2 \times x$

$t_6 = 9 \cdot t_1$ | $t_7 = 12 \cdot t_4$ | $t_8 = 6 \cdot t_5$

$t_9 = 6 \cdot t_4$ | $t_{10} = 12 \cdot t_5$ | $t_{11} = 9 \cdot t_3$

$t_{12} = t_6 + t_7$ | $t_{13} = t_{12} + t_8$ | $t_{14} = t_{13} + t_3$

$t_{15} = t_1 + t_9$ | $t_{16} = t_{15} + t_{10}$ | $t_{17} = t_{16} + t_{11}$

Expand or factorise ?

Factorised expression

$$\begin{cases} w &= (2x+y)^3 + x^3 \\ z &= (x+2y)^3 + y^3 \end{cases}$$

$t_0 = 2 \cdot x$ $t_1 = t_0 + y$ $t_2 = t_1 \times t_1$

$t_3 = t_2 \times t_1$ $t_4 = 2 \cdot y$ $t_5 = t_4 + x$

$t_6 = t_5 \times t_5$ $t_7 = t_6 \times t_5$ $t_8 = x \times x$

$t_9 = t_8 \times x$ $t_{10} = y \times y$ $t_{11} = t_{10} \times y$

$t_{12} = t_3 + t_9$ $t_{13} = t_7 + t_{11}$

8 constraints

Expanded expression

$$\begin{cases} w &= 9x^3 + 12x^2y + 6xy^2 + y^3 \\ z &= x^3 + 6x^2y + 12xy^2 + 9y^3 \end{cases}$$

$t_0 = x \times x$ $t_1 = t_0 \times x$ $t_2 = y \times y$

$t_3 = t_2 \times y$ $t_4 = t_0 \times y$ $t_5 = t_2 \times x$

$t_6 = 9 \cdot t_1$ $t_7 = 12 \cdot t_4$ $t_8 = 6 \cdot t_5$

$t_9 = 6 \cdot t_4$ $t_{10} = 12 \cdot t_5$ $t_{11} = 9 \cdot t_3$

$t_{12} = t_6 + t_7$ $t_{13} = t_{12} + t_8$ $t_{14} = t_{13} + t_3$

$t_{15} = t_1 + t_9$ $t_{16} = t_{15} + t_{10}$ $t_{17} = t_{16} + t_{11}$

6 constraints

Plonk Constraints

Additions also have a cost !

$$z = (x + y)^3 + 1$$

R1CS

$t_0 = x + y$ $\quad t_1 = t_0 \times t_0$ $\quad t_2 = t_1 \times t_0$ $\quad t_3 = t_2 + 1$

Plonk Constraints

Additions also have a cost !

$$z = (x + y)^3 + 1$$

R1CS

$t_0 = x + y$ $\quad t_1 = t_0 \times t_0$ $\quad t_2 = t_1 \times t_0$ $\quad t_3 = t_2 + 1$

2 constraints

Plonk Constraints

Additions also have a cost !

$$z = (x + y)^3 + 1$$

R1CS

$t_0 = x + y$ $t_1 = t_0 \times t_0$ $t_2 = t_1 \times t_0$ $t_3 = t_2 + 1$

2 constraints

Plonk

$t_0 = x + y$ $t_1 = t_0 \times t_0$ $t_2 = t_1 \times t_0$ $t_3 = t_2 + 1$

Plonk Constraints

Additions also have a cost !

$$z = (x + y)^3 + 1$$

R1CS

$t_0 = x + y$ $t_1 = t_0 \times t_0$ $t_2 = t_1 \times t_0$ $t_3 = t_2 + 1$

2 constraints

Plonk

$t_0 = x + y$ $t_1 = t_0 \times t_0$ $t_2 = t_1 \times t_0$ $t_3 = t_2 + 1$

3 constraints

Plonk Constraints

Additions also have a cost !

$$z = (x + y)^3 + 1$$

R1CS

$t_0 = x + y$ $t_1 = t_0 \times t_0$ $t_2 = t_1 \times t_0$ $t_3 = t_2 + 1$

2 constraints

Plonk

$t_0 = x + y$ $t_1 = t_0 \times t_0$ $t_2 = t_1 \times t_0$ $t_3 = t_2 + 1$

3 constraints

But it's more complicated than that.... (customs gates)

AIR Constraints

Example : Computing the terms of Fibonacci sequence

Computing the n-th term
for a given n

$a = 1$

$b = 0$

For i **from** 0 **to** $n - 1$

  $a = a + b$

  $b = a$

Return a

AIR Constraints

Example : Computing the terms of Fibonacci sequence

Computing the n-th term for a given n

$a = 1$

$b = 0$

For i **from** 0 **to** $n-1$

  $a = a + b$

  $b = a$

Return a

i	a_i	b_i
0	1	0
1	1	1
2	2	1
3	3	2
⋮	⋮	⋮
$n-1$	a_{n-1}	b_{n-1}

Execution Trace

AIR Constraints

Example : Computing the terms of Fibonacci sequence

Computing the n-th term for a given n

```
a = 1
b = 0
For i from 0 to n − 1
    a = a + b
    b = a
Return a
```

i	a_i	b_i
0	1	0
1	1	1
2	2	1
3	3	2
$\vdots$	$\vdots$	$\vdots$
$n-1$	a_{n-1}	b_{n-1}

Execution Trace

Intermediate verification of lines 3 and 4 ($i = 2$ and $i = 3$)

$$a_3 = 3 = 2 + 1 = a_2 + b_2 \quad \text{and} \quad b_3 = 2 = a_2 \ .$$

AIR Constraints

Example : Computing the terms of Fibonacci sequence

Computing the n-th term for a given n

$a = 1$
$b = 0$
For i **from** 0 **to** $n-1$
  $a = a + b$
  $b = a$
Return a

i	a_i	b_i
0	1	0
1	1	1
2	2	1
3	3	2
$\vdots$	$\vdots$	$\vdots$
$n-1$	a_{n-1}	b_{n-1}

Execution Trace

System of constraints

$$\begin{cases} a_0 = 1 & \text{at 1st line,} \\ b_0 = 0 & \text{at 1st line,} \\ a_{i+1} = a_i + b_i & \text{for } 0 \leq i \leq n-2 \text{ ,} \\ b_{i+1} = a_i & \text{for } 0 \leq i \leq n-2 \text{ ,} \\ a_{n-1} = Fib(n-1) & \text{at line } n-1 \text{ .} \end{cases}$$

Intermediate verification of lines 3 and 4 ($i = 2$ and $i = 3$)

$$a_3 = 3 = 2 + 1 = a_2 + b_2 \quad \text{and} \quad b_3 = 2 = a_2 \ .$$

QUIZ ! !

- ⋆ Scalar multiplication is a type of R1CS constraint. True or False ?
- ⋆ Additions matter when computing Plonk constraints. True or False ?
- ⋆ Custom gates reduce AIR constraints. True or False ?
- ⋆ How many R1CS constraints are needed to verify $y = 3 \cdot x + 1$?
- ⋆ How many R1CS constraints are needed to verify $y = 3 \cdot x^2 + x$?
- ⋆ How to obtain the fewest R1CS constraints to verify $y = (x + 1)^2$?

Take-away

How can we minimise the number of constraints ?

- ⋆ It depends on the proof system
- ⋆ Reduce the number of multiplicative gates (often)
- ⋆ Factorise or expand expressions ?
- ⋆ Custom gates

Conclusions

Short summary

⋆ Advanced protocols (FHE, MPC, ZK) are emerging

⋆ AOPs : symmetric primitives for this new context

Conclusions

Short summary

- ⋆ Advanced protocols (FHE, MPC, ZK) are emerging
- ⋆ AOPs : symmetric primitives for this new context

Next lecture

- ⋆ Design aspects of AOPs, how to classify them ?
- ⋆ Cryptanalysis aspects, are AOPs secure ?

Conclusions

Short summary

- ⋆ Advanced protocols (FHE, MPC, ZK) are emerging
- ⋆ AOPs : symmetric primitives for this new context

Next lecture

- ⋆ Design aspects of AOPs, how to classify them ?
- ⋆ Cryptanalysis aspects, are AOPs secure ?

Break time !