Trusting Without Seeing

How Cryptography Makes It Possible

Clémence Bouvier

Université de Lorraine, CNRS, Inria, LORIA

ULB, Brussels, Belgium
February 19th, 2026

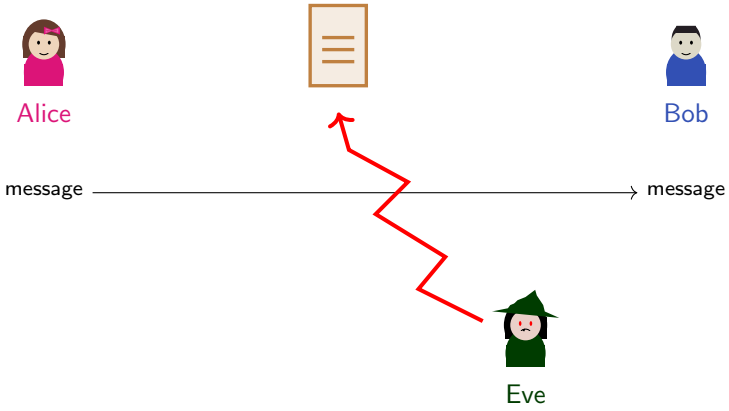
Why do we need cryptography ?

Alice

Bob

message ⟶ message

Why do we need cryptography ?

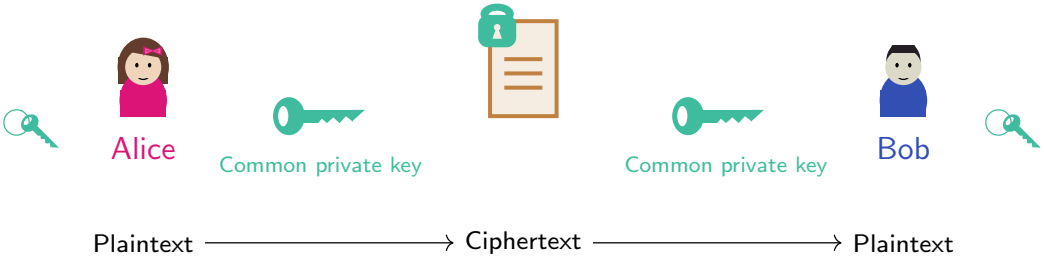


Cryptology

Science of secret messages (cryptography + cryptanalysis)

Symmetric Cryptography

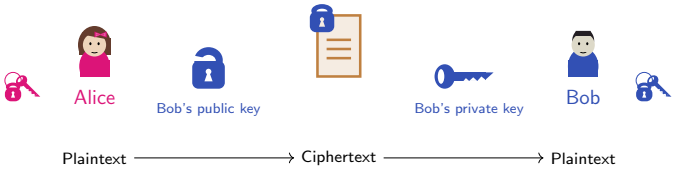
Example : AES



Asymmetric Cryptography

Example : RSA

Hybrid Cryptography



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.

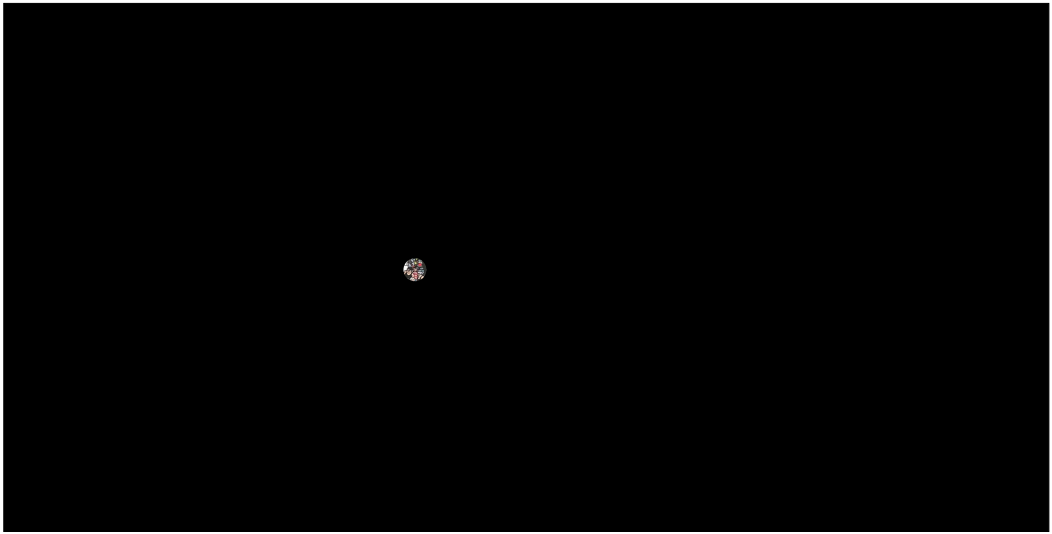
Secure a computation

Zero-Knowledge Proofs

Introduction with toy examples

Where's Wally ?

Where's Wally ?

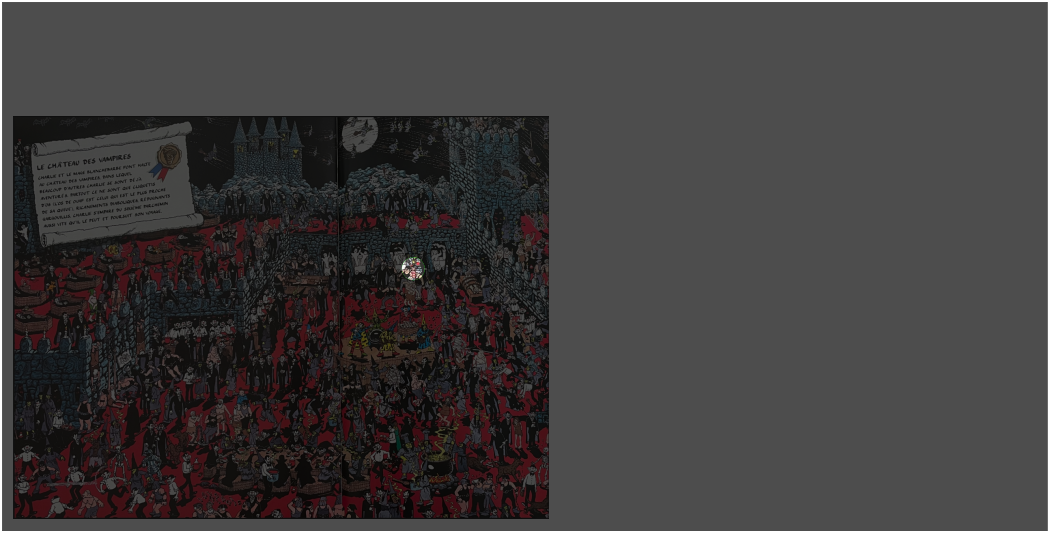


Where's Wally ?

Where's Wally ?

Where's Wally ?

Where's 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

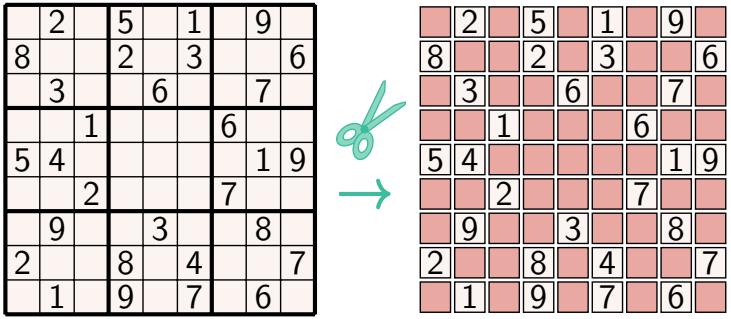
	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

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*

Unsolved Sudoku

Solved Sudoku

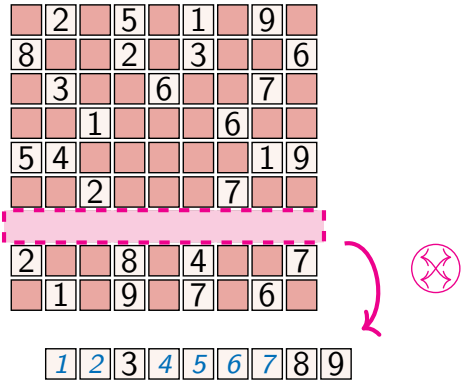
Sudoku



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

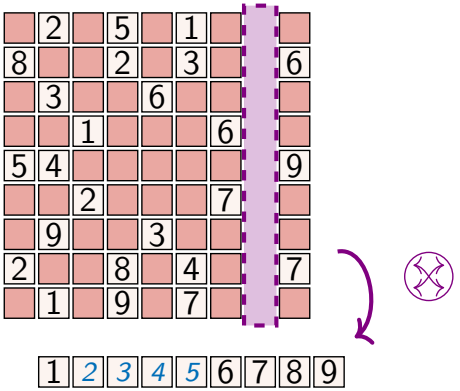


Rows checking

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

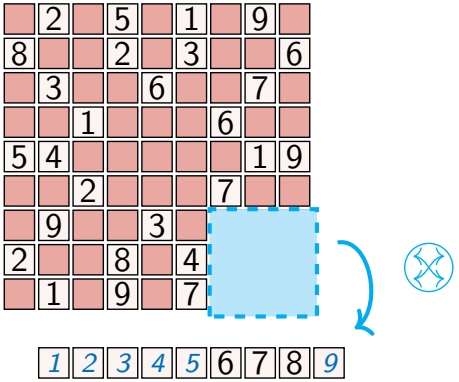


Columns checking

Sudoku

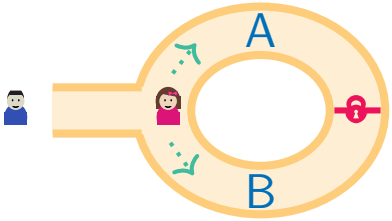
	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

Unsolved Sudoku

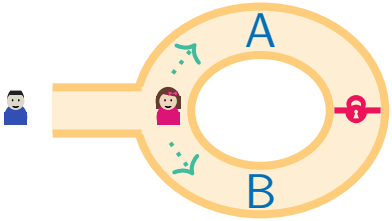


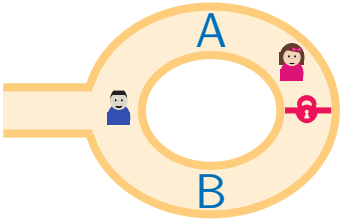
Squares checking

Ali-Baba cave

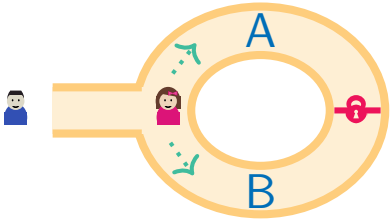


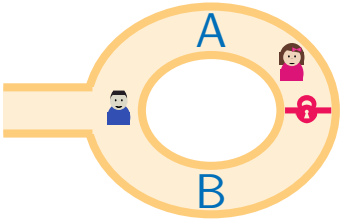
Ali-Baba cave

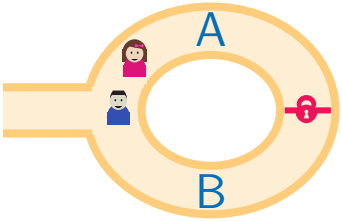




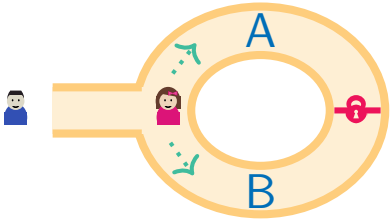
Ali-Baba cave



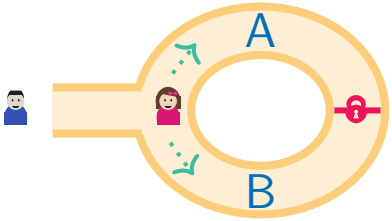


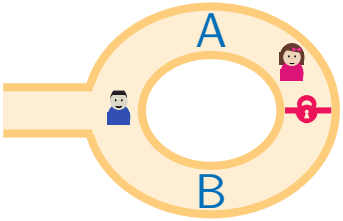


Ali-Baba cave

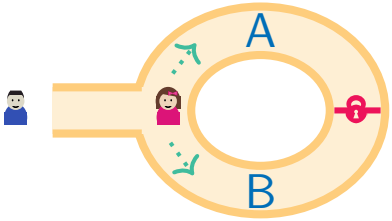


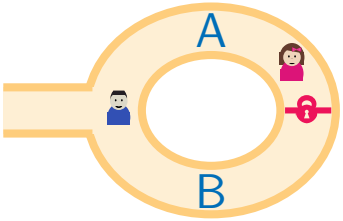
Ali-Baba cave

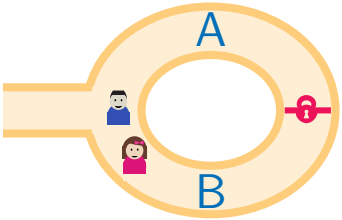




Ali-Baba cave







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.

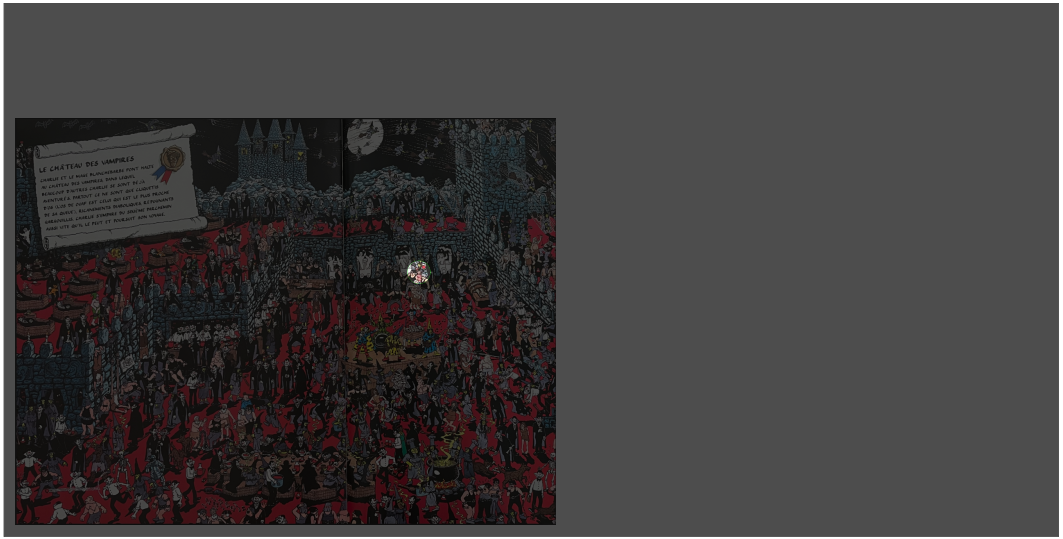
How to proceed ?

- ⋆ 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 ?

Statement **s** :

I know where is Wally !

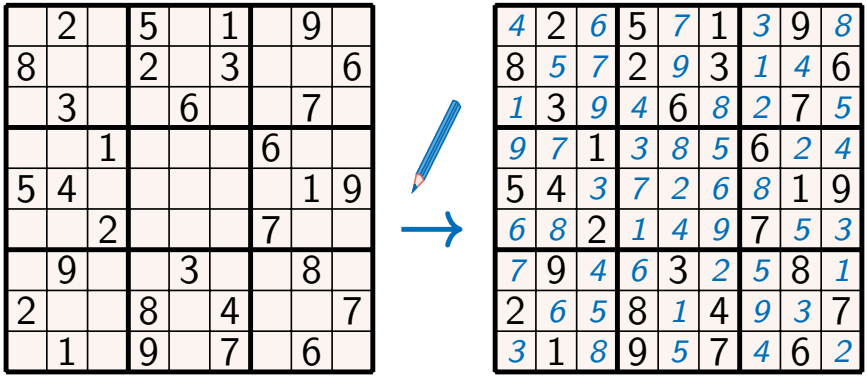
Properties

- ⋆ **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



Sudoku

Statement **s** :

I know the solution of the grid !

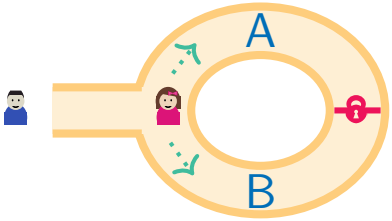
Properties

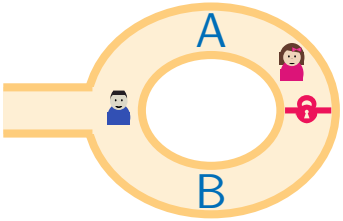
- ⋆ **Completeness**
  There is a way to convince $\mathcal{V}$.
- ⋆ **Soundness**
  If **s** is false, $\mathcal{P}$ can convince $\mathcal{V}$ with probability

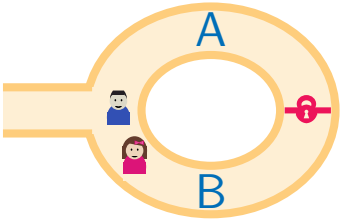
$$\frac{1}{(9-n)!} \quad \text{for } n \text{ known squares.}$$

- ⋆ **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

Statement **s** :

I know the password of the door !

Properties

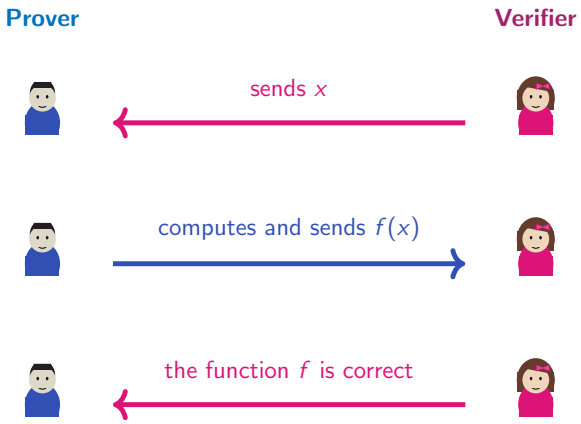
- ⋆ **Completeness**
  There is a way to convince $\mathcal{V}$.
- ⋆ **Soundness**
  If **s** is false, $\mathcal{P}$ can convince $\mathcal{V}$ with probability

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

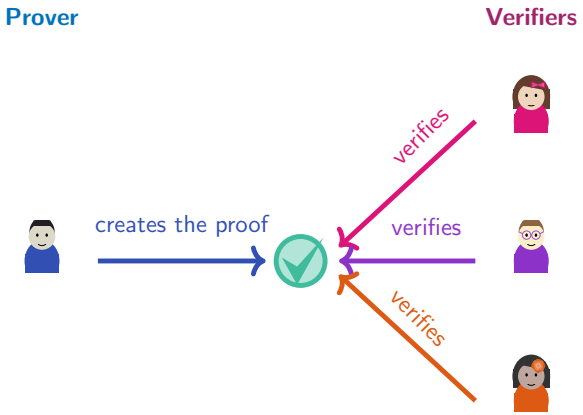
- ⋆ **Zero-knowledge**
  $\mathcal{V}$ learns nothing more than **s**.

He doesn't know the password.

Interactive Proofs (IZKP)



Non-interactive Proofs (NIZKP)



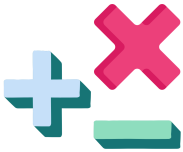
Take-away

What are ZKPs ?

- ⋆ Zero-Knowledge Proofs : proving what we can not reveal
- ⋆ Various toy examples : Wally, Sudoku, Ali Baba
- ⋆ IZKP and NIZKP

Computing constraints

Examples with R1CS and Plonk



Constraints

Constraints

Mathematical rules that must be satisfied for the proof to be valid.

Constraints

Constraints

Mathematical rules that must be satisfied for the proof to be valid.

Example : **Sudoku**

The condition

$$x \in \{1, 2, \dots 9\}$$

can be written as

$$(x-1)(x-2)\dots(x-9) = 0 \ .$$

List of constraints :

$$t_0 = x - 1$$
$$t_1 = x - 2$$
$$t_2 = t_0 \times t_1$$
$$\dots$$

Constraints

Constraints

Mathematical rules that must be satisfied for the proof to be valid.

Circuit

Network of basic arithmetic operations representing a calculation.

Example : **Sudoku**

The condition

$$x \in \{1, 2, \dots 9\}$$

can be written as

$$(x-1)(x-2)\dots(x-9) = 0 \ .$$

List of constraints :

$$t_0 = x - 1$$
$$t_1 = x - 2$$
$$t_2 = t_0 \times t_1$$
$$\dots$$

Constraints

Constraints

Mathematical rules that must be satisfied for the proof to be valid.

Circuit

Network of basic arithmetic operations representing a calculation.

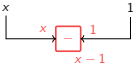
Example : **Sudoku**

The condition

$$x \in \{1, 2, \dots 9\}$$

can be written as

$$(x-1)(x-2)\dots(x-9) = 0 \ .$$



List of constraints :

$$t_0 = x - 1$$
$$t_1 = x - 2$$
$$t_2 = t_0 \times t_1$$
$$\dots$$

Constraints

Constraints

Mathematical rules that must be satisfied for the proof to be valid.

Circuit

Network of basic arithmetic operations representing a calculation.

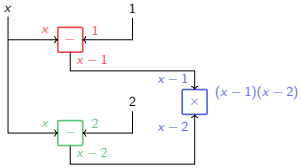
Example : **Sudoku**

The condition

$$x \in \{1, 2, \dots 9\}$$

can be written as

$$(x-1)(x-2)\dots(x-9) = 0 \ .$$



List of constraints :

$$t_0 = x - 1$$
$$t_1 = x - 2$$
$$t_2 = t_0 \times t_1$$
$$\dots$$

Constraints

Constraints

Mathematical rules that must be satisfied for the proof to be valid.

Circuit

Network of basic arithmetic operations representing a calculation.

Example : **Sudoku**

The condition

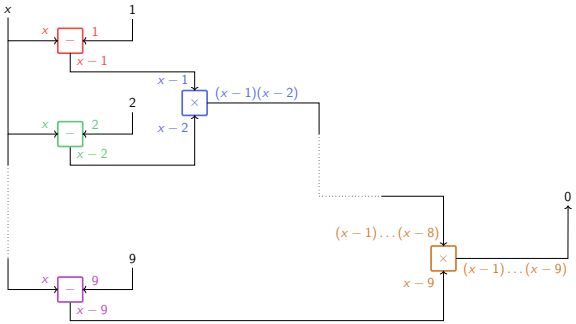
$$x \in \{1, 2, \dots 9\}$$

can be written as

$$(x-1)(x-2)\dots(x-9) = 0 \ .$$

List of constraints :

$$t_0 = x - 1$$
$$t_1 = x - 2$$
$$t_2 = t_0 \times t_1$$
$$\dots$$



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

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

$$t_0 = a \cdot x \qquad t_3 = t_2 \times t_1 \qquad t_6 = t_3 \times t_5$$
$$t_1 = t_0 + b \qquad t_4 = c \cdot x \qquad t_7 = e \cdot x$$
$$t_2 = t_1 \times t_1 \qquad t_5 = t_4 + d \qquad t_8 = t_6 + t_7$$

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

$$y = x^7$$

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

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

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

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

$t_0 = x \times x$ $\quad t_3 = 3 \cdot t_2$ $\quad t_6 = t_4 \times x$ $\quad t_9 = t_5 + t_7$

$t_1 = t_0 \times x$ $\quad t_4 = y \times y$ $\quad t_7 = 3 \cdot t_6$ $\quad t_{10} = t_8 + t_9$

$t_2 = t_0 \times y$ $\quad t_5 = t_4 \times y$ $\quad t_8 = t_1 + t_3$ $\quad t_{11} = t_{10} + 1$

Expand or factorise ?

Factorised expression

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

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

2 constraints

Expanded expression

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

$t_0 = x \times x$ $t_3 = 3 \cdot t_2$ $t_6 = t_4 \times x$ $t_9 = t_5 + t_7$

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

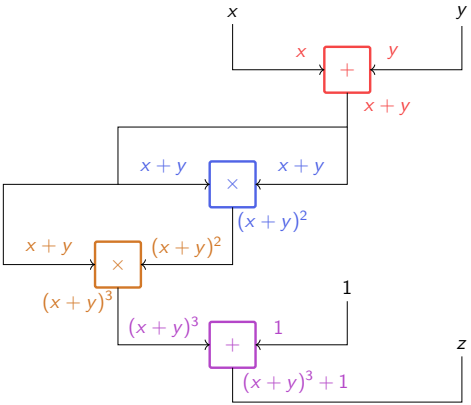
$t_2 = t_0 \times y$ $t_5 = t_4 \times y$ $t_8 = t_1 + t_3$ $t_{11} = t_{10} + 1$

6 constraints

A circuit representation

Factorised expression

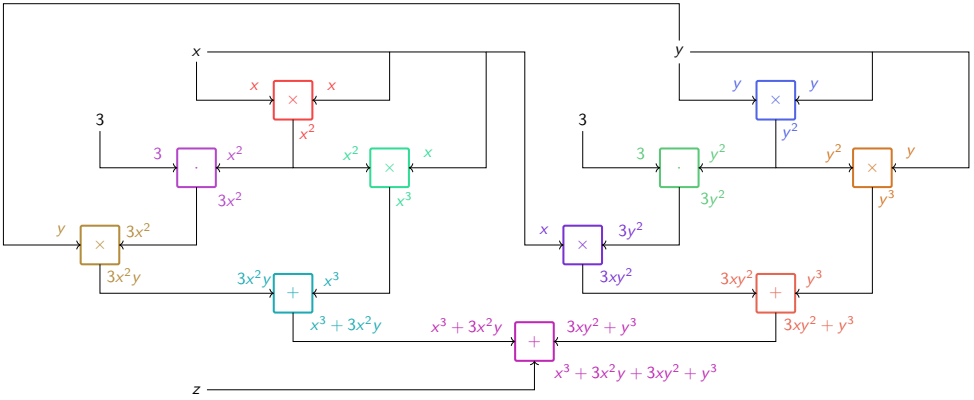
$$z = (x + y)^3 + 1$$



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

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

Plonk Constraints

Additions also have a cost !

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

R1CS

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

2 constraints

Plonk Constraints

Additions also have a cost !

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

R1CS

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

2 constraints

Plonk

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

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

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

3 constraints

Plonk Constraints

Additions also have a cost !

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

R1CS

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

2 constraints

Plonk

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

3 constraints

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

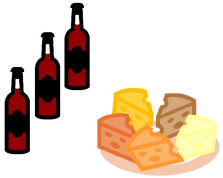
Take-away

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 ?

New AOPs

Why are they different from traditional primitives ?



AOP

« **Appellation d'origine protégée** »

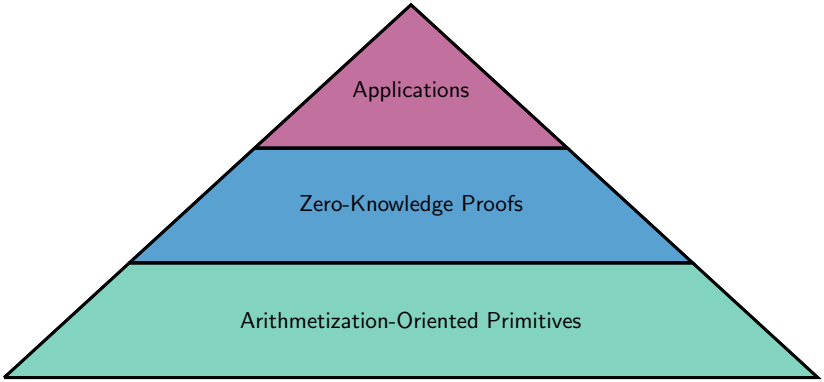
Camembert de Normandie

AOP

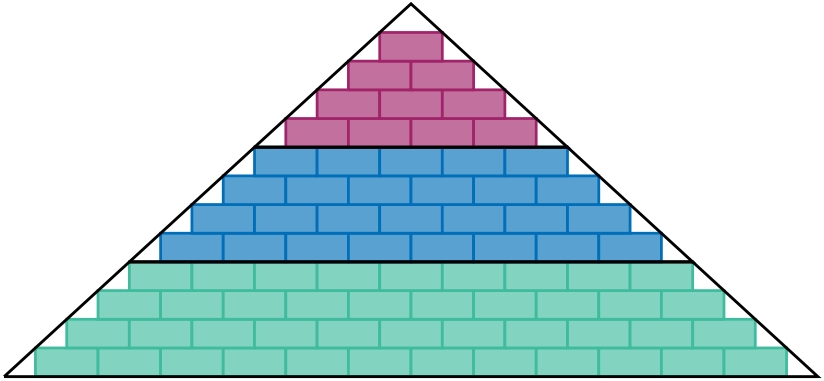
« **Arithmetization-Oriented Primitives** »

Camembert de Normandie

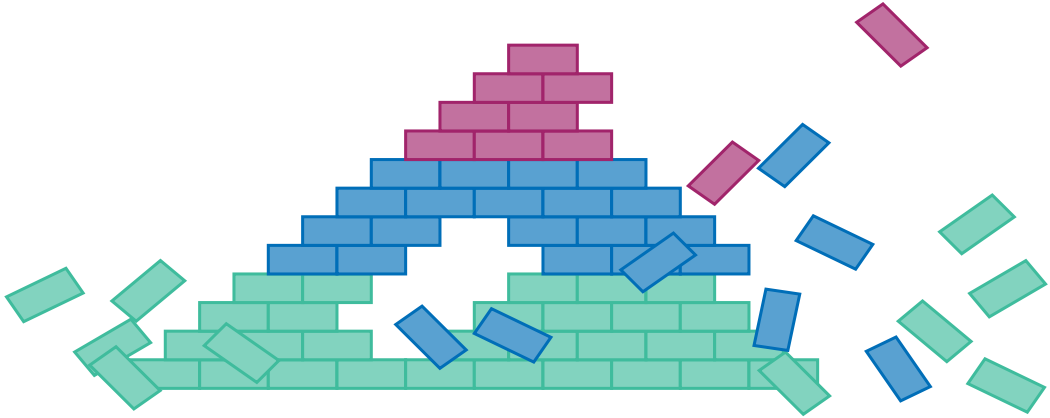
Building blocks of security



Building blocks of security



Building blocks of security



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.

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

Example

Field of AES

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

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

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

Scalar Field of Curve BLS12-381

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

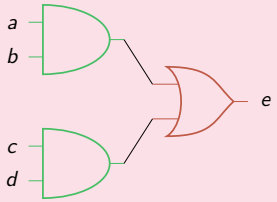
$p =$ 0x73eda753299d7d483339d80809a1d805
53bda402fffe5bfeffffffff00000001

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

New operations

Traditional case

Use of logical gates and CPU instructions.



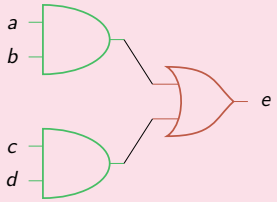
Example

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

New operations

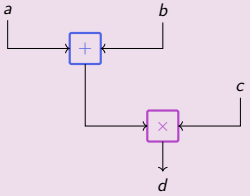
Traditional case

Use of logical gates and CPU instructions.



Arithmetization-Oriented

Use of Arithmetic circuit.

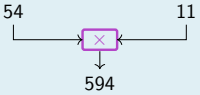


Example

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

Example

$$54 \times 11 = 594$$

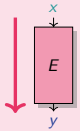


A new metric

Traditional case

Minimize time and memory.

$$y \leftarrow E(x)$$

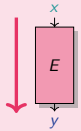


A new metric

Traditional case

Minimize time and memory.

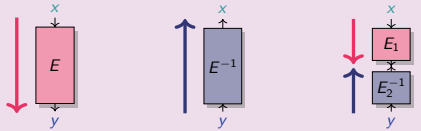
$$y \leftarrow E(x)$$



Arithmetization-Oriented

Minimize the number of multiplications.

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

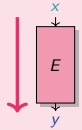


A new metric

Traditional case

Minimize time and memory.

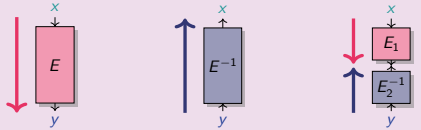
$$y \leftarrow E(x)$$



Arithmetization-Oriented

Minimize the number of multiplications.

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



Example

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

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

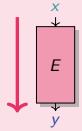
$$y = 4 \text{ (applying } E\text{)}$$

A new metric

Traditional case

Minimize time and memory.

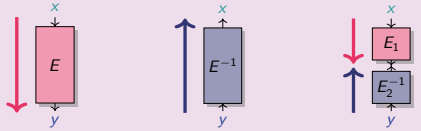
$$y \leftarrow E(x)$$



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**

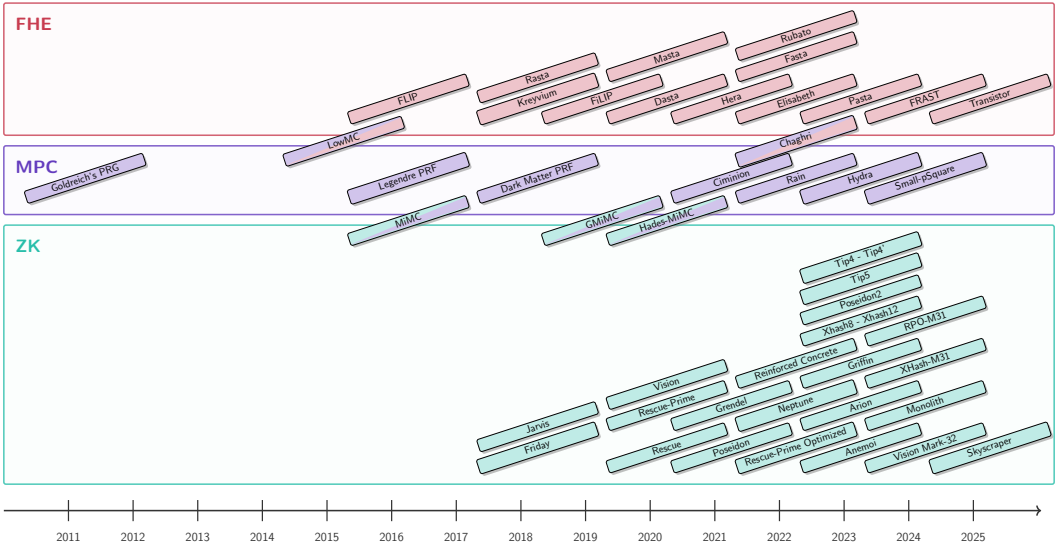
Arithmetization-Oriented

- ⋆ Alphabet :

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

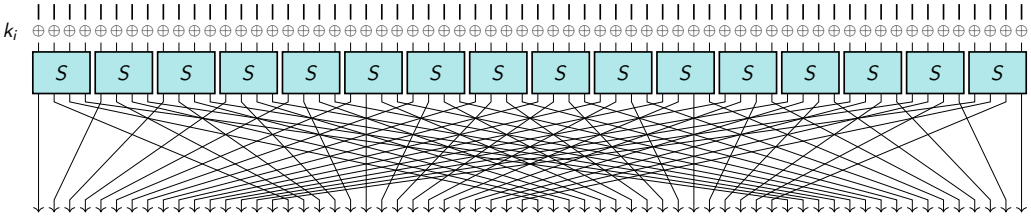
- ⋆ Operations :
  Large finite-field arithmetic
- ⋆ Metric : minimize the number of multiplications for the verification
- ⋆ ≤ 10 **years of Cryptanalysis**

Primitives


FHE
FLIP
LowMC
Rasta
Kreyvium
Masta
FiLIP
Dasta
Hera
Rubato
Fasta
Elisabeth
Pasta
FRAST
Transistor
MPC
Goldreich's PRG
Legendre PRF
MiMC
Dark Matter PRF
GMiMC
Hades-MiMC
Chaghri
Ciminion
Rain
Hydra
Small-pSquare
ZK
Tip4 - Tip4'
Tip5
Poseidon2
Xhash8 - XhashT2
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

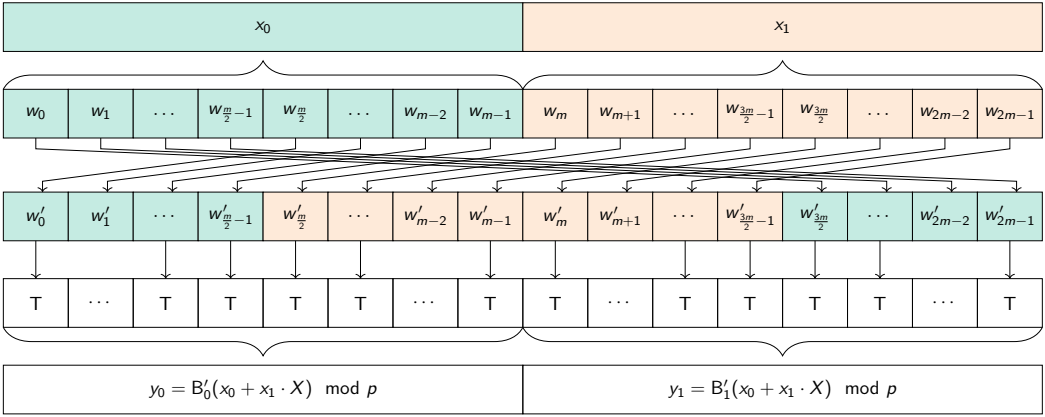
Classical design

Present round function



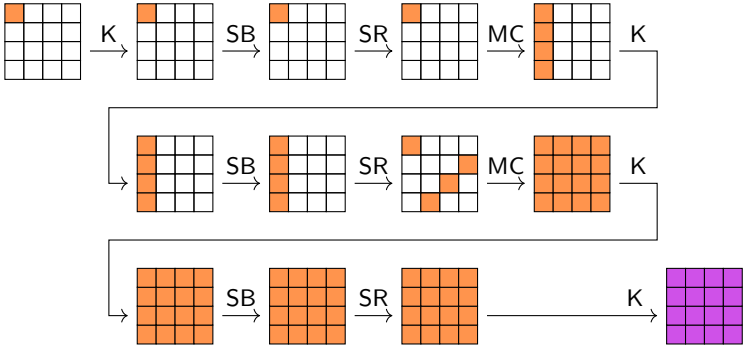
Design of AOPs

Skyscraper Bar layer



Classical cryptanalysis

AES square attack



Cryptanalysis of AOPs

Anemoi linear analysis

Theorem [Rojas-León, 2006]

Let $f \in \mathbb{F}_q[x_1, \ldots, x_n]$, s.t. $\deg(f) = d$.

Suppose that $f = f_d + f_{d'} + \cdots$, where f_d, $f_{d'}$, are resp. **the degree-d, degree-d', homogeneous component** of f, with $\gcd(d, p) = \gcd(d', p) = 1$ and $d'/d > p/(p + (p-1)^2)$.

If the following conditions are satisfied

- ⋆ the hypersurface defined by $f_d = 0$ has at worst **quasi-homogeneous isolated singularities** of degrees prime to p with **Milnor numbers** $\mu_1, \ldots, \mu_s$,
- ⋆ the hypersurface defined by $f_{d'} = 0$ contains none of these singularities,

then we have

$$|S(f)| = \left| \sum_{x \in \mathbb{F}_q^n} \omega^{f(x)} \right| \leq \left((d-1)^n - (d-d') \sum_{i=1}^{s} \mu_i \right) \cdot q^{n/2} .$$

Take-away

Challenges for AOPs

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

Conclusions

ZKPs

- ⋆ "Trusting without seeing"
- ⋆ Many proofs systems, differents type of constraints (R1CS, Plonk, ...)

AOPs

- ⋆ Symmetric primitives, building blocks for ZKPs
- ⋆ New challenges for designs and cryptanalysis

Conclusions

ZKPs

- ⋆ "Trusting without seeing"
- ⋆ Many proofs systems, differents type of constraints (R1CS, Plonk, ...)

AOPs

- ⋆ Symmetric primitives, building blocks for ZKPs
- ⋆ New challenges for designs and cryptanalysis

Thank you !