On the Design Criteria for Symmetric Primitives

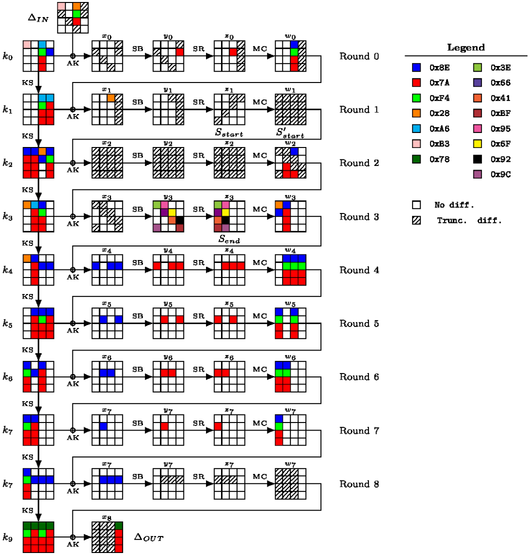
Léo Perrin

Inria, Paris
leo.perrin@inria.fr

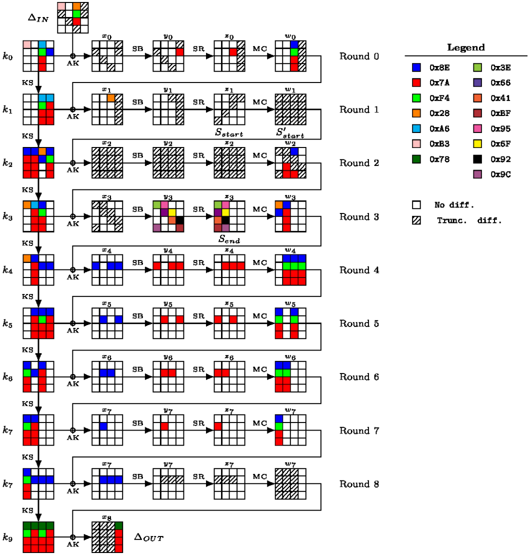
Journées C2

European Research Council
Established by the European Commission

Classical vs. new Symmetric Cryptography



Classical vs. new Symmetric Cryptography



$i = \alpha_0 - 2, \ldots, 0$. This does not change the value of the determinant, and after these row operations, the resulting determinant to compute is:

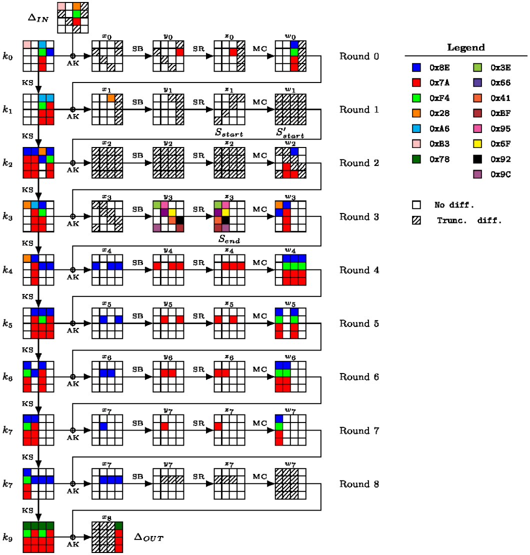
$$\det \begin{pmatrix} 0 & 0 & \ldots & 0 & x_0^{\alpha_0}\boldsymbol{I} + \sum_{i=0}^{\alpha_0-1} x_0^i M_i \\ -\boldsymbol{I} & 0 & \ldots & 0 & x_0^{\alpha_0-1}\boldsymbol{I} + \sum_{i=0}^{\alpha_0-2} x_0^i M_{i+1} \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \ldots & -\boldsymbol{I} & 0 & x_0^2\boldsymbol{I} + \sum_{i=0}^{1} x_0^i M_{i+\alpha_0-2} \\ 0 & \ldots & 0 & -\boldsymbol{I} & x_0\boldsymbol{I} + M_{\alpha_0-1} \end{pmatrix}.$$

In this block matrix representation, the determinant of the full matrix is the determinant of the top right matrix, up to the sign $(-1)^{\alpha_0+1}$. □

Complexity Analysis. We call `polyDet` the procedure returning the polynomial $\det(x_0\boldsymbol{I}_{D_I} - T_0)$ using Lemma 2. This step has a complexity $\tilde{\mathcal{O}}(D_I D_H^{\omega-1}) = \tilde{\mathcal{O}}(\alpha_0 D_H^{\omega})$ with the algorithm of [40]. Note that this is precisely the complexity that was obtained with the algorithm of [12] for systems satisfying the *stability* and *shape position* properties. In order to estimate the logarithmic factors in the complexity formula, we bound the complexity with [34, Theorem 4.4], using a polynomial matrix multiplication algorithm of complexity $\mathcal{O}(D_H^{\omega}\log(\alpha_0) + D_H^2\log(\alpha_0)\log(\log(\alpha_0)))$ [20]. This way, we bound the number of operations of `polyDet` with (when D_H is large):

$$\mathcal{O}\left(\alpha_0\log(\alpha_0)^2 D_H^{\omega} + \alpha_0\log(\alpha_0)^2\log(\log(\alpha_0))D_H^2\right) \approx \mathcal{O}(\alpha_0\log(\alpha_0)^2 D_H^{\omega}). \quad (2)$$

Classical vs. new Symmetric Cryptography



$i = \alpha_0 - 2, \ldots, 0$. This does not change the value of the determinant, and after these row operations, the resulting determinant to compute is:

$$\det \begin{pmatrix} 0 & 0 & \ldots & 0 & x_0^{\alpha_0}\boldsymbol{I} + \sum_{i=0}^{\alpha_0-1} x_0^i M_i \\ -\boldsymbol{I} & 0 & \ldots & 0 & x_0^{\alpha_0-1}\boldsymbol{I} + \sum_{i=0}^{\alpha_0-2} x_0^i M_{i+1} \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \ldots & -\boldsymbol{I} & 0 & x_0^2\boldsymbol{I} + \sum_{i=0}^{1} x_0^i M_{i+\alpha_0-2} \\ 0 & \ldots & 0 & -\boldsymbol{I} & x_0\boldsymbol{I} + M_{\alpha_0-1} \end{pmatrix}.$$

In this block matrix representation, the determinant of the full matrix is the determinant of the top right matrix, up to the sign $(-1)^{\alpha_0+1}$. □

Complexity Analysis. We call polyDet the procedure returning the polynomial $\det(x_0\boldsymbol{I}_{D_I} - T_0)$ using Lemma 2. This step has a complexity $\tilde{\mathcal{O}}(D_I D_H^{\omega-1}) = \tilde{\mathcal{O}}(\alpha_0 D_H^{\omega})$ with the algorithm of [40]. Note that this is precisely the complexity that was obtained with the algorithm of [12] for systems satisfying the *stability* and *shape position* properties. In order to estimate the logarithmic factors in the complexity formula, we bound the complexity with [34, Theorem 4.4], using a polynomial matrix multiplication algorithm of complexity $\mathcal{O}(D_H^{\omega}\log(\alpha_0) + D_H^2\log(\alpha_0)\log(\log(\alpha_0)))$ [20]. This way, we bound the number of operations of polyDet with (when D_H is large):

$$\mathcal{O}\left(\alpha_0\log(\alpha_0)^2 D_H^{\omega} + \alpha_0\log(\alpha_0)^2\log(\log(\alpha_0))D_H^2\right) \approx \mathcal{O}(\alpha_0\log(\alpha_0)^2 D_H^{\omega}). \quad (2)$$

Arithmetization-oriented ZK-friendly hash function for the BLOCKCHAIN

??

In this talk

Is there a revolution going on in symmetric cryptography?

In this talk

Is there a revolution going on in symmetric cryptography?

absolutely not.

In this talk

Is there a revolution going on in symmetric cryptography?

absolutely not.

Design criteria are changing, **again**.

In this talk

Is there a revolution going on in symmetric cryptography?

Part 1.1 (symmetric cryptography)

absolutely not.

Design criteria are changing, **again**.

Part 1.2 (Design criteria)

Part 1

1. How do we build symmetric primitives?
2. On their design constraints

In this talk

Is there a revolution going on in symmetric cryptography?

Part 1.1

absolutely not.

Design criteria are changing, **again**.

Part 1.2 Part 2

Part 1

1. How do we build symmetric primitives?
2. On their design constraints

Part 2

1. Securing computations vs. data
2. Arithmetization-Orientation?

In this talk

Is there a revolution going on in symmetric cryptography?

Part 1.1

absolutely not.

Conclusion

Design criteria are changing, **again**.

Part 1.2

Part 2

Part 1

1. How do we build symmetric primitives?
2. On their design constraints

Part 2

1. Securing computations vs. data
2. Arithmetization-Orientation?

Outline

Plan of this Section

Plan of this Section

What are symmetric primitives?

Definition (Primitive)

A primitive is a very low level algorithm, a *basic brick* used to build larger things.

What are symmetric primitives?

Definition (Primitive)

A primitive is a very low level algorithm, a *basic brick* used to build larger things.

Why would anyone need a "symmetric" primitive?

AES, SHA-3, Chacha20, Skinny, Snow-3G, Poly-1305, PRESENT, Blake, GHASH...

What are symmetric primitives?

Definition (Primitive)

A primitive is a very low level algorithm, a *basic brick* used to build larger things.

Why would anyone need a “symmetric” primitive?

AES, SHA-3, Chacha20, Skinny, Snow-3G, Poly-1305, PRESENT, Blake, GHASH...

Efficiency **Security**

A Crash Course in Symmetric Cryptography (1/2)

Let $\mathbb{F}_q$ be a finite field.

A Crash Course in Symmetric Cryptography (1/2)

Let $\mathbb{F}_q$ be a finite field.

Block Cipher Family $E_K : \mathbb{F}_q{}^n \rightarrow \mathbb{F}_q{}^n$ of bijections operating on blocks of fixed size. AES, Skinny, PRESENT...

A Crash Course in Symmetric Cryptography (1/2)

Let $\mathbb{F}_q$ be a finite field.

Block Cipher Family $E_K : {\mathbb{F}_q}^n \rightarrow {\mathbb{F}_q}^n$ of bijections operating on blocks of fixed size. AES, Skinny, PRESENT...

Stream Cipher Generates a keystream $S_K \in (\mathbb{F}_q)^*$ of arbitrary size. Chacha20, Snow-3G.

A Crash Course in Symmetric Cryptography (1/2)

Let $\mathbb{F}_q$ be a finite field.

Block Cipher Family $E_K : \mathbb{F}_q{}^n \rightarrow \mathbb{F}_q{}^n$ of bijections operating on blocks of fixed size. AES, Skinny, PRESENT...

Stream Cipher Generates a keystream $S_K \in (\mathbb{F}_q)^*$ of arbitrary size. Chacha20, Snow-3G.

Hash Function A function $H : (\mathbb{F}_q)^* \rightarrow \mathbb{F}_q{}^n$ mapping arbitrarily sized inputs to fixed-size outputs. SHA-3, Blake...

A Crash Course in Symmetric Cryptography (1/2)

Let $\mathbb{F}_q$ be a finite field.

Block Cipher Family $E_K : {\mathbb{F}_q}^n \to {\mathbb{F}_q}^n$ of bijections operating on blocks of fixed size. AES, Skinny, PRESENT...

Stream Cipher Generates a keystream $S_K \in (\mathbb{F}_q)^*$ of arbitrary size. Chacha20, Snow-3G.

Hash Function A function $H : (\mathbb{F}_q)^* \to {\mathbb{F}_q}^n$ mapping arbitrarily sized inputs to fixed-size outputs. SHA-3, Blake...

Message Authentication Code (MAC) Family of functions $M_K : (\mathbb{F}_q)^* \to {\mathbb{F}_q}^n$ mapping arbitrarily sized inputs to fixed-size outputs. Poly-1305,GHASH...

A Crash Course in Symmetric Cryptography (2/2)

The Big Trade Secret of Symmetric Cryptographers©

A Crash Course in Symmetric Cryptography (2/2)

The Big Trade Secret of Symmetric Cryptographers©

If you squint hard enough, **everything is a block cipher**

A Crash Course in Symmetric Cryptography (2/2)

The Big Trade Secret of Symmetric Cryptographers©

If you squint hard enough, **everything is a block cipher**

plaintext

A Crash Course in Symmetric Cryptography (2/2)

The Big Trade Secret of Symmetric Cryptographers©

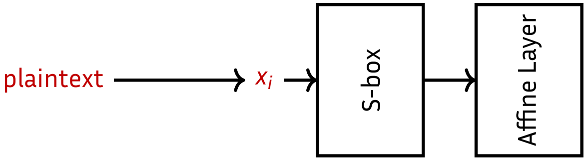
If you squint hard enough, **everything is a block cipher**

plaintext $\longrightarrow x_i$

A Crash Course in Symmetric Cryptography (2/2)

The Big Trade Secret of Symmetric Cryptographers©

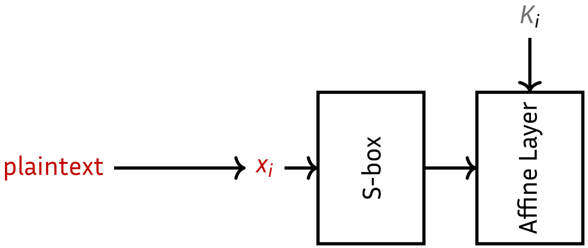
If you squint hard enough, **everything is a block cipher**



A Crash Course in Symmetric Cryptography (2/2)

The Big Trade Secret of Symmetric Cryptographers©

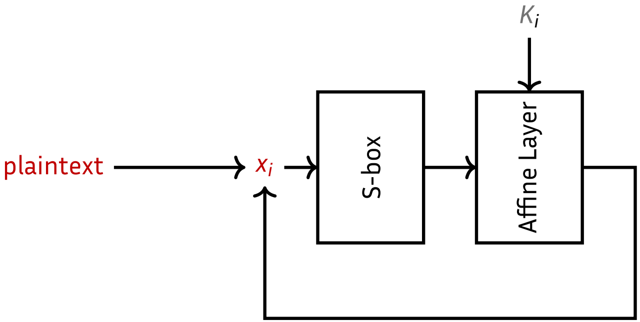
If you squint hard enough, **everything is a block cipher**



A Crash Course in Symmetric Cryptography (2/2)

The Big Trade Secret of Symmetric Cryptographers©

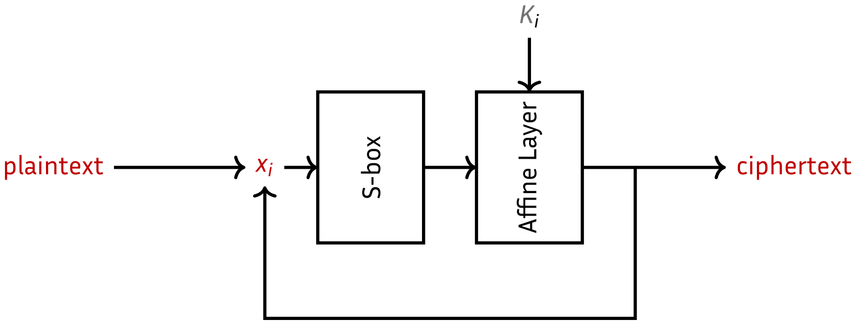
If you squint hard enough, **everything is a block cipher**



A Crash Course in Symmetric Cryptography (2/2)

The Big Trade Secret of Symmetric Cryptographers©

If you squint hard enough, **everything is a block cipher**

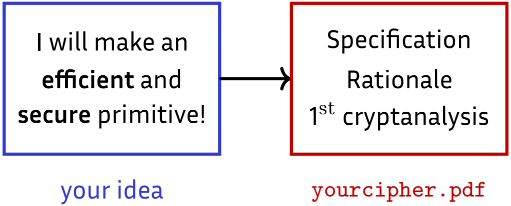


To Build a Cipher

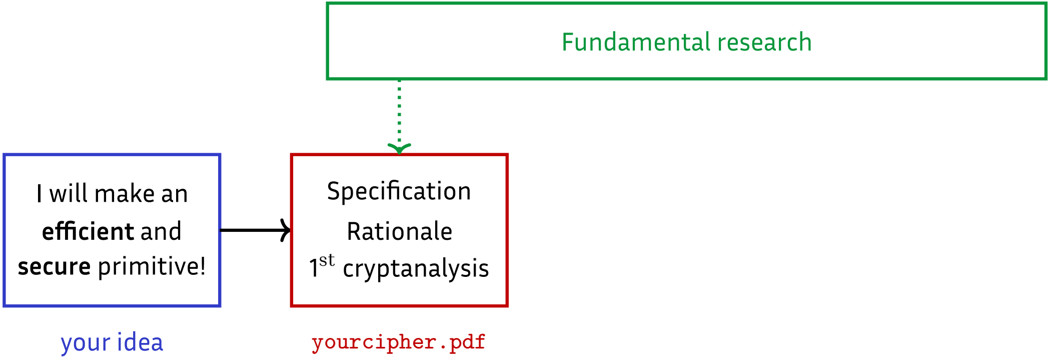
I will make an **efficient** and **secure** primitive!

your idea

To Build a Cipher



To Build a Cipher



To Build a Cipher

Fundamental research

I will make an **efficient** and **secure** primitive!

your idea

Specification
Rationale
1st cryptanalysis

`yourcipher.pdf`

Cryptanalysis!

To Build a Cipher

Fundamental research

I will make an **efficient** and **secure** primitive!

your idea

Specification
Rationale
1^{st} cryptanalysis

`yourcipher.pdf`

Cryptanalysis!

Standardisation

Usage

`yourcipher.so`

What is “Secure”?

Dear audience, what is a secure block cipher?

What is "Secure"?

Dear audience, what is a secure block cipher?

What is **secure** varies

- Has the primitive been analyzed?
  - By the designers? Design quality

What is "Secure"?

Dear audience, what is a secure block cipher?

What is **secure** varies

- Has the primitive been analyzed?
  - By the designers? — Design quality
  - By third parties? — Analyzability

What is "Secure"?

Dear audience, what is a secure block cipher?

What is **secure** varies

- Has the primitive been analyzed?
    - By the designers? — Design quality
    - By third parties? — Analyzability
- Should the primitive work in many context?

What is "Secure"?

Dear audience, what is a secure block cipher?

What is **secure** varies

- Has the primitive been analyzed?
  - By the designers? Design quality
  - By third parties? Analyzability
- Should the primitive work in many context? modularity vs. single use

What is "Secure"?

Dear audience, what is a secure block cipher?

What is **secure** varies

- Has the primitive been analyzed?
  - By the designers? Design quality
  - By third parties? Analyzability
- Should the primitive work in many context? modularity vs. single use

How do we define the **security** that the primitive must provide?

What is "Secure"?

Dear audience, what is a secure block cipher?

What is **secure** varies

- Has the primitive been analyzed?
  - By the designers? — Design quality
  - By third parties? — Analyzability
- Should the primitive work in many context? — modularity vs. single use

How do we define the **security** that the primitive must provide?

What are the relevant forms of cryptanalysis?

What is "Efficient"?

Dear audience, what is an efficient block cipher?

What is "Efficient"?

Dear audience, what is an efficient block cipher?

What is **efficient** varies

- What are the operations that we can use?

  basic logical gates? CPU instructions? AES round?...

What is "Efficient"?

Dear audience, what is an efficient block cipher?

What is **efficient** varies

- What are the operations that we can use?

  basic logical gates? CPU instructions? AES round?...

- What are the associated costs?

  throughput? physical area? ram consumption? number of masked multiplication?...

What is "Efficient"?

Dear audience, what is an efficient block cipher?

What is **efficient** varies

- What are the operations that we can use?
  basic logical gates? CPU instructions? AES round?...
- What are the associated costs?
  throughput? physical area? ram consumption? number of masked multiplication?...

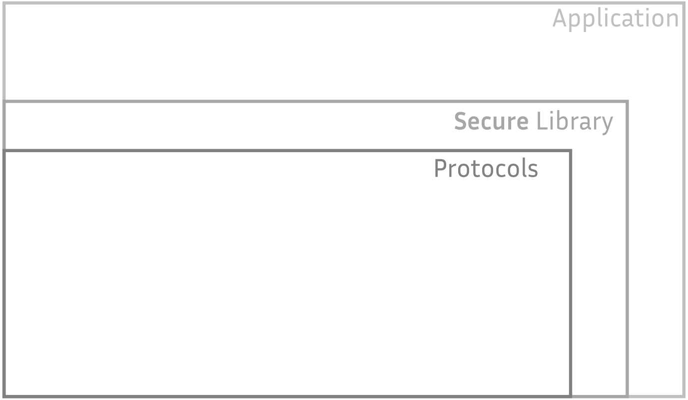
What is the intended execution context?

Plan of this Section

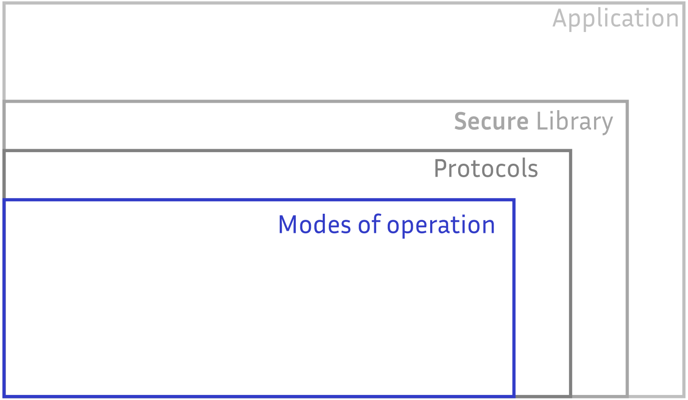
Web Encryption

Web Encryption

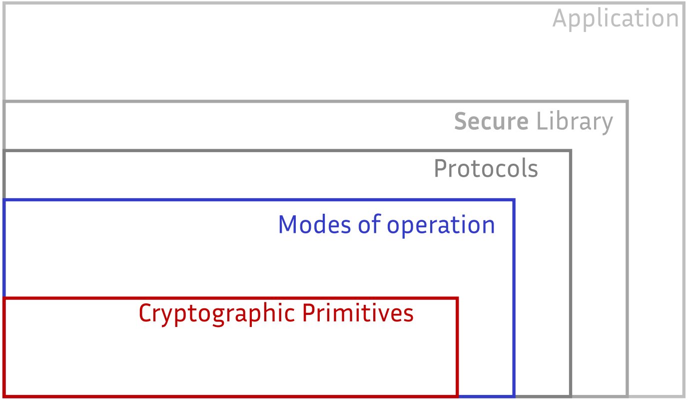
Web Encryption



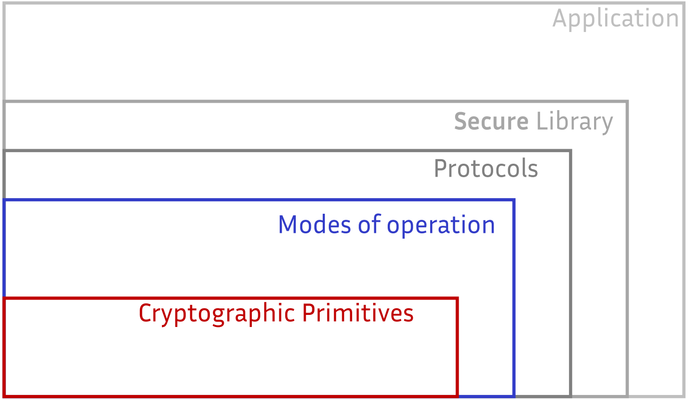
Web Encryption



Web Encryption

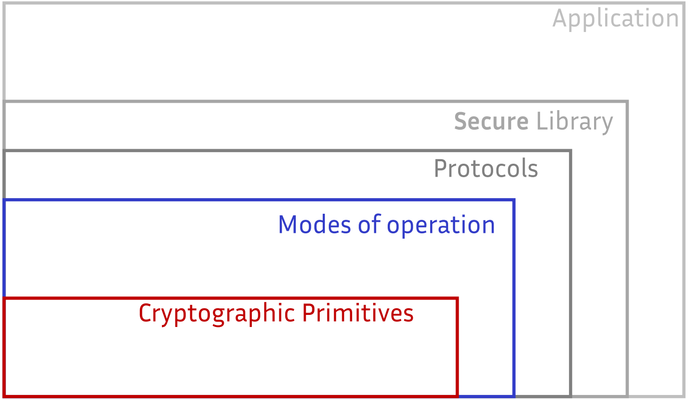


Web Encryption



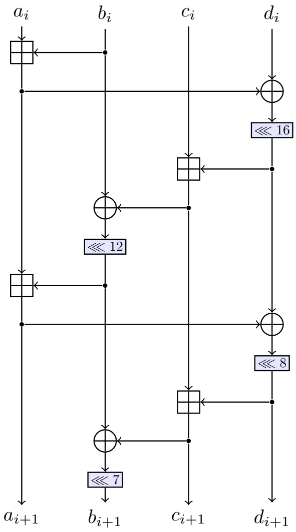
- We want software efficient (computer and smartphone but not micro-controllers) efficient AEAD for packets of a few tens to a few billion bytes.

Web Encryption



- We want software efficient (computer and smartphone but not micro-controllers) efficient AEAD for packets of a few tens to a few billion bytes.
- AES-GCM; Chacha-poly1305.

What Chacha looks like

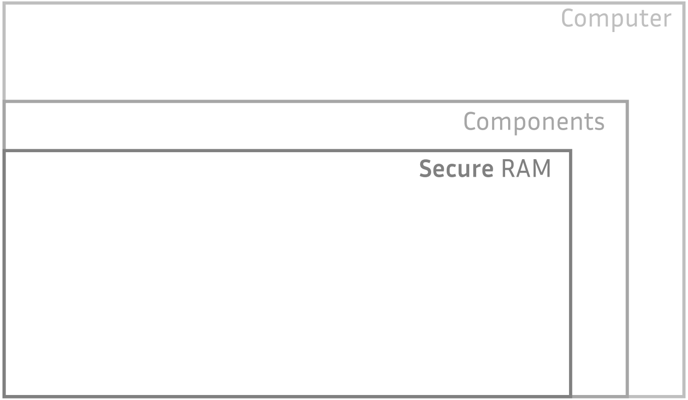


- Addition / Rotation / XOR
- 256-bit key
- 512-bit state
- Defined over 32-bit words

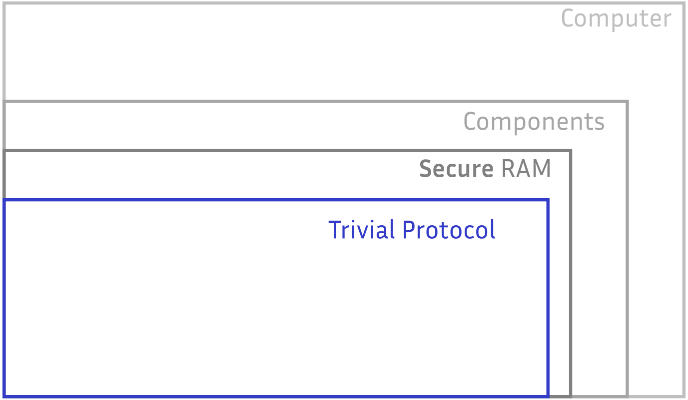
RAM Encryption

RAM Encryption

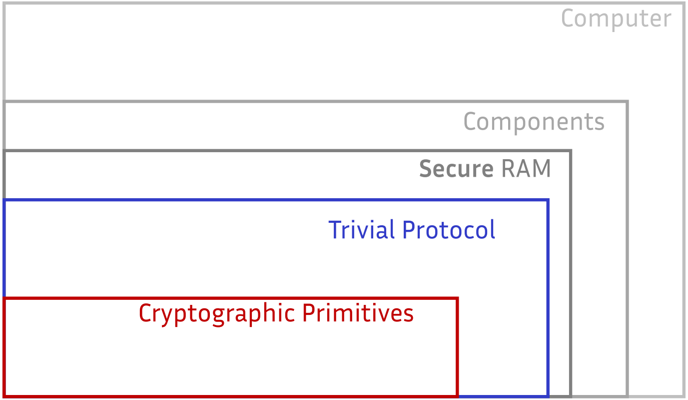
RAM Encryption



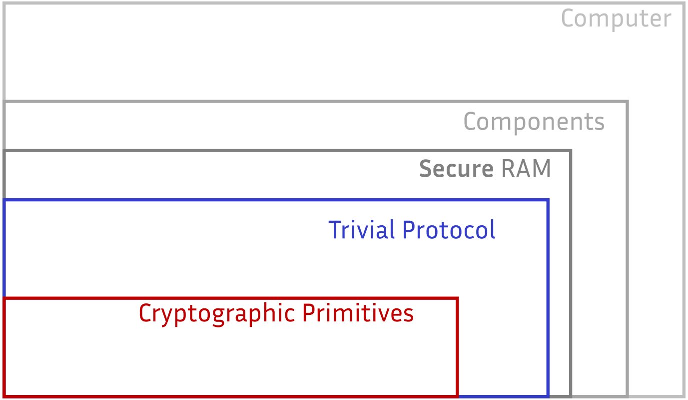
RAM Encryption



RAM Encryption

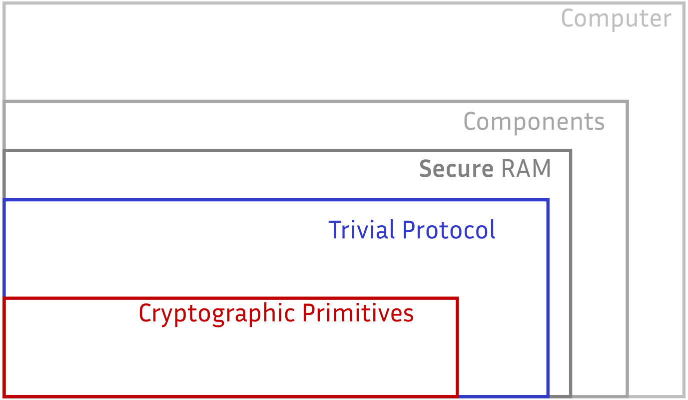


RAM Encryption



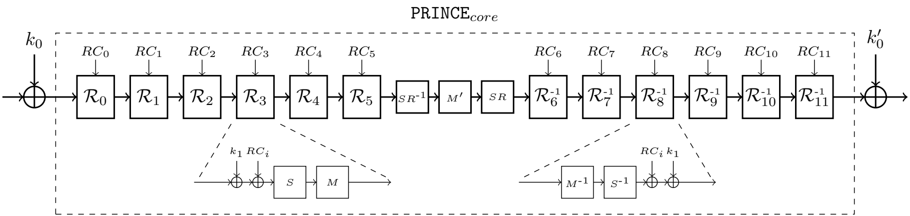
- We want very low latency block encryption for specific (and small) block sizes.

RAM Encryption



- We want very low latency block encryption for specific (and small) block sizes.
- PRINCE? QARMA? not so clear at this stage.

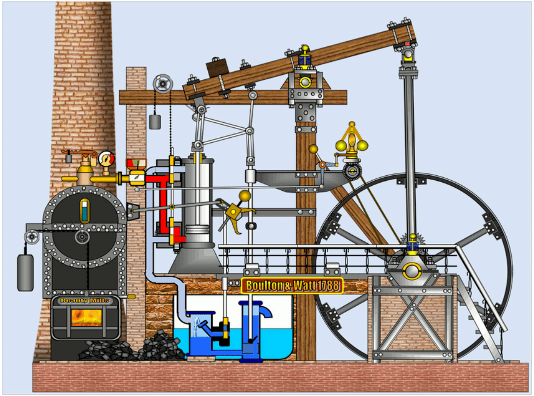
What PRINCE looks like



- 64-bit block size; 128-bit key size ($\approx$)
- 4-bit S-box optimized for hardware
- 2 different 16×16 matrices of $\mathbb{F}_2$, also optimized for hardware
- FX construction
- "α-reflexion": inverse rounds used in the second half

Some Constants

There are many different "big machines", and

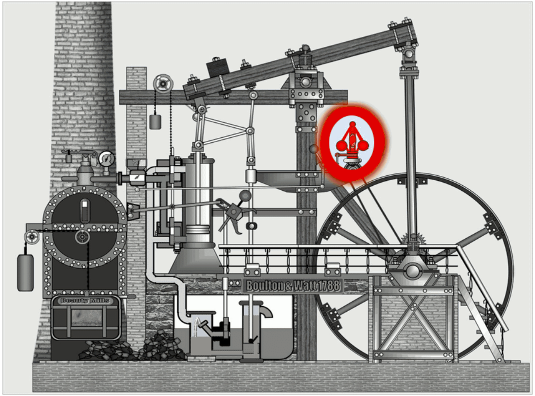


source: https://www.researchgate.net/figure/The-Steam-Engine-of-James-Watt-and-Mathew-Bolton-15_fig2_347657192

Some Constants

There are many different "big machines", and

a symmetric primitive is a very small (but crucial) cog in a very big machine,



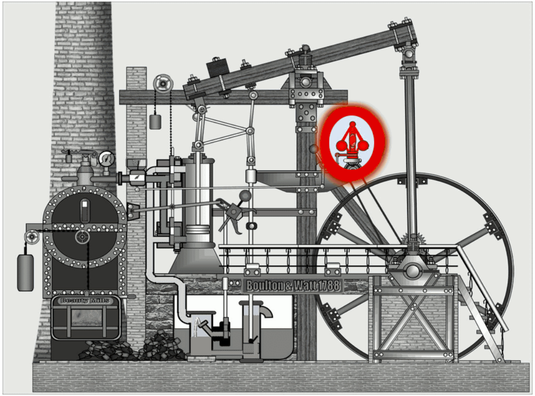
source: https://www.researchgate.net/figure/The-Steam-Engine-of-James-Watt-and-Mathew-Bolton-15_fig2_347657192

Some Constants

There are many different "big machines", and

a symmetric primitive is a very small (but crucial) cog in a very big machine,

this has a huge influence on what the primitive looks like.



source: https://www.researchgate.net/figure/The-Steam-Engine-of-James-Watt-and-Mathew-Bolton-15_fig2_347657192

To Build a Cipher (full)

Fundamental research

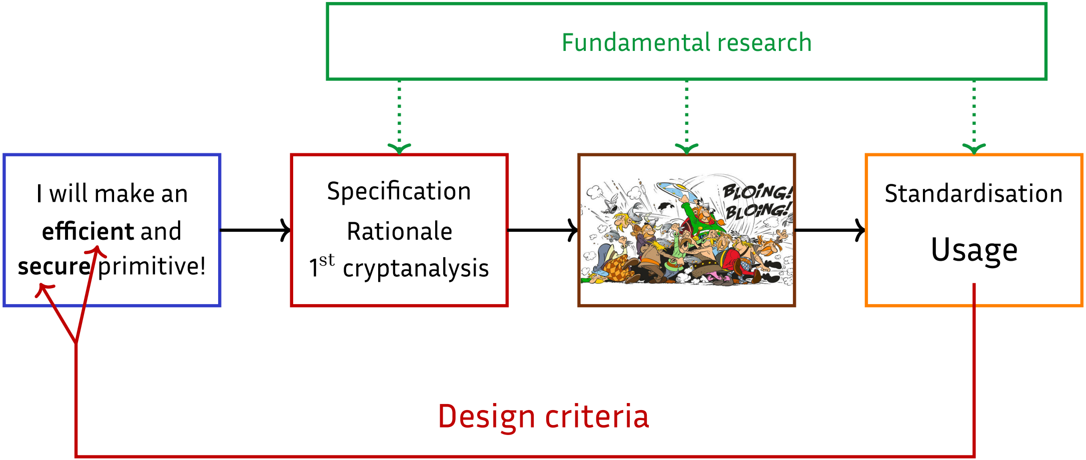
I will make an **efficient** and **secure** primitive!

Specification
Rationale
1^{st} cryptanalysis

Standardisation

Usage

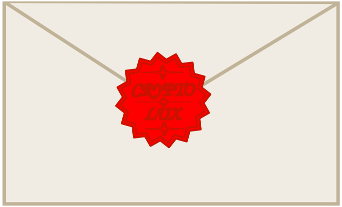
To Build a Cipher (full)



Plan of this Section

Securing Data

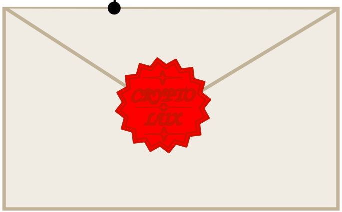
Usually, we secure data (at rest or in transit).



Securing Data

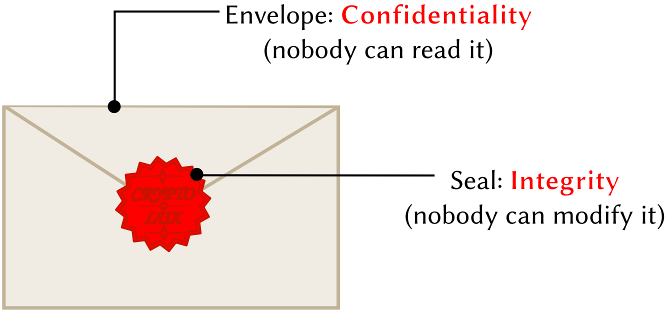
Usually, we secure data (at rest or in transit).

Envelope: Confidentiality
(nobody can read it)



Securing Data

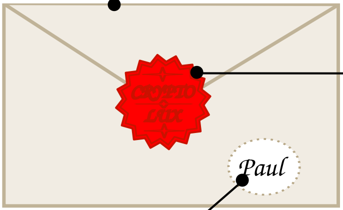
Usually, we secure data (at rest or in transit).



Securing Data

Usually, we secure data (at rest or in transit).

Envelope: Confidentiality
(nobody can read it)



Seal: Integrity
(nobody can modify it)

Signature: Authentication
(it was written by the right person)

Securing Computation

More and more protocols intend to secure computations.

- **FHE** Fully Homomorphic Encryption
- **MPC** Multi Party Computations
- **ZK-*** Zero Knowledge- [proof, argument...]

Plan of this Section

FHE

Goal

Allow a third party to perform some operations on encrypted ciphertext that correspond to meaningful operations on the corresponding plaintext.

FHE

Goal

Allow a third party to perform some operations on encrypted ciphertext that correspond to meaningful operations on the corresponding plaintext. A form of commutation

FHE

Goal

Allow a third party to perform some operations on encrypted ciphertext that correspond to meaningful operations on the corresponding plaintext. A form of commutation

Alice → Bob: $C = F_K(P)$

Bob → Alice: $C' = \mathcal{A}^{\kappa} F_K(P) = F_K\left(\mathcal{A}(P)\right)$

F_K is a homomorphic cipher, **not** a block cipher!

FHE

Goal

Allow a third party to perform some operations on encrypted ciphertext that correspond to meaningful operations on the corresponding plaintext. A form of commutation

Alice → Bob: $C = F_K(P)$

Bob → Alice: $C' = \mathcal{A}^{\kappa} F_K(P) = F_K(\mathcal{A}(P))$

F_K is a homomorphic cipher, **not** a block cipher!

An example of (not F)HE

XOR-ing a constant to a ciphertext obtained using a stream cipher XORs the same constant in the plaintext:

$$C \oplus t = (P \oplus K) \oplus t = (P \oplus t) \oplus K$$

The Symmetric Crypto They Need: Transciphering

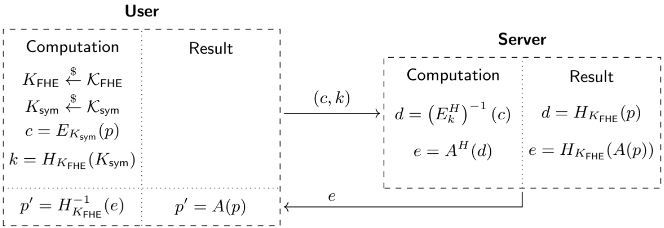


Fig. 1: The principle of transciphering, where E is a symmetric cipher (with secret key K_{sym} sampled from the space $\mathcal{K}_{\mathsf{sym}}$), H is a fully homomorphic cipher (with private key K_{FHE} sampled from the space $\mathcal{K}_{\mathsf{FHE}}$), E^H is a homomorphic evaluation of E, A corresponds to some arbitrary operations, and A^H to their homomorphic evaluation.

source: *Transistor: a TFHE-friendly Stream Cipher*

https://eprint.iacr.org/2025/282

The Symmetric Crypto They Need: Transciphering

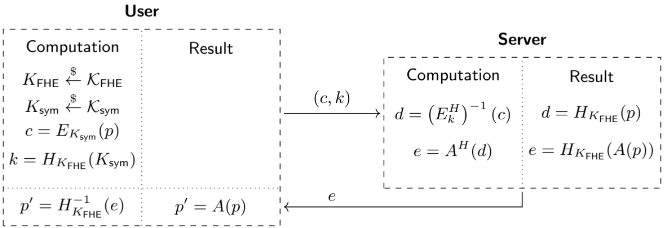


Fig. 1: The principle of transciphering, where E is a symmetric cipher (with secret key K_{sym} sampled from the space $\mathcal{K}_{\mathsf{sym}}$), H is a fully homomorphic cipher (with private key K_{FHE} sampled from the space $\mathcal{K}_{\mathsf{FHE}}$), E^H is a homomorphic evaluation of E, A corresponds to some arbitrary operations, and A^H to their homomorphic evaluation.

source: *Transistor: a TFHE-friendly Stream Cipher*

`https://eprint.iacr.org/2025/282`

A symmetric encryption algorithm with a **high throughput** when evaluated homomorphically

The case of TFHE

Operates on $\mathbb{Z}/m\mathbb{Z}$, where m can be anything, though: more efficient if m is smaller.

The case of TFHE

Operates on $\mathbb{Z}/m\mathbb{Z}$, where m can be anything, though: more efficient if m is smaller.

Operations allowed

Linear Combinations $\sum_i \alpha_i x_i$, where the α_i are constant while x_i is input/key dependent.

- Costs almost nothing in terms of time/communication complexity...
- But noise increases

The case of TFHE

Operates on $\mathbb{Z}/m\mathbb{Z}$, where m can be anything, though: more efficient if m is smaller.

Operations allowed

Linear Combinations $\sum_i \alpha_i x_i$, where the α_i are constant while x_i is input/key dependent.

- Costs almost nothing in terms of time/communication complexity...
- But noise increases

PBS (Programmable BootStrap) $y \leftarrow S(x)$

- Very time consuming...
- .. But resets the noise to a base level

The case of TFHE

Operates on $\mathbb{Z}/m\mathbb{Z}$, where m can be anything, though: more efficient if m is smaller.

Operations allowed

Linear Combinations $\sum_i \alpha_i x_i$, where the α_i are constant while x_i is input/key dependent.

- Costs almost nothing in terms of time/communication complexity...
- But noise increases

PBS (Programmable BootStrap) $y \leftarrow S(x)$

- Very time consuming...
- .. But resets the noise to a base level
- Can be composed with arbitrary table lookups!

The case of TFHE

Operates on $\mathbb{Z}/m\mathbb{Z}$, where m can be anything, though: more efficient if m is smaller.

Operations allowed

Linear Combinations $\sum_i \alpha_i x_i$, where the α_i are constant while x_i is input/key dependent.

- Costs almost nothing in terms of time/communication complexity...
- But noise increases

PBS (Programmable BootStrap) $y \leftarrow S(x)$

- Very time consuming...
- .. But resets the noise to a base level
- Can be composed with arbitrary table lookups!

(not T)FHE operates differently, but **noise** is still present

Examples of stream ciphers for transciphering

Elisabeth-4 [CHMS22]

$q = 2^4$ Can be linearized [GBJR23]

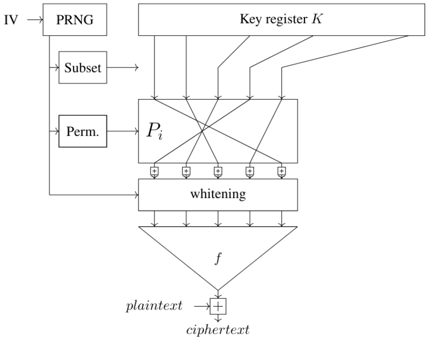


Fig. 1: The group filter permutator design

Examples of stream ciphers for transciphering

Elisabeth-4 [CHMS22]

$q = 2^4$ Can be linearized [GBJR23]

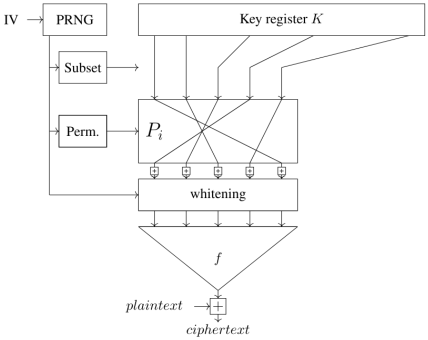


Fig. 1: The group filter permutator design

· ASTA

$q = 2$ or large prime

Many, *many* variants (Rasta, Dasta, Pasta, Masta)

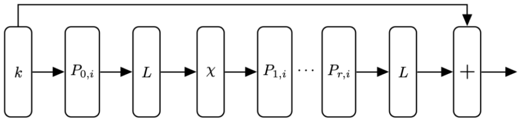


Figure 2: Generation of i-th block of DASTA.

source: *Dasta – Alternative Linear Layer for Rasta*

Examples of stream ciphers for transciphering

Elisabeth-4 [CHMS22]

$q = 2^4$ Can be linearized [GBJR23]

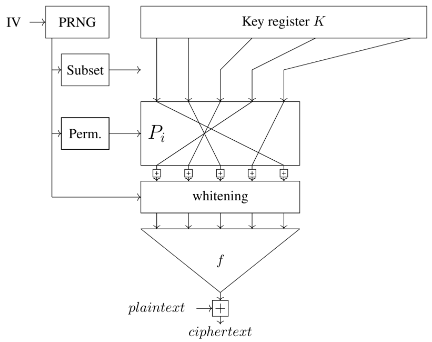


Fig. 1: The group filter permutator design

· ASTA

q = 2 or large prime

Many, *many* variants (Rasta, Dasta, Pasta, Masta)

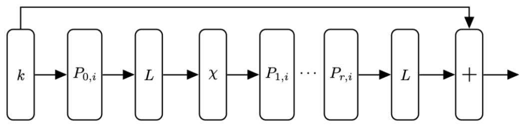


Figure 2: Generation of i-th block of DASTA.

source: *Dasta – Alternative Linear Layer for Rasta*

Shameless plug

Be sure to check the talk of Nicolas Bon this afternoon!

Plan of this Section

Multi-Party Computations

Goal

Allow multiple parties to evaluate a function together even if some parties are not trustworthy.

Multi-Party Computations

Goal

Allow multiple parties to evaluate a function together even if some parties are not trustworthy.

Example

Shamir's secret sharing: $n+1$ points are necessary to interpolate a degree n function.

Multi-Party Computations

Goal

Allow multiple parties to evaluate a function together even if some parties are not trustworthy.

Example

Shamir's secret sharing: $n+1$ points are necessary to interpolate a degree n function.

Applications

- Masking (the side-channel attack counter-measure)
- MPC-in-the-head paradigm (e.g. for Picnic signatures)
- Trojan resilience
- ...

The SymCry They Need: a Lot of Different Things

Masking-friendly (Tweakable) BCs. Low number of multiplications in the underlying field.

Examples: `small-pSquare` [GMM+24b], `Fantomas` [GLSV15],...

The SymCry They Need: a Lot of Different Things

Masking-friendly (Tweakable) BCs. Low number of multiplications in the underlying field.
Examples: small-pSquare [GMM+24b], Fantomas [GLSV15],...

Picnic-friendly BCs. Low number of multiplications, security only for a single query.
Example: LowMC [GRR+16]

The SymCry They Need: a Lot of Different Things

Masking-friendly (Tweakable) BCs. Low number of multiplications in the underlying field.
Examples: `small-pSquare` [GMM+24b], `Fantomas` [GLSV15],...

Picnic-friendly BCs. Low number of multiplications, security only for a single query.
Example: `LowMC` [GRR+16]

Trojan Resilient BCs Reliance only on linear operations (but over different ring/fields).

The SymCry They Need: a Lot of Different Things

Masking-friendly (Tweakable) BCs. Low number of multiplications in the underlying field.
Examples: `small-pSquare` [GMM+24b], `Fantomas` [GLSV15],...

Picnic-friendly BCs. Low number of multiplications, security only for a single query.
Example: `LowMC` [GRR+16]

Trojan Resilient BCs Reliance only on linear operations (but over different ring/fields).
Example: `MOE` [BFL+21]

The SymCry They Need: a Lot of Different Things

Masking-friendly (Tweakable) BCs. Low number of multiplications in the underlying field.
Examples: `small-pSquare` [GMM+24b], `Fantomas` [GLSV15],...

Picnic-friendly BCs. Low number of multiplications, security only for a single query.
Example: `LowMC` [GRR+16]

Trojan Resilient BCs Reliance only on linear operations (but over different ring/fields).
Example: `MOE` [BFL+21]

Pseudo-random Correlated Functions Very low degree evaluation; no chosen plaintext attack allowed; only low data complexity.
Examples: `Crypto DarkMatter` [BIP+18], `VDLPN` [BCG+20]

Plan of this Section

Zero-Knowledge

Principle

I want to convince you I know a secret without revealing it

Zero-Knowledge

Principle

I want to convince you I know a secret without revealing it

A generic goal

To be able to prove/argue that a function was evaluated correctly without revealing its input.

Zero-Knowledge

Principle

I want to convince you I know a secret without revealing it

A generic goal

To be able to prove/argue that a function was evaluated correctly without revealing its input.
Ex: Sudoku time!

Zero-Knowledge

Principle

I want to convince you I know a secret without revealing it

A generic goal

To be able to prove/argue that a function was evaluated correctly without revealing its input.
Ex: Sudoku time!

Applications

- Offload computation to untrusted 3rd parties
- Electronic vote (mixnets)
- Signatures (e.g. with Solid)

Zero-Knowledge

Principle

I want to convince you I know a secret without revealing it

A generic goal

To be able to prove/argue that a function was evaluated correctly without revealing its input.
Ex: Sudoku time!

Applications

- Offload computation to untrusted 3rd parties
- Electronic vote (mixnets)
- Signatures (e.g. with Solid)
- BLOCKCHAIN!!1!

The SymCry They Need: Easily Verifiable Hash Functions

(what follows is a simplification)

They want **hash functions** where the round function has a...

Low Degree Verification

The SymCry They Need: Easily Verifiable Hash Functions

(what follows is a simplification)

They want **hash functions** where the round function has a...

Low Degree Verification

A function $F : \mathbb{F}_q \rightarrow \mathbb{F}_q$ admits a *low degree verification* if there exists a low degree function $G : {\mathbb{F}_q}^2 \rightarrow \mathbb{F}_q$ such that

$$F(x) - z = 0 \;\Leftrightarrow\; G(x, z) = 0 \,.$$

The SymCry They Need: Easily Verifiable Hash Functions

(what follows is a simplification)

They want **hash functions** where the round function has a...

Low Degree Verification

A function $F : \mathbb{F}_q \rightarrow \mathbb{F}_q$ admits a *low degree verification* if there exists a low degree function $G : {\mathbb{F}_q}^2 \rightarrow \mathbb{F}_q$ such that

$$F(x) - z = 0 \Leftrightarrow G(x, z) = 0 .$$

Direct evaluation if F has a low degree, we are fine: $G(x, z) = F(x) - z$

The SymCry They Need: Easily Verifiable Hash Functions

(what follows is a simplification)

They want **hash functions** where the round function has a...

Low Degree Verification

A function $F : \mathbb{F}_q \rightarrow \mathbb{F}_q$ admits a *low degree verification* if there exists a low degree function $G : {\mathbb{F}_q}^2 \rightarrow \mathbb{F}_q$ such that

$$F(x) - z = 0 \Leftrightarrow G(x, z) = 0 .$$

Direct evaluation if F has a low degree, we are fine: $G(x, z) = F(x) - z$

Indirect evaluation if F^{-1} has a low degree, then $G(x, z) = x - F^{-1}(z)$

The SymCry They Need: Easily Verifiable Hash Functions

(what follows is a simplification)

They want **hash functions** where the round function has a...

Low Degree Verification

A function $F : \mathbb{F}_q \rightarrow \mathbb{F}_q$ admits a *low degree verification* if there exists a low degree function $G : {\mathbb{F}_q}^2 \rightarrow \mathbb{F}_q$ such that

$$F(x) - z = 0 \Leftrightarrow G(x, z) = 0 .$$

Direct evaluation if F has a low degree, we are fine: $G(x, z) = F(x) - z$

Indirect evaluation if F^{-1} has a low degree, then $G(x, z) = x - F^{-1}(z)$

Generalization it is sufficient that F is CCZ-equivalent to a low degree function.

Plan of this Section

A Basic Example of Arithmetization

Verifying if $y = c(ax+b)^{10} + x$ in R1CS

A Basic Example of Arithmetization

Verifying if $y = c(ax + b)^{10} + x$ in R1CS

1. $t_0 = ax$
2. $t_1 = t_0 + b$
3. $t_2 = t_1 \times t_1$
4. $t_3 = t_2 \times t_2$
5. $t_4 = t_3 \times t_3$
6. $t_5 = t_2 \times t_4$
7. $t_6 = ct_5$
8. $y = t_6 + x$

A Basic Example of Arithmetization

Verifying if $y = c(ax + b)^{10} + x$ in R1CS

1. $t_0 = ax$
2. $t_1 = t_0 + b$
3. $t_2 = t_1 \times t_1$
4. $t_3 = t_2 \times t_2$
5. $t_4 = t_3 \times t_3$
6. $t_5 = t_2 \times t_4$
7. $t_6 = ct_5$
8. $y = t_6 + x$

A Basic Example of Arithmetization

Verifying if $y = c(ax + b)^{10} + x$ in R1CS

1. $t_0 = ax$
2. $t_1 = t_0 + b$
3. $t_2 = t_1 \times t_1$
4. $t_3 = t_2 \times t_2$
5. $t_4 = t_3 \times t_3$
6. $t_5 = t_2 \times t_4$
7. $t_6 = ct_5$
8. $y = t_6 + x$

This verification costs 4 R1CS constraints

A Basic Example of Arithmetization

Verifying if $y = c(ax + b)^{10} + x$ in R1CS

1. $t_0 = ax$
2. $t_1 = t_0 + b$
3. $t_2 = t_1 \times t_1$
4. $t_3 = t_2 \times t_2$
5. $t_4 = t_3 \times t_3$
6. $t_5 = t_2 \times t_4$
7. $t_6 = ct_5$
8. $y = t_6 + x$

This verification costs 4 R1CS constraints

- How to turn a computation into an arithmetic circuit depends on the operations allowed
- Its cost is also arithmetization-dependent—though low degree is usually welcome!

A Basic Example of Arithmetization

Verifying if $y = c(ax + b)^{10} + x$ in R1CS

1. $t_0 = ax$
2. $t_1 = t_0 + b$
3. $t_2 = t_1 \times t_1$
4. $t_3 = t_2 \times t_2$
5. $t_4 = t_3 \times t_3$
6. $t_5 = t_2 \times t_4$
7. $t_6 = ct_5$
8. $y = t_6 + x$

This verification costs 4 R1CS constraints

- How to turn a computation into an arithmetic circuit depends on the operations allowed
- Its cost is also arithmetization-dependent—though low degree is usually welcome!
- Arithmetization over a field of odd size $\longrightarrow$ nonbinary ciphers

A not basic at all example of arithmetization

The cost of each operation depends on the arithmetization!

Plonk $\neq$ R1CS

A not basic at all example of arithmetization

The cost of each operation depends on the arithmetization!

Plonk ≠ R1CS

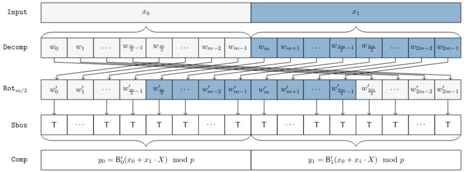


Figure 3: The Bar layer $\mathsf{B}' : \mathbb{F}_{p^n} \to \mathbb{F}_{p^n}$ for $n = 2$ in detail, including the decomposition, the rotation, the S-box, and the composition.

source: *Skyscraper: Fast Hashing on Big Primes*, `https://eprint.iacr.org/2025/058.pdf`

A not basic at all example of arithmetization

The cost of each operation depends on the arithmetization!

Plonk $\neq$ R1CS

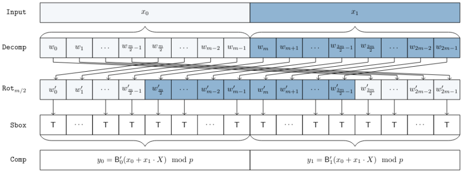


Figure 3: The Bar layer $\mathsf{B}' : \mathbb{F}_{p^n} \to \mathbb{F}_{p^n}$ for $n = 2$ in detail, including the decomposition, the rotation, the S-box, and the composition.

source: *Skyscraper: Fast Hashing on Big Primes*, `https://eprint.iacr.org/2025/058.pdf`

Shameless plug

Be sure to check the talks of Antoine Bak and **Guilhem Jazeron** tomorrow morning!

Symmetric Techniques for Advanced Protocols

MPC FHE ZK

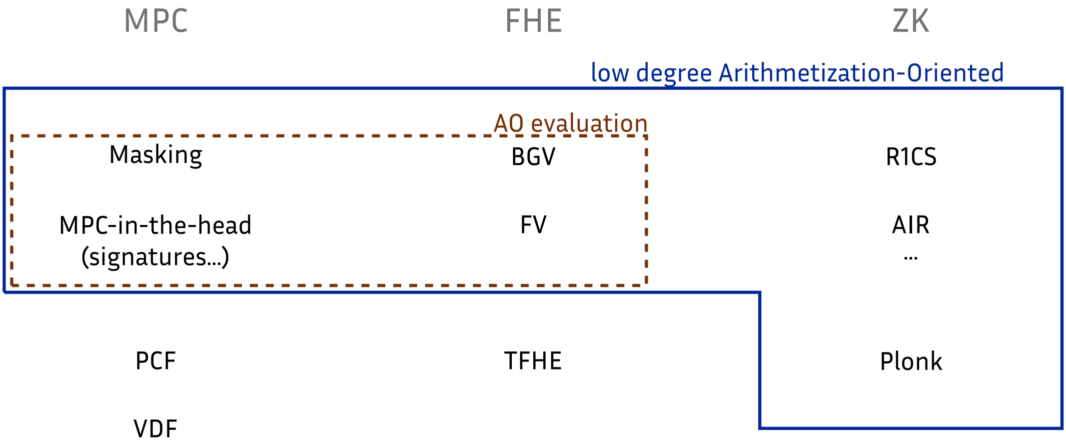
Symmetric Techniques for Advanced Protocols

MPC	FHE	ZK
Masking	BGV	R1CS
MPC-in-the-head (signatures...)	FV	AIR ...
PCF	TFHE	Plonk
VDF		

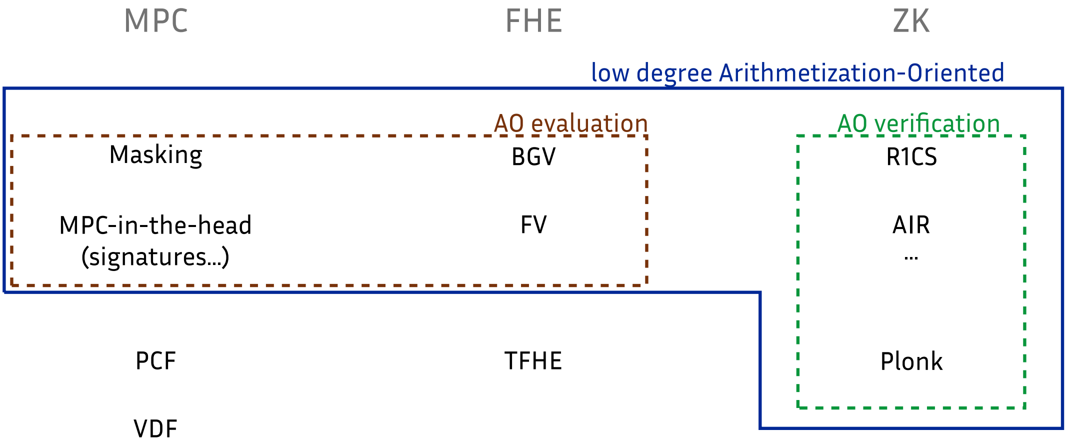
Symmetric Techniques for Advanced Protocols

MPC	FHE	ZK
		low degree Arithmetization-Oriented
Masking	BGV	R1CS
MPC-in-the-head (signatures...)	FV	AIR ...
PCF	TFHE	Plonk
VDF		

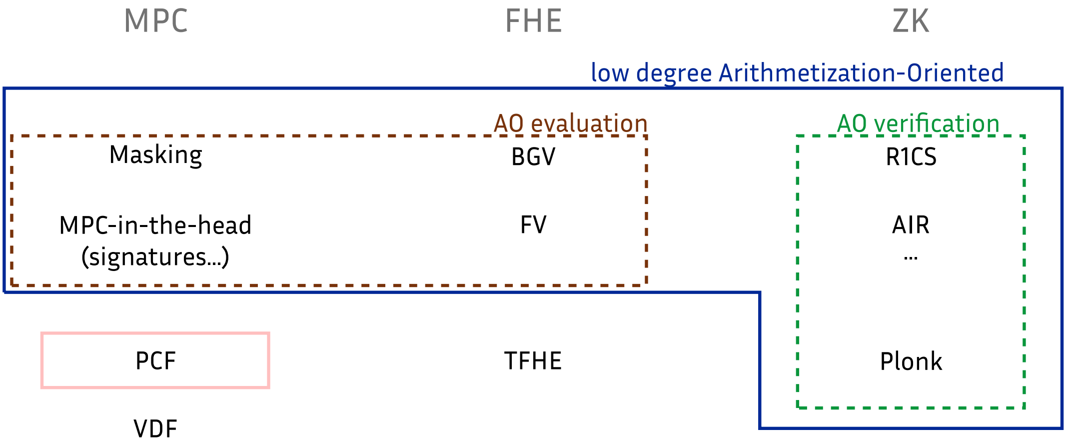
Symmetric Techniques for Advanced Protocols



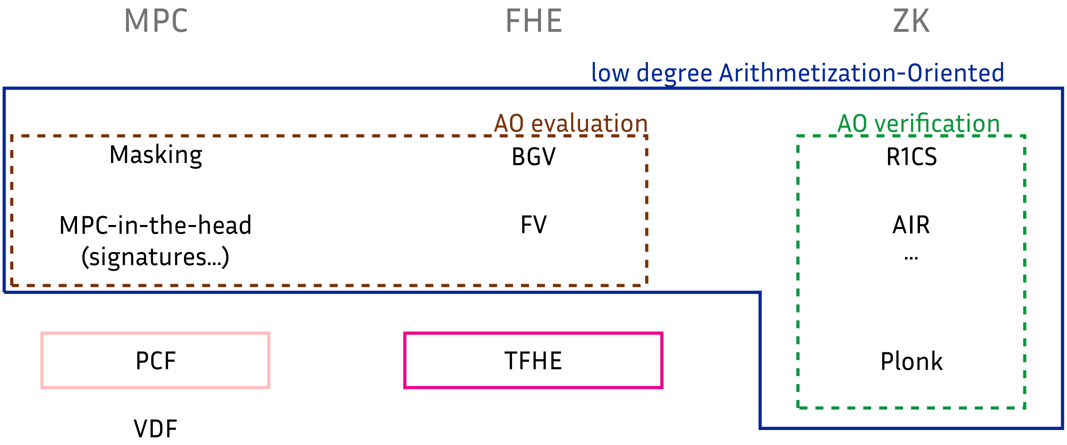
Symmetric Techniques for Advanced Protocols



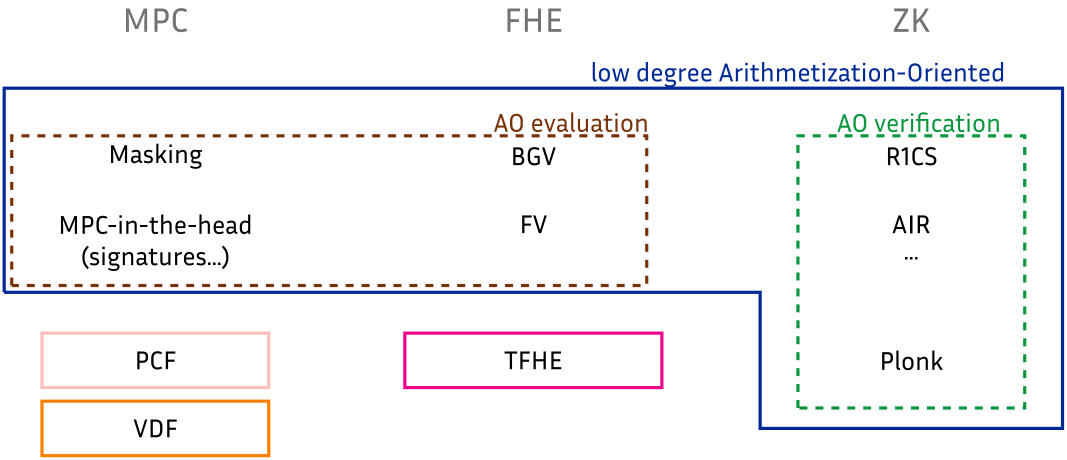
Symmetric Techniques for Advanced Protocols



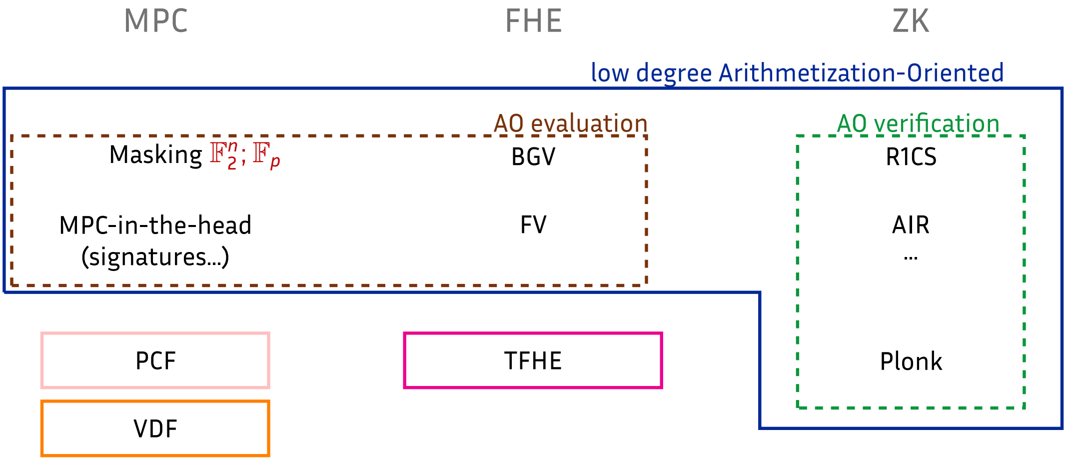
Symmetric Techniques for Advanced Protocols



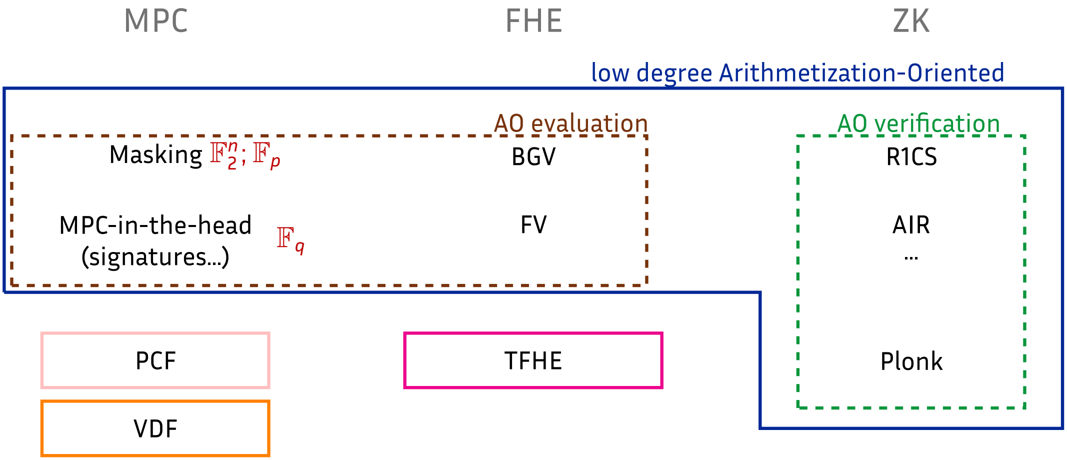
Symmetric Techniques for Advanced Protocols



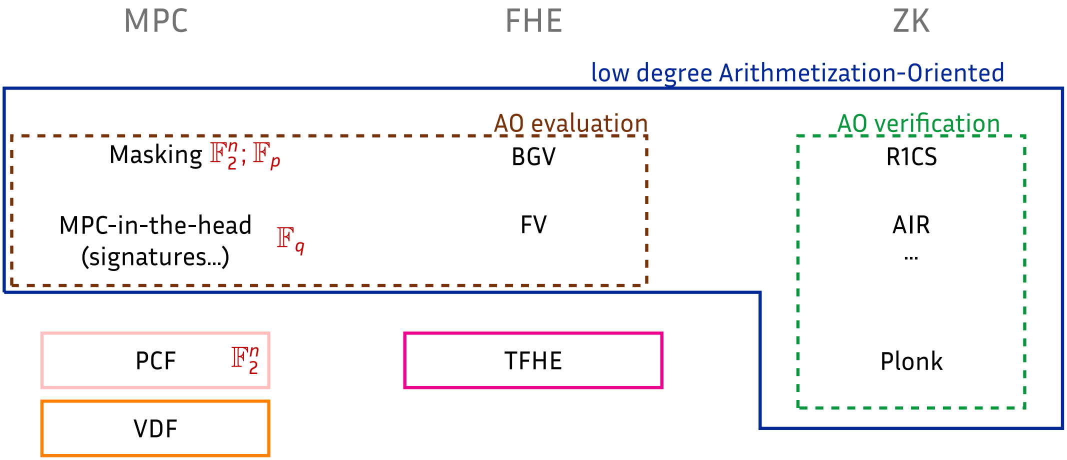
Symmetric Techniques for Advanced Protocols



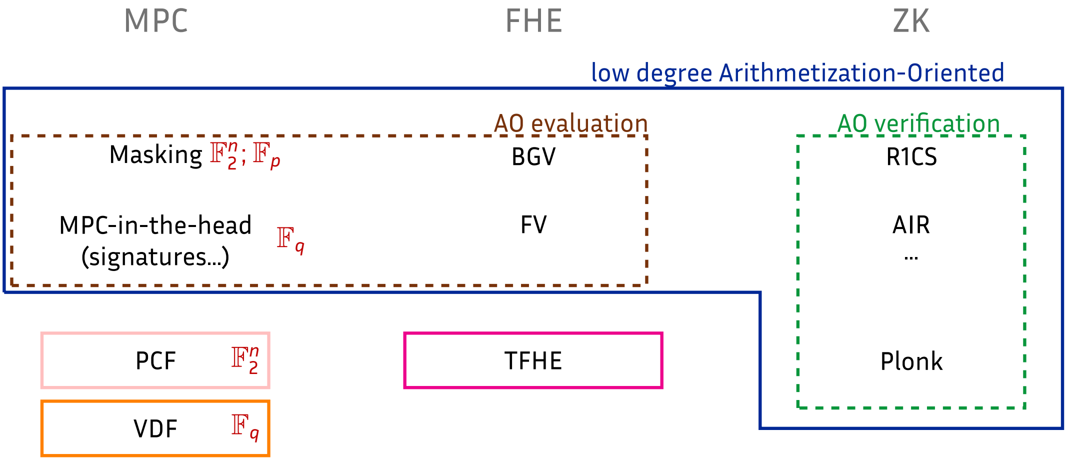
Symmetric Techniques for Advanced Protocols



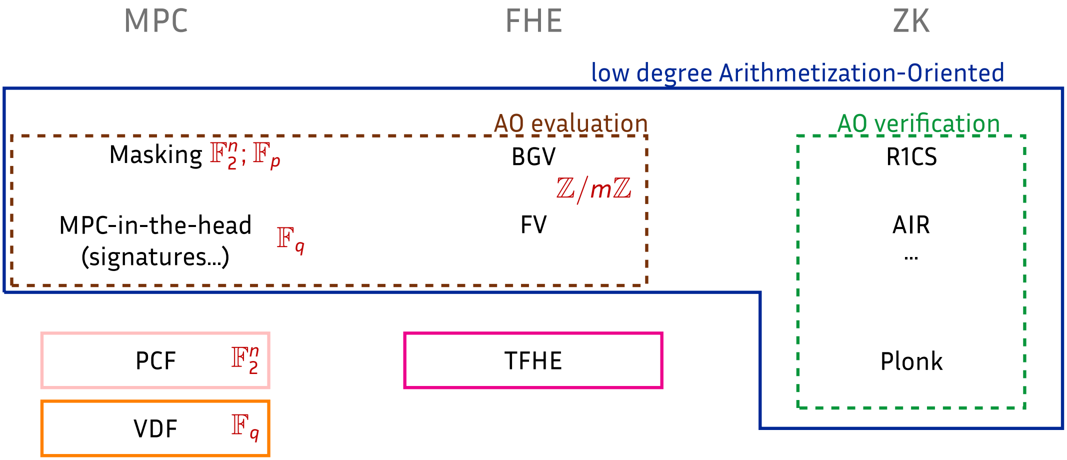
Symmetric Techniques for Advanced Protocols



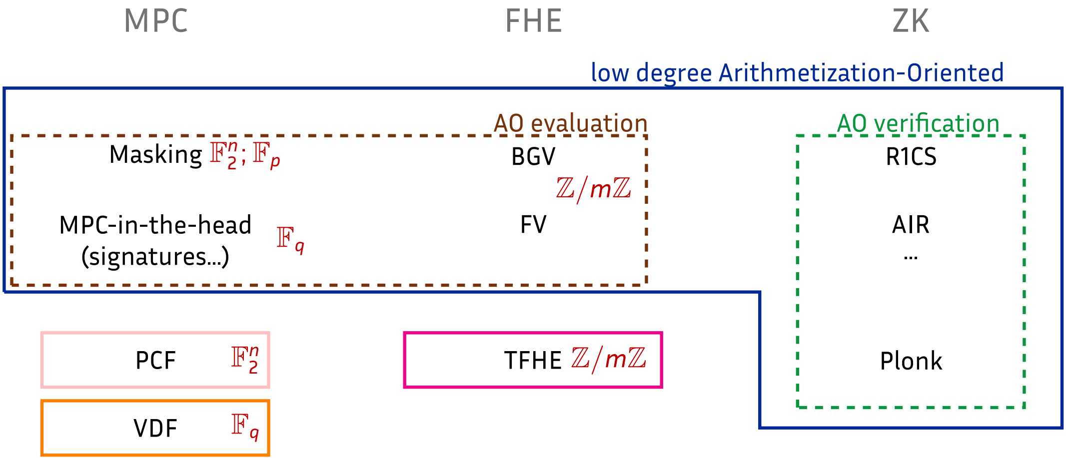
Symmetric Techniques for Advanced Protocols



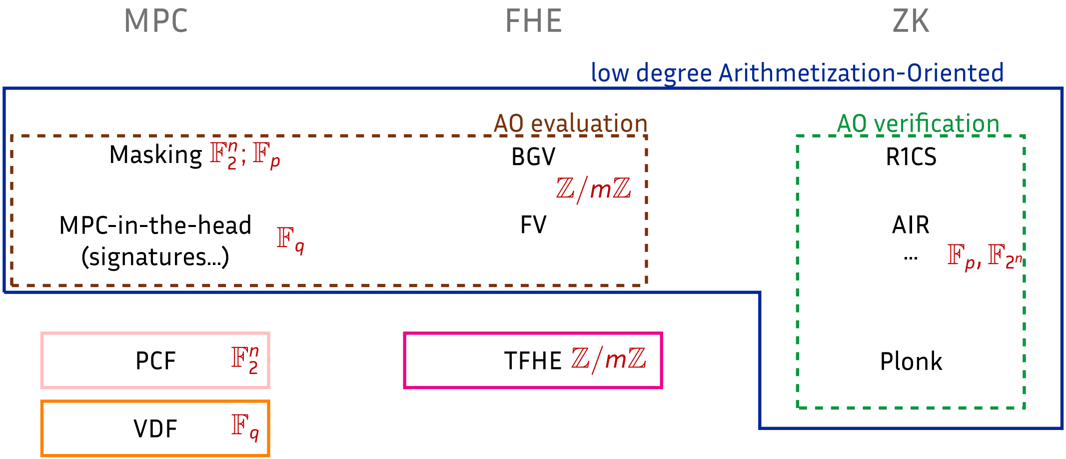
Symmetric Techniques for Advanced Protocols



Symmetric Techniques for Advanced Protocols

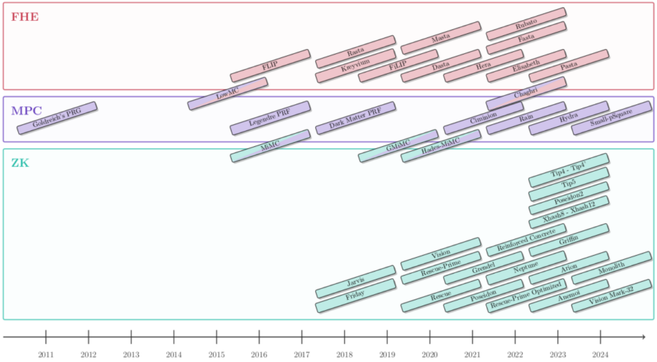


Symmetric Techniques for Advanced Protocols



A Cambrian Explosion

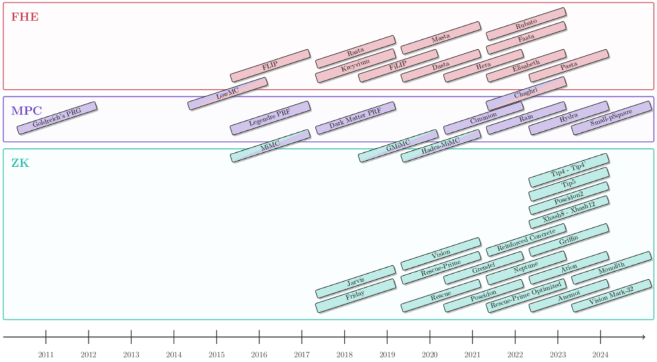
credit: Clémence Bouvier [Bou23]



https://stap-zoo.com/

A Cambrian Explosion

credit: Clémence Bouvier [Bou23]



https://stap-zoo.com/

An **increased diversity of design criteria** leads to a Cambrian explosion of new symmetric primitives!

Plan of this Section

Plan of this Section

Design Process

Once upon a time, a symmetric cryptographer designed a new symmetric primitive

Design Process

Once upon a time, a symmetric cryptographer designed a new symmetric primitive

- Long initial phase to figure out what operations are the most efficient

Design Process

Once upon a time, a symmetric cryptographer designed a new symmetric primitive

- Long initial phase to figure out what operations are the most efficient
- New tricks for the initial security analysis

Design Process

Once upon a time, a symmetric cryptographer designed a new symmetric primitive

- Long initial phase to figure out what operations are the most efficient
- New tricks for the initial security analysis
- Tight collaboration with implementers $\neq$ the people doing the cryptanalysis

Design Process

Once upon a time, a symmetric cryptographer designed a new symmetric primitive

- Long initial phase to figure out what operations are the most efficient
- New tricks for the initial security analysis
- Tight collaboration with implementers $\neq$ the people doing the cryptanalysis
- Multiple iterations during the design to maximize efficiency

Design Process

Once upon a time, a symmetric cryptographer designed a new symmetric primitive

- Long initial phase to figure out what operations are the most efficient
- New tricks for the initial security analysis
- Tight collaboration with implementers $\neq$ the people doing the cryptanalysis
- Multiple iterations during the design to maximize efficiency

Dear audience, which specific primitive am I talking about?

Design Process

Once upon a time, a symmetric cryptographer designed a new symmetric primitive

- Long initial phase to figure out what operations are the most efficient
- New tricks for the initial security analysis
- Tight collaboration with implementers $\neq$ the people doing the cryptanalysis
- Multiple iterations during the design to maximize efficiency

Dear audience, which specific primitive am I talking about?
all of them!

Opinion 1

STAPs are nothing special: we (symmetric people) need to do what we always did.

Underlying Alphabet

$\mathbb{F}_q$ and $\mathbb{F}_2^n$ are not the same!

Underlying Alphabet

$\mathbb{F}_q$ and $\mathbb{F}_2^n$ are not the same!

- Indeed.

Underlying Alphabet

$\mathbb{F}_q$ and $\mathbb{F}_2^n$ are not the same!

- Indeed. In particular, q prime means no Frobenius automorphisms, meaning linear operations are only constant multiplications.

Underlying Alphabet

$\mathbb{F}_q$ and $\mathbb{F}_2^n$ are not the same!

- Indeed. In particular, q prime means no Frobenius automorphisms, meaning linear operations are only constant multiplications.
- Protocol flexibility means STAPs are often in fact primitives generators, i.e., algorithms **generating** primitives.

Underlying Alphabet

$\mathbb{F}_q$ and $\mathbb{F}_2^n$ are not the same!

- Indeed. In particular, q prime means no Frobenius automorphisms, meaning linear operations are only constant multiplications.
- Protocol flexibility means STAPs are often in fact primitives generators, i.e., algorithms **generating** primitives.
- Low degree arithmetization implies low degree algebraic modeling

beware of algebraic attacks!

Underlying Alphabet

$\mathbb{F}_q$ and $\mathbb{F}_2^n$ are not the same!

- Indeed. In particular, q prime means no Frobenius automorphisms, meaning linear operations are only constant multiplications.
- Protocol flexibility means STAPs are often in fact primitives generators, i.e., algorithms **generating** primitives.
- Low degree arithmetization implies low degree algebraic modeling

beware of algebraic attacks!

(there already was a mass extinction event, the **Freelunch attack**

(shameless plug: check out Aurélien's talk later this morning)

Underlying Alphabet

$\mathbb{F}_q$ and $\mathbb{F}_2^n$ are not the same!

- Indeed. In particular, q prime means no Frobenius automorphisms, meaning linear operations are only constant multiplications.
- Protocol flexibility means STAPs are often in fact primitives generators, i.e., algorithms **generating** primitives.
- Low degree arithmetization implies low degree algebraic modeling

beware of algebraic attacks!

(there already was a mass extinction event, the **Freelunch attack**

(shameless plug: check out Aurélien’s talk later this morning)

Opinion 2

Working over $\mathbb{F}_q$ (especially if low degree arithmetizations are needed) and the need for primitive generators introduce **new cryptanalysis vectors**

Underlying Alphabet

$\mathbb{F}_q$ and $\mathbb{F}_2^n$ are not the same!

- Indeed. In particular, q prime means no Frobenius automorphisms, meaning linear operations are only constant multiplications.
- Protocol flexibility means STAPs are often in fact primitives generators, i.e., algorithms **generating** primitives.
- Low degree arithmetization implies low degree algebraic modeling

beware of algebraic attacks!

(there already was a mass extinction event, the **Freelunch attack**

(shameless plug: check out Aurélien's talk later this morning)

Opinion 2

Working over $\mathbb{F}_q$ (especially if low degree arithmetizations are needed) and the need for primitive generators introduce **new cryptanalysis vectors**, but design approaches will rely on tried and true methods.

Primitive Overdose

credit: Diego Delso, CC BY-SA 4.0,

https://commons.wikimedia.org/w/index.php?curid=108259695

Cryptanalysis has not followed the design explosion

Primitive Overdose

credit: Diego Delso, CC BY-SA 4.0,

https://commons.wikimedia.org/w/index.php?curid=108259695

Cryptanalysis has not followed the design explosion

Potential Explanations

- Not all primitives are designed to **ease analysis**
- So many (too many) minor variants...

Primitive Overdose

credit: Diego Delso, CC BY-SA 4.0,

https://commons.wikimedia.org/w/index.php?curid=108259695

Cryptanalysis has not followed the design explosion

Potential Explanations

- Not all primitives are designed to **ease analysis**
- So many (too many) minor variants...

Opinion 3

1. We need more cryptanalysis!
2. We must become better at handling primitive generators.

Plan of this Section

What Have We Learnt?

- The shape of symmetric primitives has always been dictated by external constraints

What Have We Learnt?

- The shape of symmetric primitives has always been dictated by external constraints
- New advanced protocols intended to secure computations need symmetric primitives

What Have We Learnt?

- The shape of symmetric primitives has always been dictated by external constraints
- New advanced protocols intended to secure computations need symmetric primitives
- The **constraints** they impose on the designs imply an unusual look at first glance

What Have We Learnt?

- The shape of symmetric primitives has always been dictated by external constraints
- New advanced protocols intended to secure computations need symmetric primitives
- The **constraints** they impose on the designs imply an unusual look at first glance
- The **constraints** they impose vary, and **we should be careful about lumping them together**

What Have We Learnt?

- The shape of symmetric primitives has always been dictated by external constraints
- New advanced protocols intended to secure computations need symmetric primitives
- The **constraints** they impose on the designs imply an unusual look at first glance
- The **constraints** they impose vary, and **we should be careful about lumping them together**
- A lot of new sets of constraints $\implies$ a lof of new designs (Cambrian explosion)...

What Have We Learnt?

- The shape of symmetric primitives has always been dictated by external constraints
- New advanced protocols intended to secure computations need symmetric primitives
- The **constraints** they impose on the designs imply an unusual look at first glance
- The **constraints** they impose vary, and **we should be careful about lumping them together**
- A lot of new sets of constraints $\implies$ a lof of new designs (Cambrian explosion)...
- ... but very little cryptanalysis.

We need more cryptanalysis!

What Have We Learnt?

- The shape of symmetric primitives has always been dictated by external constraints
- New advanced protocols intended to secure computations need symmetric primitives
- The **constraints** they impose on the designs imply an unusual look at first glance
- The **constraints** they impose vary, and **we should be careful about lumping them together**
- A lot of new sets of constraints $\Longrightarrow$ a lof of new designs (Cambrian explosion)...
- ... but very little cryptanalysis.

We need more cryptanalysis!

Design criteria dictate the work of symmetric cryptographers
they always have, and always will!

What Have We Learnt?

- The shape of symmetric primitives has always been dictated by external constraints
- New advanced protocols intended to secure computations need symmetric primitives
- The **constraints** they impose on the designs imply an unusual look at first glance
- The **constraints** they impose vary, and **we should be careful about lumping them together**
- A lot of new sets of constraints $\implies$ a lof of new designs (Cambrian explosion)...
- ... but very little cryptanalysis.

We need more cryptanalysis!

Design criteria dictate the work of symmetric cryptographers
they always have, and always will!

Thank you!

Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec.
Design of symmetric-key primitives for advanced cryptographic protocols.
IACR Trans. Symm. Cryptol., 2020(3):1–45, 2020.

Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner.
Ciphers for MPC and FHE.
In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 430–454, Sofia, Bulgaria, April 26–30, 2015. Springer Berlin Heidelberg, Germany.

Jules Baudrin, Sonia Belaïd, Nicolas Bon, Christina Boura, Anne Canteaut, Gaëtan Leurent, Pascal Paillier, Léo Perrin, Matthieu Rivain, Yann Rotella, and Samuel Tap.
Transistor: a TFHE-friendly stream cipher.
Cryptology ePrint Archive, Paper 2025/282, 2025.

Clémence Bouvier, Pierre Briaud, Pyrros Chaidos, Léo Perrin, Robin Salen, Vesselin Velichkov, and Danny Willems.

New design techniques for efficient arithmetization-oriented hash functions: Anemoi permutations and Jive compression mode.
In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part III*, volume 14083 of *LNCS*, pages 507–539, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl.
Correlated pseudorandom functions from variable-density lpn.
In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1069–1080. IEEE, 2020.

Olivier Bronchain, Sebastian Faust, Virginie Lallemand, Gregor Leander, Léo Perrin, and François-Xavier Standaert.
MOE: Multiplication operated encryption with trojan resilience.
IACR Trans. Symm. Cryptol., 2021(1):78–129, 2021.

Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu.
Exploring crypto dark matter: New simple PRF candidates and their applications.
In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 699–729, Panaji, India, November 11–14, 2018. Springer, Cham, Switzerland.

Clémence Bouvier.
Cryptanalysis and design of symmetric primitives defined over large finite fields.
Theses, Sorbonne Université, November 2023.

Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey.
Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression.
In Thomas Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 313–333, Bochum, Germany, March 20–23, 2016. Springer Berlin Heidelberg, Germany.

Mingyu Cho, Woohyuk Chung, Jincheol Ha, Jooyoung Lee, Eun-Gyeol Oh, and Mincheol Son.
Frast: Tfhe-friendly cipher based on random s-boxes.
IACR Transactions on Symmetric Cryptology, 2024(3):1–43, Sep. 2024.

Jihoon Cho, Jincheol Ha, Seongkwang Kim, ByeongHak Lee, Joohee Lee, Jooyoung Lee, Dukjae Moon, and Hyojin Yoon.
Transciphering framework for approximate homomorphic encryption.
In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 640–669, Singapore, December 6–10, 2021. Springer, Cham, Switzerland.

- Orel Cosseron, Clément Hoffmann, Pierrick Méaux, and François-Xavier Standaert.
  Towards case-optimized hybrid homomorphic encryption - featuring the elisabeth stream cipher.
  In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part III*, volume 13793 of *LNCS*, pages 32–67, Taipei, Taiwan, December 5–9, 2022. Springer, Cham, Switzerland.
- Christoph Dobraunig, Lorenzo Grassi, Anna Guinet, and Daniël Kuijsters.
  Ciminion: Symmetric encryption based on Toffoli-gates over large finite fields.
  In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 3–34, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland.
- Henri Gilbert, Rachelle Heim Boissier, Jérémy Jean, and Jean-René Reinhard.
  Cryptanalysis of elisabeth-4.
  In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part III*, volume 14440 of *LNCS*, pages 256–284, Guangzhou, China, December 4–8, 2023. Springer, Singapore, Singapore.
- Lorenzo Grassi, Yonglin Hao, Christian Rechberger, Markus Schofnegger, Roman Walch, and Qingju Wang.

Horst meets fluid-SPN: Griffin for zero-knowledge applications.
In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part III*, volume 14083 of *LNCS*, pages 573–606, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

Lorenzo Grassi, Reinhard Lüftenegger, Christian Rechberger, Dragos Rotaru, and Markus Schofnegger.
On a generalization of substitution-permutation networks: The HADES design strategy.
In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 674–704, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland.

Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici.
LS-designs: Bitslice encryption for efficient masked software implementations.
In Carlos Cid and Christian Rechberger, editors, *FSE 2014*, volume 8540 of *LNCS*, pages 18–37, London, UK, March 3–5, 2015. Springer Berlin Heidelberg, Germany.

Lorenzo Grassi, Loïc Masure, Pierrick Méaux, Thorben Moos, and François-Xavier Standaert.
Generalized feistel ciphers for efficient prime field masking - full version.
Cryptology ePrint Archive, Report 2024/431, 2024.

Lorenzo Grassi, Loïc Masure, Pierrick Méaux, Thorben Moos, and François-Xavier Standaert.

Generalized feistel ciphers for efficient prime field masking - full version.
Cryptology ePrint Archive, Paper 2024/431, 2024.

Lorenzo Grassi, Christian Rechberger, Dragos Rotaru, Peter Scholl, and Nigel P. Smart.
MPC-friendly symmetric key primitives.
In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 430–443, Vienna, Austria, October 24–28, 2016. ACM Press.

Clément Hoffmann, Pierrick Méaux, and François-Xavier Standaert.
The patching landscape of elisabeth-4 and the mixed filter permutator paradigm.
In Anupam Chattopadhyay, Shivam Bhasin, Stjepan Picek, and Chester Rebeiro, editors, *INDOCRYPT 2023, Part I*, volume 14459 of *LNCS*, pages 134–156, Goa, India, December 10–13, 2023. Springer, Cham, Switzerland.

Plan of this Section

Plan of this Section

TFHE: corresponding stream ciphers

Elisabeth-4 [CHMS22] ; $q = 2^4$
Uses a constant key register on which index-dependent non-linear functions are applied.
Can be linearized [GBJR23]

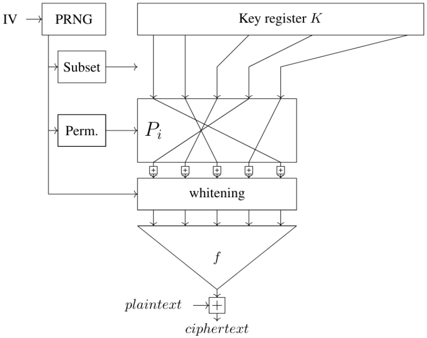


Fig. 1: The group filter permutator design

source: *Towards Case-Optimized Hybrid Homomorphic Encryption Featuring the Elisabeth Stream Cipher*

TFHE: corresponding stream ciphers

Elisabeth-4 [CHMS22] ; $q = 2^4$
Uses a constant key register on which index-dependent non-linear functions are applied.
Can be linearized [GBJR23]

Gabriel... [HMS23]
(Elisabeth-4 follow-ups)

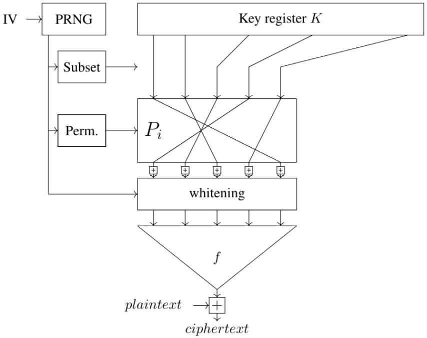


Fig. 1: The group filter permutator design

source: *Towards Case-Optimized Hybrid Homomorphic Encryption Featuring the Elisabeth Stream Cipher*

TFHE: corresponding stream ciphers

Elisabeth-4 [CHMS22] ; $q = 2^4$
Uses a constant key register on which index-dependent non-linear functions are applied.
Can be linearized [GBJR23]

Gabriel... [HMS23]
(Elisabeth-4 follow-ups)

FRAST [CCH$^+$24] ; $q = 2^4$ A block cipher in a CTR-mode variant.
See you at the rump session :D

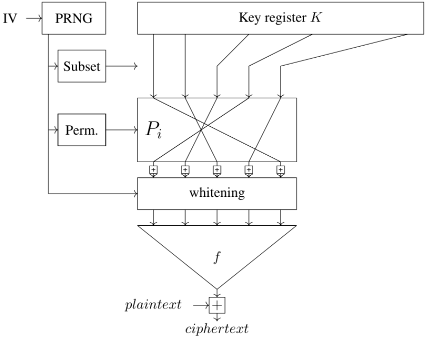


Fig. 1: The group filter permutator design

source: *Towards Case-Optimized Hybrid Homomorphic Encryption Featuring the Elisabeth Stream Cipher*

TFHE: corresponding stream ciphers

Elisabeth-4 [CHMS22] ; $q = 2^4$
Uses a constant key register on which index-dependent non-linear functions are applied.
Can be linearized [GBJR23]

Gabriel... [HMS23]
(Elisabeth-4 follow-ups)

FRAST [CCH+24] ; $q = 2^4$ A block cipher in a CTR-mode variant.
See you at the rump session :D

Transistor [BBB+25] ; $q = 2^4 + 1$
SNOW-like round structure
See you at Anne's invited talk :D

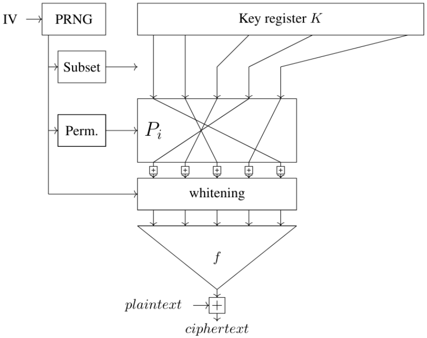


Fig. 1: The group filter permutator design

source: *Towards Case-Optimized Hybrid Homomorphic Encryption Featuring the Elisabeth Stream Cipher*

BGV/FV: corresponding stream ciphers

- **ASTA** $q = 2$ or large prime
  Use very few rounds with a low degree.
  Rely on large, randomly generated, nonce-dependent matrices.

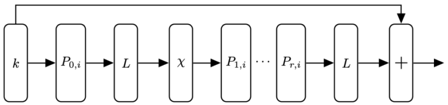


Figure 2: Generation of i-th block of DASTA.

source:
Dasta – Alternative Linear Layer for Rasta

BGV/FV: corresponding stream ciphers

·ASTA $q = 2$ or large prime
Use very few rounds with a low degree.
Rely on large, randomly generated, nonce-dependent matrices.

"Kreyvium" [CCF+16] $q = 2$
Basically Trivium!
Binary state updated with NLFSRs.

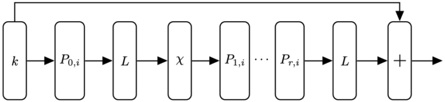


Figure 2: Generation of i-th block of DASTA.

source:
Dasta – Alternative Linear Layer for Rasta

BGV/FV: corresponding stream ciphers

- **ASTA** $q = 2$ or large prime
  Use very few rounds with a low degree.
  Rely on large, randomly generated, nonce-dependent matrices.
- **“Kreyvium”** [CCF+16] $q = 2$
  Basically Trivium!
  Binary state updated with NLFSRs.
- **HERA** [CHK+21] q large prime
  A block cipher in a kind of CTR-mode variant.

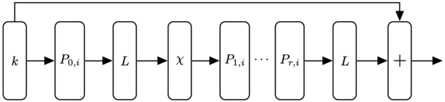
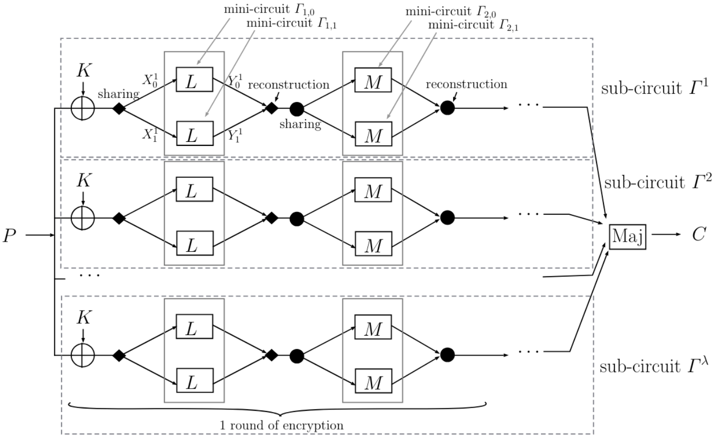


Figure 2: Generation of i-th block of DASTA.

source:
Dasta – Alternative Linear Layer for Rasta

Plan of this Section

Trojan Resilience



source: *MOE: Multiplication Operated Encryption with Trojan Resilience*

https://tosc.iacr.org/index.php/ToSC/article/view/8834

MPC-Friendly Encryption

LowMC [ARS+15] $q = 2$

SPN with partial layer of quadratic S-boxes.

Rely on large, randomly generated matrices.

Only one encryption/key; broken anyway

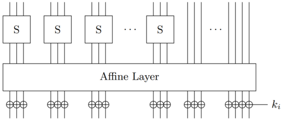


Fig. 1. Depiction of one round of encryption with LowMC.

source:

Ciphers for MPC and FHE

MPC-Friendly Encryption

LowMC [ARS+15] $q = 2$
SPN with partial layer of quadratic S-boxes.
Rely on large, randomly generated matrices.
Only one encryption/key; broken anyway

Ciminion [DGGK21] no specific constraints on q
3-branch Feistel network with a single multiplication/round.

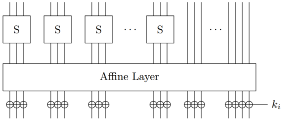


Fig. 1. Depiction of one round of encryption with LowMC.

source:
Ciphers for MPC and FHE

MPC-Friendly Encryption

LowMC [ARS+15] $q = 2$
- SPN with partial layer of quadratic S-boxes.
- Rely on large, randomly generated matrices.
- Only one encryption/key; broken anyway

Cimინion [DGGK21] no specific constraints on q
- 3-branch Feistel network with a single multiplication/round.

small-pSquare [GMM+24a] $p = q = 127$
- Generalized Feistel network with low degree round function.
- Optimized specifically for hardware masking.

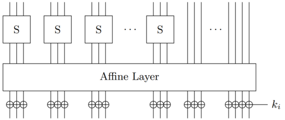


Fig. 1. Depiction of one round of encryption with LowMC.

source:
Ciphers for MPC and FHE

MPC-Friendly Encryption

LowMC [ARS+15] $q = 2$
SPN with partial layer of quadratic S-boxes.
Rely on large, randomly generated matrices.
Only one encryption/key; broken anyway

Ciminion [DGGK21] no specific constraints on q
3-branch Feistel network with a single multiplication/round.

small-pSquare [GMM+24a] $p = q = 127$
Generalized Feistel network with low degree round function.
Optimized specifically for hardware masking.

MOE [BFL+21] $q = 2^{128}$, $m = 2^{128}$
Dedicated structure with linear operations in $\mathbb{F}_q$ and $\mathbb{Z}/q\mathbb{Z}$. Intended for hardware trojan resilience.

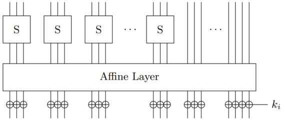


Fig. 1. Depiction of one round of encryption with LowMC.

source:
Ciphers for MPC and FHE

Pseudo-Correlated Functions

A new challenger!

At the start of some MPC protocols, it is necessary to share some bits that are correlated between the participants.

Very low multiplicative depth ·

Pseudo-Correlated Functions

A new challenger!

At the start of some MPC protocols, it is necessary to share some bits that are correlated between the participants.

Very low multiplicative depth · Very low data complexity ·

Pseudo-Correlated Functions

A new challenger!

At the start of some MPC protocols, it is necessary to share some bits that are correlated between the participants.

Very low multiplicative depth · Very low data complexity · Only known plaintext, no chosen plaintext!

Pseudo-Correlated Functions

A new challenger!

At the start of some MPC protocols, it is necessary to share some bits that are correlated between the participants.

Very low multiplicative depth · Very low data complexity · Only known plaintext, no chosen plaintext!

Crypto DarkMatter [BIP+18]

$$F_k(x) := \left(\sum_{i=0}^{n-1} k_i x_i \mod 2 + \sum_{i=0}^{n-1} k_i x_i \mod 3 \right) \mod 2, \quad \text{for } x \in \{0,1\}^n.$$

Pseudo-Correlated Functions

A new challenger!

At the start of some MPC protocols, it is necessary to share some bits that are correlated between the participants.

Very low multiplicative depth · Very low data complexity · Only known plaintext, no chosen plaintext!

Crypto DarkMatter [BIP+18]

$$F_k(x) := \left(\sum_{i=0}^{n-1} k_i x_i \mod 2 + \sum_{i=0}^{n-1} k_i x_i \mod 3 \right) \mod 2, \quad \text{for } x \in \{0,1\}^n.$$

VDLPN [BCG+20]

$$f_k(x) = \bigoplus_{i=1}^{D} \bigoplus_{j=1}^{w} \bigwedge_{\ell=1}^{i} (x_{i,j,\ell} \oplus k_{i,j,\ell}),$$

Plan of this Section

ZK-Friendly Hash Functions

Poseidon [GLR+20] no specific constraints q
SPN with partial layer of low degree monomial S-boxes.

Full rounds – partial round – full rounds.

ZK-Friendly Hash Functions

Poseidon [GLR+20] no specific constraints q

SPN with partial layer of low degree monomial S-boxes.

Full rounds – partial round – full rounds.

Rescue [AAB+20] q large prime

SPN with low degree monomials and their inverses.

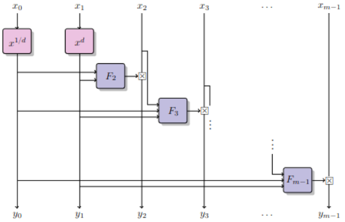
Most “AES-like”, also most secure at this stage.

ZK-Friendly Hash Functions

Poseidon [GLR+20] no specific constraints q
SPN with partial layer of low degree monomial S-boxes.
Full rounds – partial round – full rounds.

Rescue [AAB+20] q large prime
SPN with low degree monomials and their inverses.
Most "AES-like", also most secure at this stage.

Griffin [GHR+23] q large prime
Feistel variant with multiplications instead.
Particularly vulnerable to algebraic attacks!



source: *Cryptanalysis and design of symmetric primitives defined over large finite fields*, PhD thesis of C. Bouvier

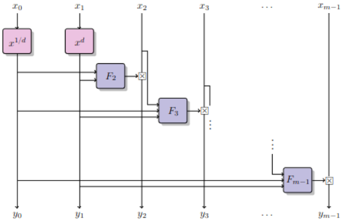
ZK-Friendly Hash Functions

Poseidon [GLR+20] no specific constraints q
SPN with partial layer of low degree monomial S-boxes.
Full rounds – partial round – full rounds.

Rescue [AAB+20] q large prime
SPN with low degree monomials and their inverses.
Most “AES-like”, also most secure at this stage.

Griffin [GHR+23] q large prime
Feistel variant with multiplications instead.
Particularly vulnerable to algebraic attacks!

Anemoi [BBC+23] $q = 2^n$ or large prime
Uses the “Flystel”, a high degree S-box
CCZ-equivalent to a function of low degree.



source: *Cryptanalysis and design of symmetric primitives defined over large finite fields*, PhD thesis of C. Bouvier