Transistor

a TFHE-friendly stream cipher

Jules Baudrin, Sonia Belaïd, Nicolas Bon, Christina Boura, Anne Canteaut, Gaëtan Leurent, Pascal Paillier, Léo Perrin, Matthieu Rivain, Yann Rotella, and Samuel Tap

CryptoExperts

WE INNOVATE TO SECURE YOUR BUSINESS

29th October 2024

Outline

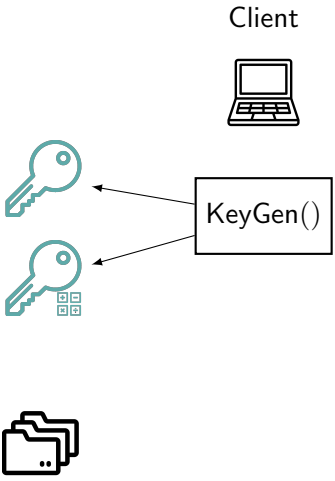
- FHE and the bandwith problem
- Transciphering
- TFHE and a wishlist
- Design of `Transistor`

FHE in one slide

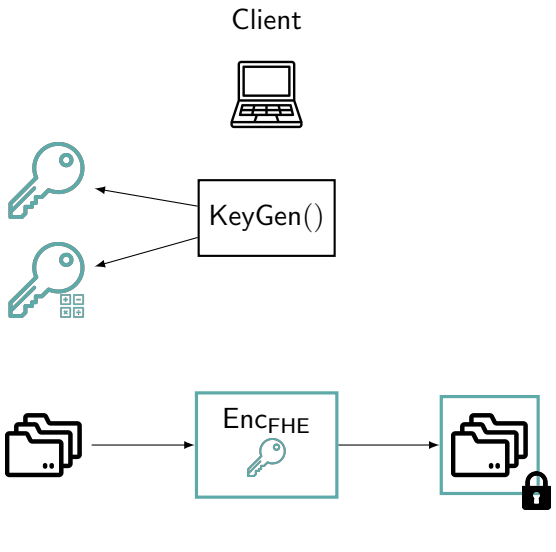
Client

Server

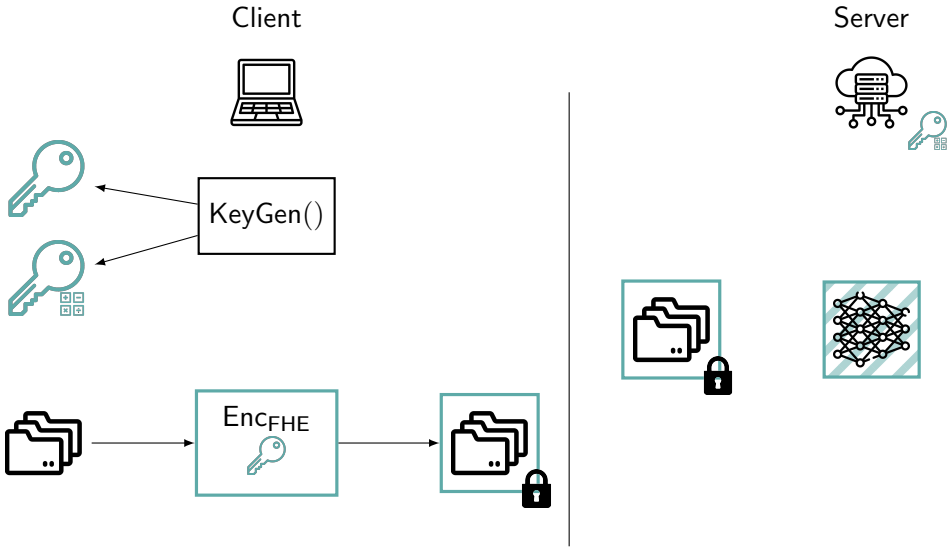
FHE in one slide



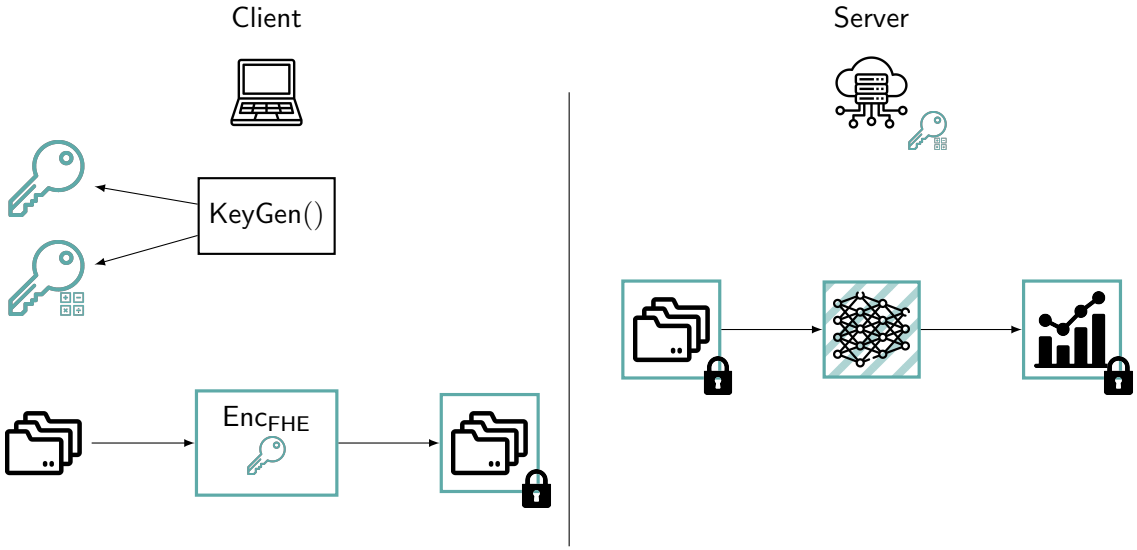
FHE in one slide



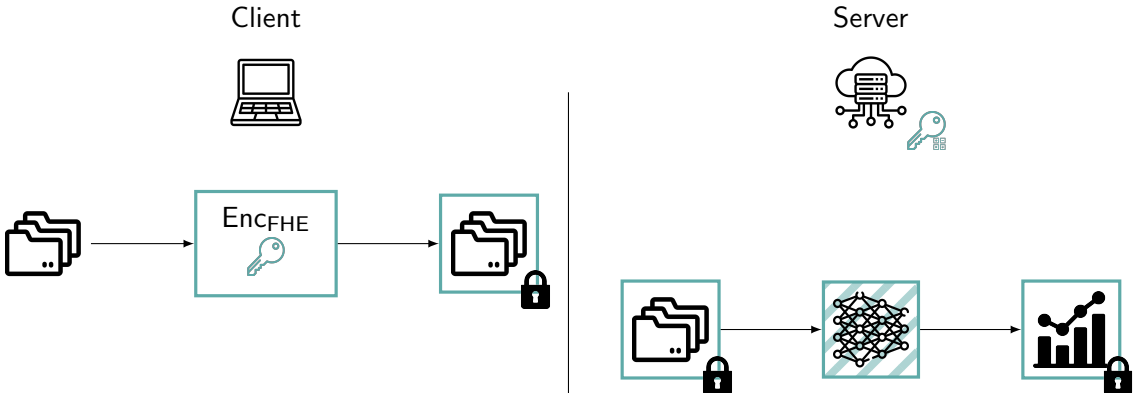
FHE in one slide



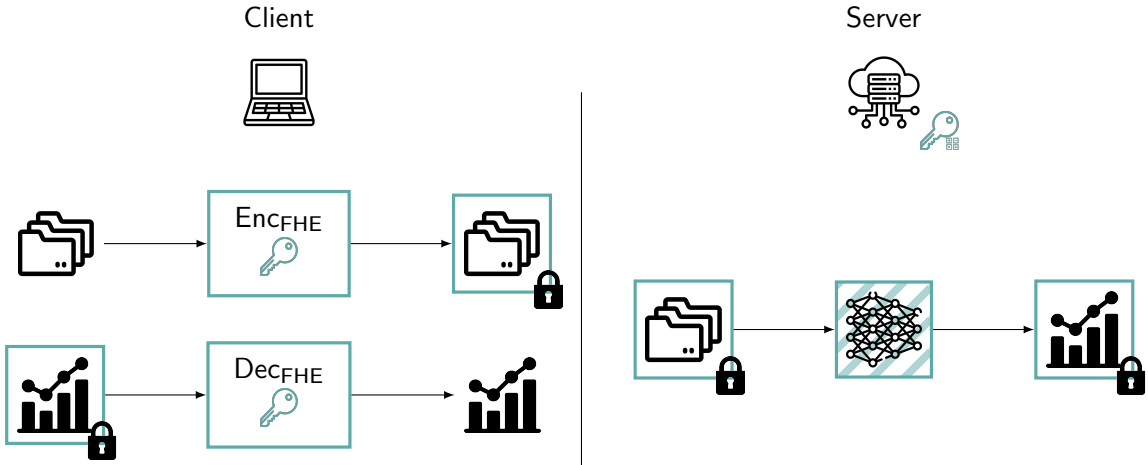
FHE in one slide



FHE in one slide



FHE in one slide



The Bandwidth Problem

Example: Layout of a TFHE ciphertext.

The Bandwidth Problem

Example: Layout of a TFHE ciphertext. To encrypt one bit m:

$$c = (a_0, \ldots, a_{n-1}, b)$$

The Bandwidth Problem

Example: Layout of a TFHE ciphertext. To encrypt one bit m:

$$c = (a_0, \ldots, a_{n-1}, b)$$

$(n+1)$ elements of $\mathbb{Z}_q$, with $q = 2^{64}$ and $500 \leq n \leq 1500$. Roughly, expansion factor of $1000 \times 64 = 64000$.

The Bandwidth Problem

Example: Layout of a TFHE ciphertext. To encrypt one bit m:

$$c = \left(\underbrace{a_0, \ldots, a_{n-1}}_{\text{Random}}, b \right)$$

$(n+1)$ elements of $\mathbb{Z}_q$, with $q = 2^{64}$ and $500 \leq n \leq 1500$. Roughly, expansion factor of $1000 \times 64 = 64000$.

Possible to generate the a_i's with a seed: expansion factor comes down to $128 + 64 = 192$.

The Bandwidth Problem

Example: Layout of a TFHE ciphertext. To encrypt one bit m:

$$c = \left(\underbrace{a_0, \ldots, a_{n-1}}_{\text{Random}}, b \right)$$

$(n+1)$ elements of $\mathbb{Z}_q$, with $q = 2^{64}$ and $500 \leq n \leq 1500$. Roughly, expansion factor of $1000 \times 64 = 64000$.

Possible to generate the a_i's with a seed: expansion factor comes down to $128 + 64 = 192$.

How to reduce this bandwith overhead ?

Outline

- FHE and the bandwith problem
- Transciphering
- TFHE and a wishlist
- Design of `Transistor`

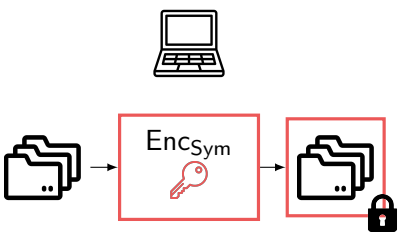
Transciphering

Client

Server

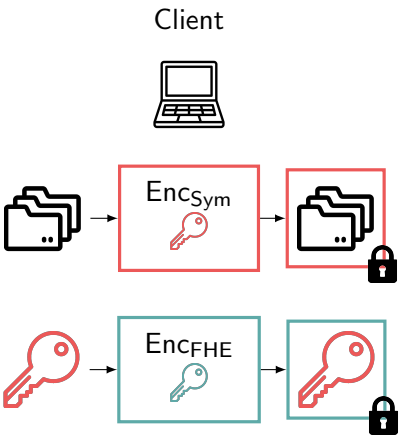
Transciphering

Client

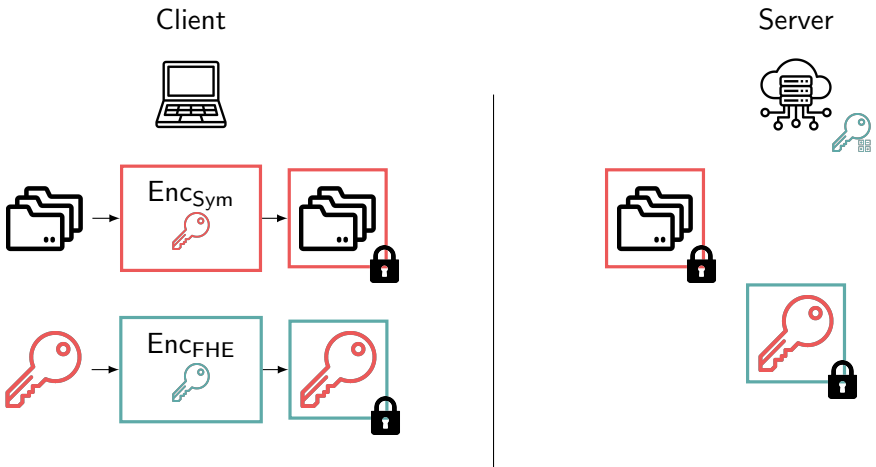


Server

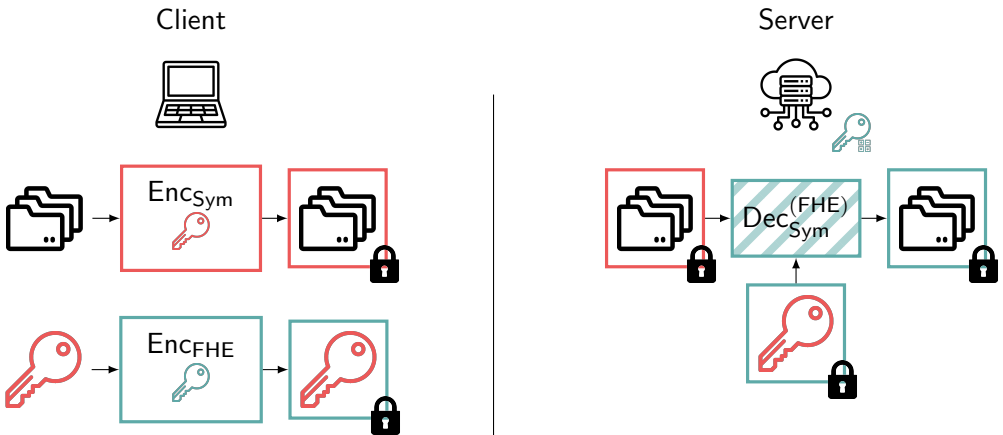
Transciphering



Transciphering



Transciphering



Which symmetric cipher ?

Using standard ciphers ?

- Evaluation of AES is too slow due to large Sbox size
- Lightweight ciphers such as ASCON, PRESENT, ... more promising, but still not fast enough

Which symmetric cipher ?

Using standard ciphers ?

- Evaluation of AES is too slow due to large Sbox size
- Lightweight ciphers such as ASCON, PRESENT, ... more promising, but still not fast enough

Or FHE-tailored ones ?

- Active line of research
- But often relies on innovative primitives (i.e. often broken)

Which symmetric cipher ?

Using standard ciphers ?

- Evaluation of AES is too slow due to large Sbox size
- Lightweight ciphers such as ASCON, PRESENT, ... more promising, but still not fast enough

Or FHE-tailored ones ?

- Active line of research
- But often relies on innovative primitives (i.e. often broken)

With `Transistor`, we look for the best of both worlds (fast in FHE and secure)

Outline

- FHE and the bandwith problem
- Transciphering
- TFHE and a wishlist
- Design of `Transistor`

TFHE

- Plaintext space : $\mathbb{Z}_p$ with p of a few bits.

TFHE

- Plaintext space : $\mathbb{Z}_p$ with p of a few bits.
- Secret Key: $\vec{s} = (s_0, \ldots, s_{n-1}) \in \{0,1\}^n$

TFHE

- Plaintext space : $\mathbb{Z}_p$ with p of a few bits.
- Secret Key: $\vec{s} = (s_0, \ldots, s_{n-1}) \in \{0,1\}^n$
- Encryption: $\vec{c} = (a_0, \ldots, a_{n-1}, b = \sum_{i=0}^{n-1} a_i \cdot s_i + \frac{q}{p} \cdot m + e) \in \mathbb{Z}_q^{n+1}$ with $\vec{a}$ random and e a small Gaussian noise.

TFHE

- Plaintext space : $\mathbb{Z}_p$ with p of a few bits.
- Secret Key: $\vec{s} = (s_0, \ldots, s_{n-1}) \in \{0,1\}^n$
- Encryption: $\vec{c} = (a_0, \ldots, a_{n-1}, b = \sum_{i=0}^{n-1} a_i \cdot s_i + \frac{q}{p} \cdot m + e) \in \mathbb{Z}_q^{n+1}$ with $\vec{a}$ random and e a small Gaussian noise.
- Linear homomorphism:
  - Sum of ciphertexts
  - Multiplication by a plaintext

  Very fast. **Increase the noise**.

TFHE

- Plaintext space : $\mathbb{Z}_p$ with p of a few bits.
- Secret Key: $\vec{s} = (s_0, \ldots, s_{n-1}) \in \{0,1\}^n$
- Encryption: $\vec{c} = (a_0, \ldots, a_{n-1}, b = \sum_{i=0}^{n-1} a_i \cdot s_i + \frac{q}{p} \cdot m + e) \in \mathbb{Z}_q^{n+1}$ with $\vec{a}$ random and e a small Gaussian noise.
- Linear homomorphism:
  - Sum of ciphertexts
  - Multiplication by a plaintext

  Very fast. **Increase the noise**.
- Programmable Bootstrapping (PBS):
  - Evaluates a **Look-Up Table**
  - **Resets** the noise to a nominal level

  Very slow. Gets even slower as p increases.

TFHE

- Plaintext space : $\mathbb{Z}_p$ with p of a few bits.
- Secret Key: $\vec{s} = (s_0, \ldots, s_{n-1}) \in \{0,1\}^n$
- Encryption: $\vec{c} = (a_0, \ldots, a_{n-1}, b = \sum_{i=0}^{n-1} a_i \cdot s_i + \frac{q}{p} \cdot m + e) \in \mathbb{Z}_q^{n+1}$ with $\vec{a}$ random and e a small Gaussian noise.
- Linear homomorphism:
  - Sum of ciphertexts
  - Multiplication by a plaintext

  Very fast. **Increase the noise**.
- Programmable Bootstrapping (PBS):
  - Evaluates a **Look-Up Table**
  - **Resets** the noise to a nominal level

  Very slow. Gets even slower as p increases.
- **Negacyclicity problem:**
  - If p is **even**, restricts some homomorphic operations.
  - Disappears when p is **odd**

Our wishlist for a TFHE-friendly cipher

Our wishlist for a TFHE-friendly cipher

- A prime field

Our wishlist for a TFHE-friendly cipher

- A prime field
- As little non-linear operations as possible

Our wishlist for a TFHE-friendly cipher

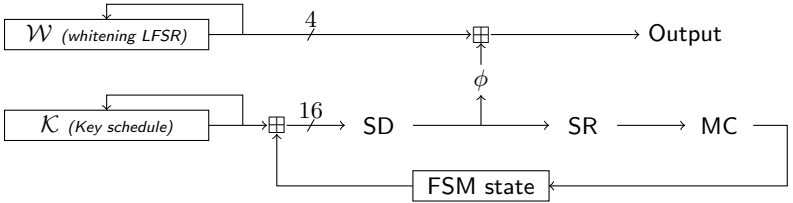
- A prime field
- As little non-linear operations as possible
- Keep noise growth acceptable

Outline

- FHE and the bandwith problem
- Transciphering
- TFHE and a wishlist
- Design of `Transistor`

Design of Transistor

Prime field: $\mathbb{F}_{17}$

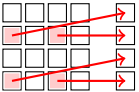


(a) General structure (rectangles correspond to registers).

(b) SD.

(c) SR.

(d) MC.



(e) ϕ.

Figure: A high level view of `Transistor`.

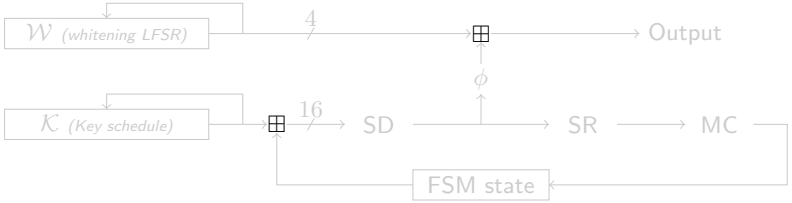
Design of Transistor

Our wishlist

- A prime field
- As little non-linear operations as possible
- Smallest noise growth possible

Design of Transistor

Prime field: $\mathbb{F}_{17}$



(a) General structure (rectangles correspond to registers).

(b) SD.

(c) SR.

(d) MC.

(e) ϕ.

Figure: A high level view of `Transistor`.

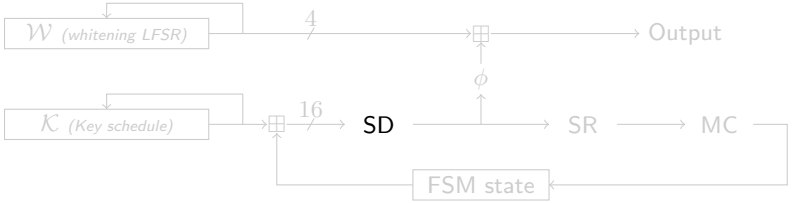
Design of Transistor

Our wishlist

- A prime field
- As little non-linear operations as possible
- Smallest noise growth possible

Design of Transistor

Prime field: $\mathbb{F}_{17}$



(a) General structure (rectangles correspond to registers).

(b) SD.

(c) SR.

(d) MC.

(e) ϕ.

Figure: A high level view of `Transistor`.

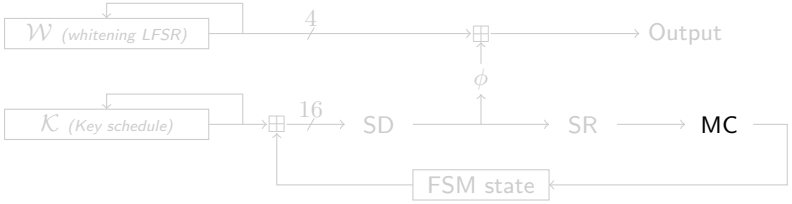
Design of Transistor

Our wishlist

- A prime field
- As little non-linear operations as possible
- Smallest noise growth possible

Design of Transistor

Prime field: $\mathbb{F}_{17}$



(a) General structure (rectangles correspond to registers).

(b) SD.

(c) SR.

(d) MC.

(e) ϕ.

Figure: A high level view of `Transistor`.

MixColumns

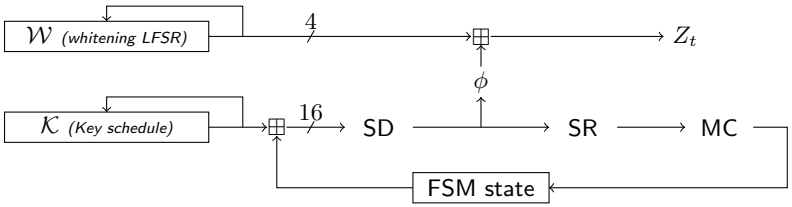
We opted for a 4×4 Maximum Distance Separable (MDS) to ensure optimal diffusion. The matrix we chose is

$$M = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & -1 & 1 & -2 \\ 1 & 1 & -2 & -1 \\ 1 & -2 & -1 & 1 \end{bmatrix} . \tag{1}$$

We verified that there is no MDS matrix in $\mathbb{F}_{17}$ with coefficients in $\{-1, 1\}$ by exhaustively testing all such matrices. By testing all matrices with coefficients in $\{-2, -1, 1, 2\}$, we found a total of $30\,720$ MDS matrices with an ℓ_2-norm of 7. We selected M for its symmetries, particularly because it is its own transpose.

Design of Transistor

Prime field: $\mathbb{F}_{17}$

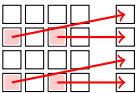


(a) General structure (rectangles correspond to registers).

(b) SD.

(c) SR.

(d) MC.

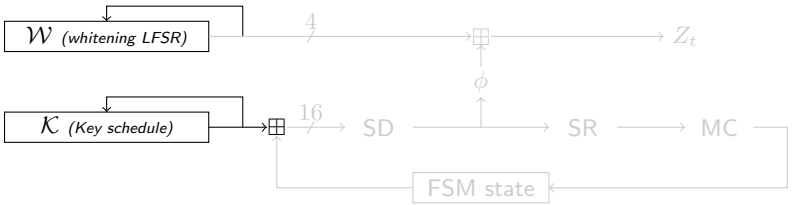


(e) ϕ.

Figure: A high level view of `Transistor`.

Design of Transistor

Prime field: $\mathbb{F}_{17}$



(a) General structure (rectangles correspond to registers).

(b) SD.

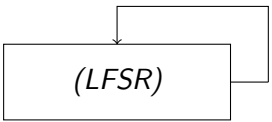
(c) SR.

(d) MC.

(e) ϕ.

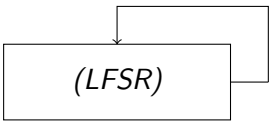
Figure: A high level view of `Transistor`.

Silent LFSR



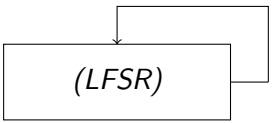
- Naive approach would be to maintain an encrypted state, and update it by computing a linear combination with the feedback coefficients.

Silent LFSR



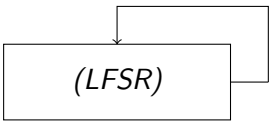
- Naive approach would be to maintain an encrypted state, and update it by computing a linear combination with the feedback coefficients.
- However, this method would cause the noise in the state to accumulate over time

Silent LFSR



- Naive approach would be to maintain an encrypted state, and update it by computing a linear combination with the feedback coefficients.
- However, this method would cause the noise in the state to accumulate over time
- Solution: Computing on the fly the coefficients of the linear combination in clear

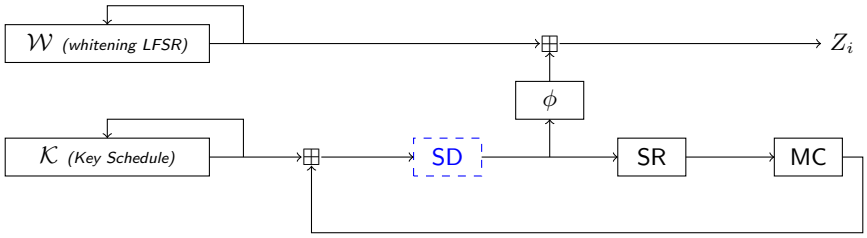
Silent LFSR



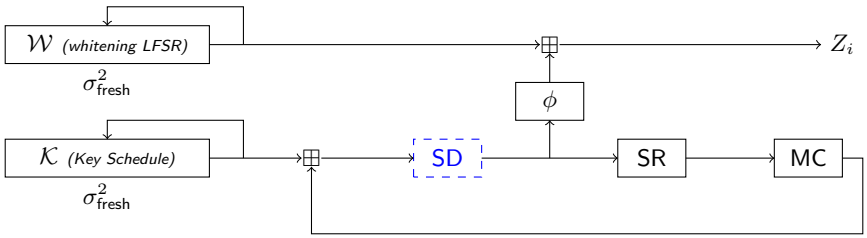
- Naive approach would be to maintain an encrypted state, and update it by computing a linear combination with the feedback coefficients.
- However, this method would cause the noise in the state to accumulate over time
- Solution: Computing on the fly the coefficients of the linear combination in clear

The noise variance in the output of the silent LFSR remains stable over time, without using any PBS.

Noise Management

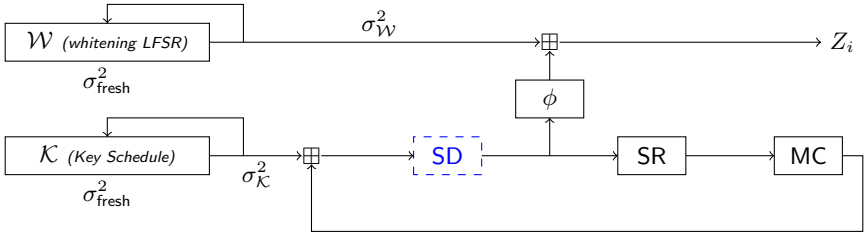


Noise Management



The LFSR are loaded with fresh ciphertexts of noise σ_{fresh}

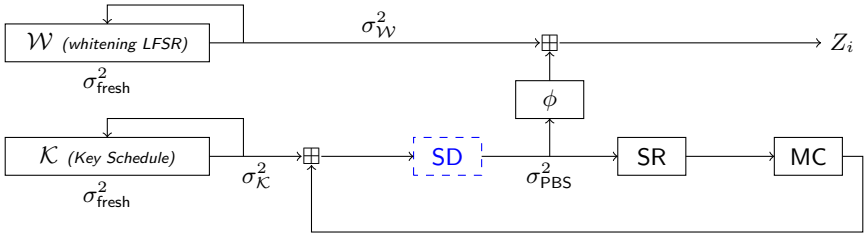
Noise Management



The output of *silent* LFSRs have noise

$$\sigma^2_{\mathcal{K}} \leq |\mathcal{K}| \cdot \left(\frac{p-1}{2}\right)^2 \cdot \sigma^2_{\text{fresh}} \quad \text{and} \quad \sigma^2_{\mathcal{W}} \leq |\mathcal{W}| \cdot \left(\frac{p-1}{2}\right)^2 \cdot \sigma^2_{\text{fresh}}$$

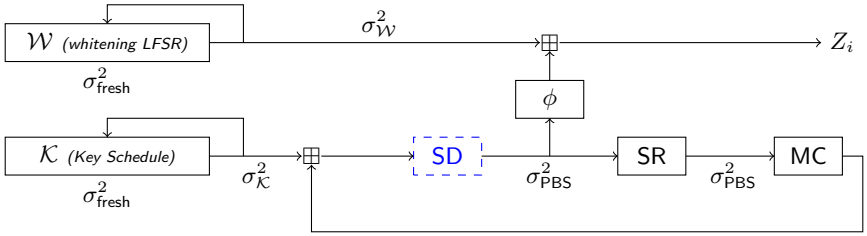
Noise Management



The output of a PBS has noise **independent** from the noise of the input. We just have:

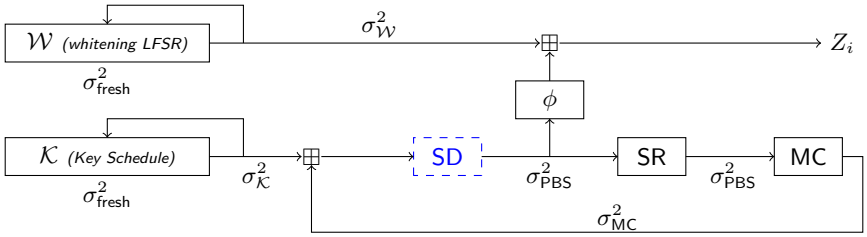
$$\sigma_{\text{PBS}} \gg \sigma_{\text{fresh}}$$

Noise Management



ShiftRows simply reorders the ciphertexts, thus adds no noise.

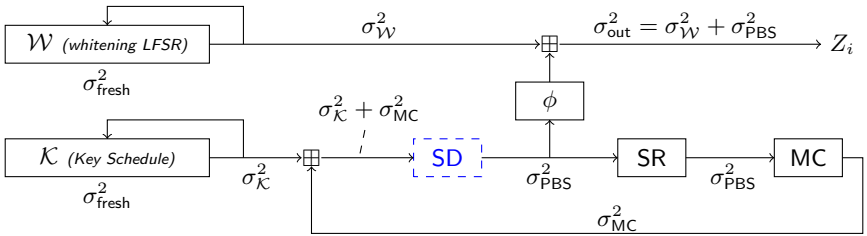
Noise Management



The variance is multiplied by the norm of the diffusion matrix

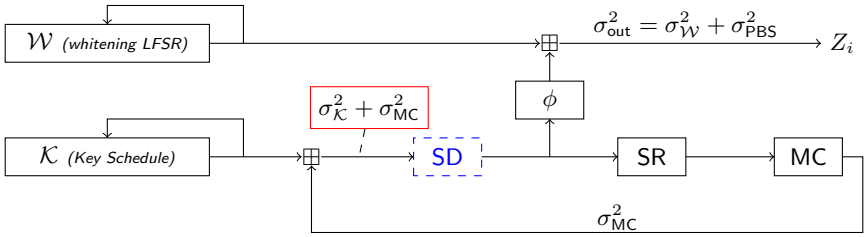
$$\sigma^2_{\text{MC}} = L^2_{\text{MC}} \cdot \sigma^2_{\text{PBS}}.$$

Noise Management



Variances are summed.

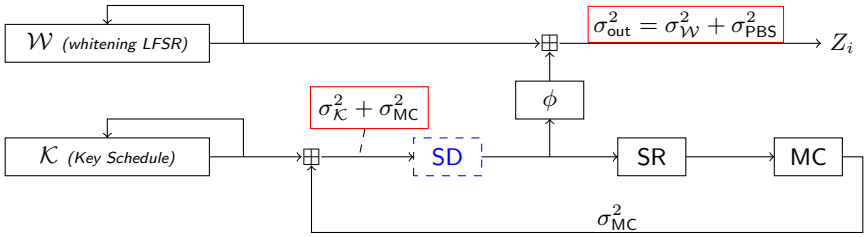
Noise Management



There are two "hot spots" where things can go wrong:

- Right before the PBS (risk of false bootstrapping result)

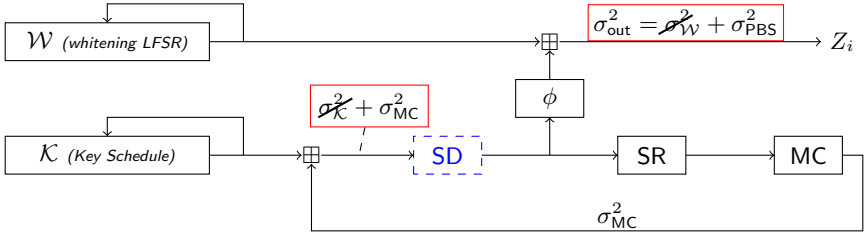
Noise Management



There are two "hot spots" where things can go wrong:

- Right before the PBS (risk of false bootstrapping result)
- At the output of the cipher (risk of producing a keystream of noise too high, thus unusable)

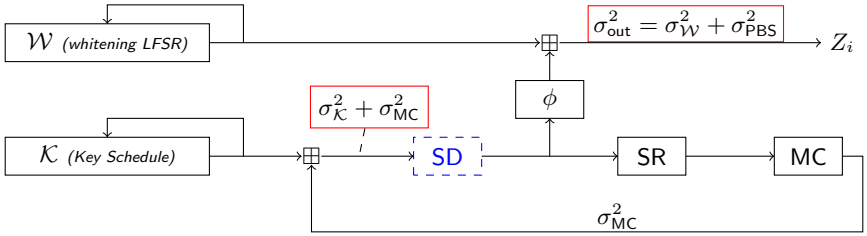
Noise Management



As $\sigma_{\text{MC}} \gg \sigma^2_{\mathcal{K}}$ and $\sigma_{\text{PBS}} \gg \sigma^2_{\mathcal{W}}$, the noise produced by the LFSRs is negligible.

Takeaway 1: No Restriction on the size of the LFSRs

Noise Management



Takeaway 2 : Dimensioning the TFHE parameters for `Transistor` can be reduced to select parameters for a simple PBS

Performances

Table: Execution timings of `FRAST` and `Transistor`.

Cipher	p_{err}	Setup	Latency	Throughput
`FRAST`	2^{-80}	25 s (8 threads)	6.2 s	20.66 bits/s
`Transistor`	2^{-128}	No	251 ms	65.10 bits/s

Bonus: Takeways on cryptanalysis (Disclaimer: I have no clue)

A quick tour of the security arguments:

- Time-Memory-Data Trade-Offs put a limit on the length of the keystream we can generate with a single key: 2^{31}.
- Guess-and-Determine: as one quarter of the state is used to generate the keystream, one has to guess $\frac{3}{4}$ of the LFSR, which puts a lower bound on its size. $|\mathcal{K}| = 64$ and $\mathcal{W}| = 32$
- The paper demonstrates that **Three consecutive outputs are statistically independent of the secret key**. The proof only requires the matrix of MixColumns to be MDS instead of automatic search methods like in `Rocca`.
- Linear approximation, Fast Correlation attacks, Algebraic Attacks