Code-Based Cryptography for FPGAs

Dr. Ruben Niederhagen, February 8, 2018

Fraunhofer SIT

Introduction
Global Map

public-key cryptography

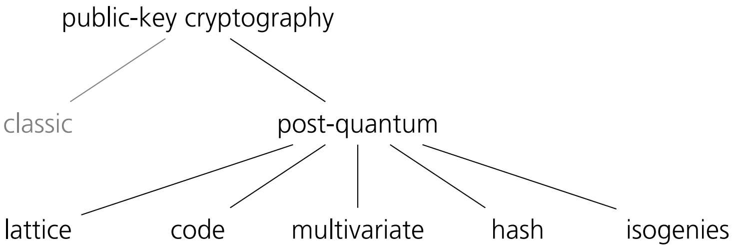
Introduction
Global Map

public-key cryptography

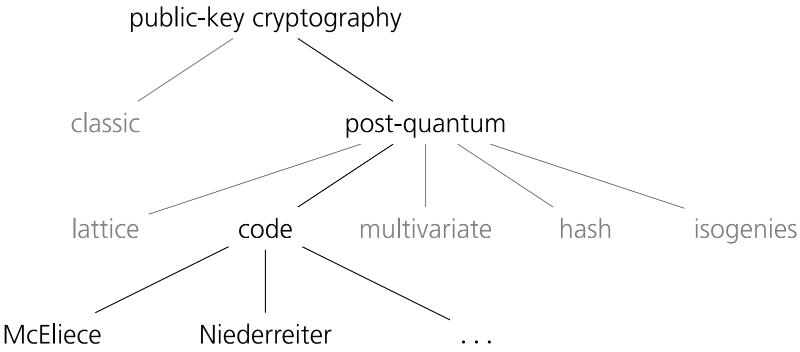
classic

post-quantum

Introduction
Global Map

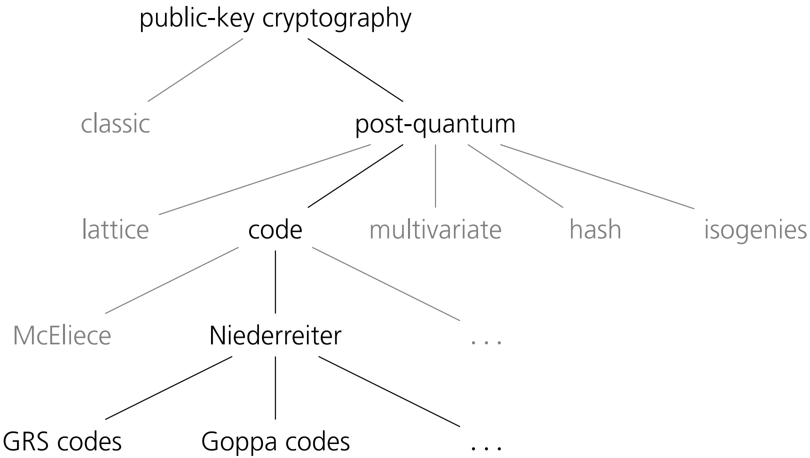


Introduction
Global Map

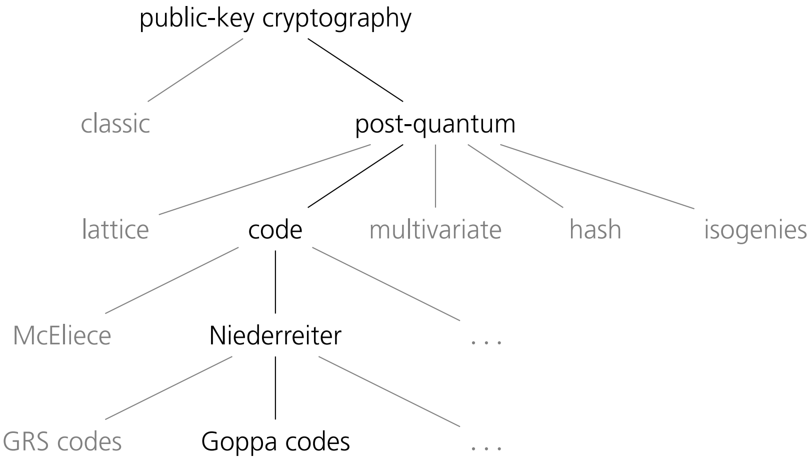


Fraunhofer SIT

Introduction
Global Map



Introduction
Global Map



Introduction
Motivation

Why code-based schemes in hardware?

Introduction
Motivation

Why code-based schemes in hardware?

- Code-based schemes are well-understood:
  - Long history of research.

Why code-based schemes in hardware?

- Code-based schemes are well-understood:
  - Long history of research.
  - Security parameters widely accepted.

Introduction
Motivation

Why code-based schemes in hardware?

- Code-based schemes are well-understood:
  - Long history of research.
  - Security parameters widely accepted.
- Code-based schemes are expensive:
  - High-throughput scenario: web server...

Introduction
Motivation

Why code-based schemes in hardware?

- Code-based schemes are well-understood:
  - Long history of research.
  - Security parameters widely accepted.
- Code-based schemes are expensive:
  - High-throughput scenario: web server...
  - Low-energy scenario: embedded devices, SmartCards, ...

Introduction
Motivation

Why code-based schemes in hardware?

- Code-based schemes are well-understood:
  - Long history of research.
  - Security parameters widely accepted.
- Code-based schemes are expensive:
  - High-throughput scenario: web server...
  - Low-energy scenario: embedded devices, SmartCards, ...

$\Longrightarrow$ Hardware implementation as accelerator and for efficiency.

Introduction
Error-Correcting Codes — McEliece and Niederreiter

01101100

Introduction

Error-Correcting Codes — McEliece and Niederreiter

01101100

encode

10011001001

Introduction

Error-Correcting Codes — McEliece and Niederreiter

01101100

encode

10011001001 - - - - - - - - - - - - - - -▸ 10010001011

transmitt

Introduction
Error-Correcting Codes — McEliece and Niederreiter

01101100

encode

10011001001 transmitt 10010001011 error correction 10011001001

Introduction

Error-Correcting Codes — McEliece and Niederreiter

01101100

encode

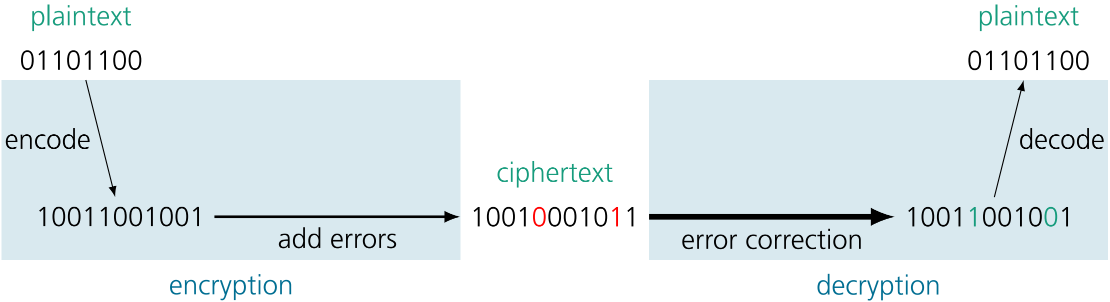
10011001001 - - - transmitt - - -> 10010001011 — error correction —> 10011001001

decode

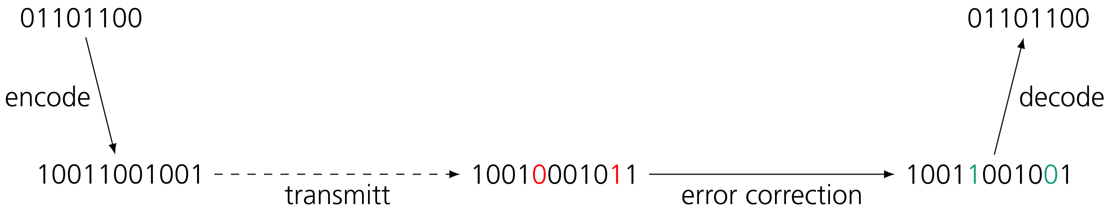
01101100

Introduction

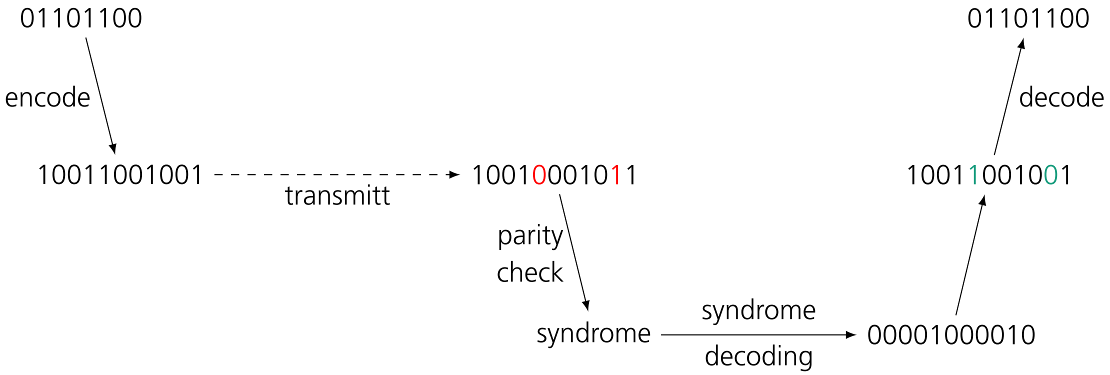
Error-Correcting Codes — McEliece and Niederreiter



Introduction
Error-Correcting Codes — McEliece and Niederreiter

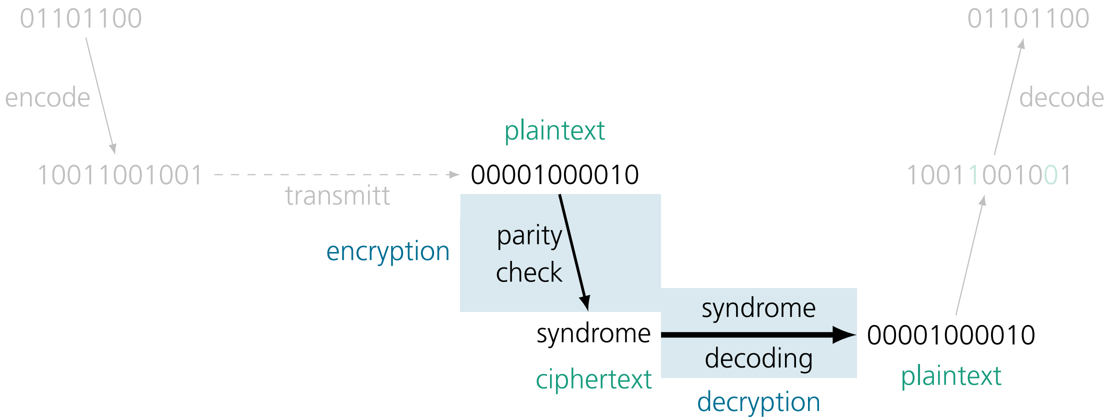


Introduction
Error-Correcting Codes — McEliece and Niederreiter



Fraunhofer SIT

Introduction
Error-Correcting Codes — McEliece and Niederreiter



Niederreiter Cryptosystem

Algorithm 1: Key-generation algorithm for the Niederreiter cryptosystem.

Input : System parameters: m, t, and n.

Output: Private key $(g(x), (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}))$ and public key K.

Choose random sequence $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \in \mathbb{F}(2^m)^n$ of distinct elements.

Niederreiter Cryptosystem

Algorithm 1: Key-generation algorithm for the Niederreiter cryptosystem.

Input : System parameters: m, t, and n.

Output: Private key $(g(x), (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}))$ and public key K.

Choose random sequence $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \in \mathbb{F}(2^m)^n$ of distinct elements.

Choose a random irreducible polynomial $g(x)$ of degree t.

Niederreiter Cryptosystem

Algorithm 1: Key-generation algorithm for the Niederreiter cryptosystem.

Input : System parameters: m, t, and n.

Output: Private key $(g(x), (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}))$ and public key K.

Choose random sequence $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \in \mathbb{F}(2^m)^n$ of distinct elements.

Choose a random irreducible polynomial $g(x)$ of degree t.

Compute the $t \times n$ parity check matrix

$$H = \begin{bmatrix} 1/g(\alpha_0) & 1/g(\alpha_1) & \cdots & 1/g(\alpha_{n-1}) \\ \alpha_0/g(\alpha_0) & \alpha_1/g(\alpha_1) & \cdots & \alpha_{n-1}/g(\alpha_{n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_0^{t-1}/g(\alpha_0) & \alpha_1^{t-1}/g(\alpha_1) & \cdots & \alpha_{n-1}^{t-1}/g(\alpha_{n-1}) \end{bmatrix}.$$

Niederreiter Cryptosystem

Algorithm 1: Key-generation algorithm for the Niederreiter cryptosystem.

Input : System parameters: m, t, and n.

Output: Private key $(g(x), (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}))$ and public key K.

1 Choose random sequence $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \in \mathbb{F}(2^m)^n$ of distinct elements.

2 Choose a random irreducible polynomial $g(x)$ of degree t.

3 Compute the $t \times n$ parity check matrix

$$H = \begin{bmatrix} 1/g(\alpha_0) & 1/g(\alpha_1) & \cdots & 1/g(\alpha_{n-1}) \\ \alpha_0/g(\alpha_0) & \alpha_1/g(\alpha_1) & \cdots & \alpha_{n-1}/g(\alpha_{n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_0^{t-1}/g(\alpha_0) & \alpha_1^{t-1}/g(\alpha_1) & \cdots & \alpha_{n-1}^{t-1}/g(\alpha_{n-1}) \end{bmatrix}.$$

4 Transform H to a $mt \times n$ binary parity check matrix H'.

Niederreiter Cryptosystem

Algorithm 1: Key-generation algorithm for the Niederreiter cryptosystem.

Input : System parameters: m, t, and n.

Output: Private key $(g(x), (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}))$ and public key K.

1 Choose random sequence $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \in \mathbb{F}(2^m)^n$ of distinct elements.

2 Choose a random irreducible polynomial $g(x)$ of degree t.

3 Compute the $t \times n$ parity check matrix

$$H = \begin{bmatrix} 1/g(\alpha_0) & 1/g(\alpha_1) & \cdots & 1/g(\alpha_{n-1}) \\ \alpha_0/g(\alpha_0) & \alpha_1/g(\alpha_1) & \cdots & \alpha_{n-1}/g(\alpha_{n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_0^{t-1}/g(\alpha_0) & \alpha_1^{t-1}/g(\alpha_1) & \cdots & \alpha_{n-1}^{t-1}/g(\alpha_{n-1}) \end{bmatrix}.$$

4 Transform H to a $mt \times n$ binary parity check matrix H'.

5 Transform H' into its systematic form $[\mathbb{I}_{mt}|K]$.

Niederreiter Cryptosystem

Algorithm 1: Key-generation algorithm for the Niederreiter cryptosystem.

Input : System parameters: m, t, and n.

Output: Private key $(g(x), (\alpha_0, \alpha_1, \dots, \alpha_{n-1}))$ and public key K.

Choose random sequence $(\alpha_0, \alpha_1, \dots, \alpha_{n-1}) \in \mathbb{F}(2^m)^n$ of distinct elements.

Choose a random irreducible polynomial $g(x)$ of degree t.

Compute the $t \times n$ parity check matrix

$$\begin{bmatrix} 1/g(\alpha_0) & 1/g(\alpha_1) & \cdots & 1/g(\alpha_{n-1}) \\ \alpha_0/g(\alpha_0) & \alpha_1/g(\alpha_1) & \cdots & \alpha_{n-1}/g(\alpha_{n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_0^{t-1}/g(\alpha_0) & \alpha_1^{t-1}/g(\alpha_1) & \cdots & \alpha_{n-1}^{t-1}/g(\alpha_{n-1}) \end{bmatrix}.$$

Permute list of all 2^m elements, pick the first n elements.

Transform H to a $mt \times n$ binary parity check matrix H'.

Transform H' into its systematic form $[\mathbb{I}_{mt}|K]$.

Niederreiter Cryptosystem

Permute list of all 2^m elements, pick the first n elements.

- **Option 1:** Use Fisher-Yates shuffle.
  - Biased if not well implemented,
  - non-biased implementations need floating-point arithmetic or are not constant time.

Niederreiter Cryptosystem

Permute list of all 2^m elements, pick the first n elements.

- **Option 1:** Use Fisher-Yates shuffle.
  - Biased if not well implemented,
  - non-biased implementations need floating-point arithmetic or are not constant time.
- **Option 2:** Use a constant-time sorting algorithm.
  Sample 2^m random 32-bit values r_i.
  Generate a list of tuples $\{(r_0, 0), (r_i, 1), \ldots, (r_{2^m-1}, a^{m-1} + a^{m-2} \cdots + a + 1)\}$.
  Sort list by the first element.
  Obtain the permutation by reading the second elements.
  - Expensive: more cycles, more logic.

Niederreiter Cryptosystem

Algorithm 1: Key-generation algorithm for the Niederreiter cryptosystem.

Input : System parameters: m, t, and n.

Output: Private key $(g(x), (\alpha_0, \alpha_1, \dots, \alpha_{n-1}))$ and public key K.

1 Choose random sequence $(\alpha_0, \alpha_1, \dots, \alpha_{n-1}) \in \mathbb{F}(2^m)^n$ of distinct elements.

2 Choose a random irreducible polynomial $g(x)$ of degree t.

3 Compute the $t \times n$ parity check matrix

$$\begin{bmatrix} 1/g(\alpha_0) & 1/g(\alpha_1) & \cdots & 1/g(\alpha_{n-1}) \\ & & & \alpha_{n-1}/g(\alpha_{n-1}) \\ \vdots & \vdots & & \vdots \\ \alpha_0^{t-1}/g(\alpha_0) & \alpha_1^{t-1}/g(\alpha_1) & \cdots & \alpha_{n-1}^{t-1}/g(\alpha_{n-1}) \end{bmatrix}.$$

Permute list of all 2^m elements, pick the first n elements.

4 Transform H to a $mt \times n$ binary parity check matrix H'.

5 Transform H' into its systematic form $[\mathbb{I}_{mt}|K]$.

Niederreiter Cryptosystem

Algorithm 1: Key-generation algorithm for the Niederreiter cryptosystem.

Input : System parameters: m, t, and n.

Output: Private key $(g(x), (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}))$ and public key K.

Choose random sequence $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \in \mathbb{F}(2^m)^n$ of distinct elements.

Choose a random irreducible polynomial $g(x)$ of degree t.

Compute the $t \times n$ parity check matrix

$$\begin{bmatrix} 1/g(\alpha_0) & 1/g(\alpha_1) & \cdots & 1/g(\alpha_{n-1}) \\ \alpha_0/g(\alpha_0) & \alpha_1/g(\alpha_1) & \cdots & \alpha_{n-1}/g(\alpha_{n-1}) \\ & & \ddots & \vdots \\ \alpha_0^{t-1}/g(\alpha_0) & \alpha_1^{t-1}/g(\alpha_1) & \cdots & \alpha_{n-1}^{t-1}/g(\alpha_{n-1}) \end{bmatrix}.$$

Generate an irreducible polynomial.

Transform H to a $mt \times n$ binary parity check matrix H'.

Transform H' into its systematic form $[\mathbb{I}_{mt}|K]$.

Niederreiter Cryptosystem

Generate an irreducible polynomial of degree t.

- **Option 1:** Randomly chose $t + 1$ coefficients, check if obtained polynomial is irreducible.
  - Needs about t iterations $\Rightarrow$ not constant time,
  - checking for irreducibility is expensive (extended Euclidean algorithm).

Niederreiter Cryptosystem

Generate an irreducible polynomial of degree t.

- **Option 1:** Randomly chose $t+1$ coefficients, check if obtained polynomial is irreducible.
  - Needs about t iterations $\Rightarrow$ not constant time,
  - checking for irreducibility is expensive (extended Euclidean algorithm).
- **Option 2:** Construct an irreducible polynomial.
  - Idea: Compute minimal polynomial of an element $r \in \mathbb{F}(2^m)[x]/f$ with deg$(f) = t$.
  - Compute several powers in $\mathbb{F}(2^m)[x]/f$,
  - solve a linear equation system over $\mathbb{F}(2^m)$ of dimension $t \times t+1$.

Niederreiter Cryptosystem

Algorithm 1: Key-generation algorithm for the Niederreiter cryptosystem.

Input : System parameters: m, t, and n.

Output: Private key $(g(x), (\alpha_0, \alpha_1, \dots, \alpha_{n-1}))$ and public key K.

1. Choose random sequence $(\alpha_0, \alpha_1, \dots, \alpha_{n-1}) \in \mathbb{F}(2^m)^n$ of distinct elements.
2. Choose a random irreducible polynomial $g(x)$ of degree t.
3. Compute the $t \times n$ parity check matrix

$$\begin{bmatrix} 1/g(\alpha_0) & 1/g(\alpha_1) & \cdots & 1/g(\alpha_{n-1}) \\ \alpha_0/g(\alpha_0) & \alpha_1/g(\alpha_1) & \cdots & \alpha_{n-1}/g(\alpha_{n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_0^{t-1}/g(\alpha_0) & \alpha_1^{t-1}/g(\alpha_1) & \cdots & \alpha_{n-1}^{t-1}/g(\alpha_{n-1}) \end{bmatrix}.$$

Generate an irreducible polynomial.

4. Transform H to a $mt \times n$ binary parity check matrix H'.
5. Transform H' into its systematic form $[\mathbb{I}_{mt}|K]$.

Niederreiter Cryptosystem

Algorithm 1: Key-generation algorithm for the Niederreiter cryptosystem.

Input : System parameters: m, t, and n.

Output: Private key $(g(x), (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}))$ and public key K.

1 Choose random sequence $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \in \mathbb{F}(2^m)^n$ of distinct elements.

2 Choose a random irreducible polynomial $g(x)$ of degree t.

3 Compute the $t \times n$ parity check matrix

$$H = \begin{bmatrix} 1/g(\alpha_0) & 1/g(\alpha_1) & \cdots & 1/g(\alpha_{n-1}) \\ \alpha_0/g(\alpha_0) & \alpha_1/g(\alpha_1) & \cdots & \alpha_{n-1}/g(\alpha_{n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ & & & \alpha_{n-1}^{t-1}/g(\alpha_{n-1}) \end{bmatrix}.$$

Evaluate g at all 2^m elements using additive FFT.

4 Transform H to a $mt \times n$ binary parity check matrix H'.

5 Transform H' into its systematic form $[\mathbb{I}_{mt}|K]$.

Fraunhofer SIT

Niederreiter Cryptosystem

Algorithm 1: Key-generation algorithm for the Niederreiter cryptosystem.

Input : System parameters: m, t, and n.

Output: Private key $(g(x), (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}))$ and public key K.

1 Choose random sequence $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \in \mathbb{F}(2^m)^n$ of distinct elements.

2 Choose a random irreducible polynomial $g(x)$ of degree t.

3 Compute the $t \times n$ parity check matrix

$$\begin{bmatrix} 1/g(\alpha_0) & 1/g(\alpha_1) & \cdots & 1/g(\alpha_{n-1}) \\ \alpha_0/g(\alpha_0) & \alpha_1/g(\alpha_1) & \cdots & \alpha_{n-1}/g(\alpha_{n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_0^{t-1}/g(\alpha_0) & \alpha_1^{t-1}/g(\alpha_1) & \cdots & \alpha_{n-1}^{t-1}/g(\alpha_{n-1}) \end{bmatrix}.$$

Gaussian elimination.

4 Transform H to a $mt \times n$ binary parity check matrix H'.

5 Transform H' into its systematic form $[\mathbb{I}_{mt}|K]$.

Niederreiter Cryptosystem

Algorithm 2: Encryption algorithm for the Niederreiter cryptosystem.

Input : Plaintext e, public key K.

Output: Ciphertext c.

Compute $c = [\mathbb{I}_{mt}|K] \times e$.

Niederreiter Cryptosystem

Algorithm 2: Encryption algorithm for the Niederreiter cryptosystem.

Input : Plaintext e, public key K.

Output: Ciphertext c.

Compute $c = [\mathbb{I}_{mt}|K] \times e$.

Return the ciphertext c.

Niederreiter Cryptosystem

Algorithm 3: Decryption algorithm for the Niederreiter cryptosystem.

Input : Ciphertext c, secret key $(g(x), (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}))$.

Output: Plaintext e.

Compute the double-size $2t \times n$ parity check matrix

$$H^{(2)} = \begin{bmatrix} 1/g^2(\alpha_0) & 1/g^2(\alpha_1) & \cdots & 1/g^2(\alpha_{n-1}) \\ \alpha_0/g^2(\alpha_0) & \alpha_1/g^2(\alpha_1) & \cdots & \alpha_{n-1}/g^2(\alpha_{n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_0^{2t-1}/g^2(\alpha_0) & \alpha_1^{2t-1}/g^2(\alpha_1) & \cdots & \alpha_{n-1}^{2t-1}/g^2(\alpha_{n-1}) \end{bmatrix}.$$

Niederreiter Cryptosystem

Algorithm 3: Decryption algorithm for the Niederreiter cryptosystem.

Input : Ciphertext c, secret key $(g(x), (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}))$.

Output: Plaintext e.

1 Compute the double-size $2t \times n$ parity check matrix

$$H^{(2)} = \begin{bmatrix} 1/g^2(\alpha_0) & 1/g^2(\alpha_1) & \cdots & 1/g^2(\alpha_{n-1}) \\ \alpha_0/g^2(\alpha_0) & \alpha_1/g^2(\alpha_1) & \cdots & \alpha_{n-1}/g^2(\alpha_{n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_0^{2t-1}/g^2(\alpha_0) & \alpha_1^{2t-1}/g^2(\alpha_1) & \cdots & \alpha_{n-1}^{2t-1}/g^2(\alpha_{n-1}) \end{bmatrix}.$$

2 Transform $H^{(2)}$ to a $2mt \times n$ binary parity check matrix $H'^{(2)}$.

Niederreiter Cryptosystem

Algorithm 3: Decryption algorithm for the Niederreiter cryptosystem.

Input : Ciphertext c, secret key $(g(x), (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}))$.

Output: Plaintext e.

Compute the double-size $2t \times n$ parity check matrix

$$H^{(2)} = \begin{bmatrix} 1/g^2(\alpha_0) & 1/g^2(\alpha_1) & \cdots & 1/g^2(\alpha_{n-1}) \\ \alpha_0/g^2(\alpha_0) & \alpha_1/g^2(\alpha_1) & \cdots & \alpha_{n-1}/g^2(\alpha_{n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_0^{2t-1}/g^2(\alpha_0) & \alpha_1^{2t-1}/g^2(\alpha_1) & \cdots & \alpha_{n-1}^{2t-1}/g^2(\alpha_{n-1}) \end{bmatrix}.$$

Transform $H^{(2)}$ to a $2mt \times n$ binary parity check matrix $H'^{(2)}$.

Compute the double-size syndrome: $S^{(2)} = H'^{(2)} \times (c|0)$.

Fraunhofer SIT

Niederreiter Cryptosystem

Algorithm 3: Decryption algorithm for the Niederreiter cryptosystem.

Input : Ciphertext c, secret key $(g(x), (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}))$.

Output: Plaintext e.

1. Compute the double-size $2t \times n$ parity check matrix

$$H^{(2)} = \begin{bmatrix} 1/g^2(\alpha_0) & 1/g^2(\alpha_1) & \cdots & 1/g^2(\alpha_{n-1}) \\ \alpha_0/g^2(\alpha_0) & \alpha_1/g^2(\alpha_1) & \cdots & \alpha_{n-1}/g^2(\alpha_{n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_0^{2t-1}/g^2(\alpha_0) & \alpha_1^{2t-1}/g^2(\alpha_1) & \cdots & \alpha_{n-1}^{2t-1}/g^2(\alpha_{n-1}) \end{bmatrix}.$$

2. Transform $H^{(2)}$ to a $2mt \times n$ binary parity check matrix $H'^{(2)}$.
3. Compute the double-size syndrome: $S^{(2)} = H'^{(2)} \times (c|0)$.
4. Compute the error-locator polynomial $\sigma(x)$ from $S^{(2)}$.

Niederreiter Cryptosystem

Algorithm 3: Decryption algorithm for the Niederreiter cryptosystem.

Input : Ciphertext c, secret key $(g(x), (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}))$.

Output: Plaintext e.

Compute the double-size $2t \times n$ parity check matrix

$$H^{(2)} = \begin{bmatrix} 1/g^2(\alpha_0) & 1/g^2(\alpha_1) & \cdots & 1/g^2(\alpha_{n-1}) \\ \alpha_0/g^2(\alpha_0) & \alpha_1/g^2(\alpha_1) & \cdots & \alpha_{n-1}/g^2(\alpha_{n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_0^{2t-1}/g^2(\alpha_0) & \alpha_1^{2t-1}/g^2(\alpha_1) & \cdots & \alpha_{n-1}^{2t-1}/g^2(\alpha_{n-1}) \end{bmatrix}.$$

Transform $H^{(2)}$ to a $2mt \times n$ binary parity check matrix $H'^{(2)}$.

Compute the double-size syndrome: $S^{(2)} = H'^{(2)} \times (c|0)$.

Compute the error-locator polynomial $\sigma(x)$ from $S^{(2)}$.

Evaluate the error-locator polynomial $\sigma(x)$ at $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$.

Niederreiter Cryptosystem

Algorithm 3: Decryption algorithm for the Niederreiter cryptosystem.

Input : Ciphertext c, secret key $(g(x), (\alpha_0, \alpha_1, \dots, \alpha_{n-1}))$.

Output: Plaintext e.

1 Compute the double-size $2t \times n$ parity check matrix

$$H^{(2)} = \begin{bmatrix} 1/g^2(\alpha_0) & 1/g^2(\alpha_1) & \cdots & 1/g^2(\alpha_{n-1}) \\ \alpha_0/g^2(\alpha_0) & \alpha_1/g^2(\alpha_1) & \cdots & \alpha_{n-1}/g^2(\alpha_{n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ & & & \alpha_{n-1}^{2t-1}/g^2(\alpha_{n-1}) \end{bmatrix}.$$

Evaluate g and σ at all 2^m elements using additive FFT.

2 Transform $H^{(2)}$ to a $2mt \times n$ binary parity check matrix $H'^{(2)}$.

3 Compute the double-size syndrome: $S^{(2)} = H'^{(2)} \times (c|0)$.

4 Compute the error-locator polynomial $\sigma(x)$ from $S^{(2)}$.

5 Evaluate the error-locator polynomial $\sigma(x)$ at $(\alpha_0, \alpha_1, \dots, \alpha_{n-1})$.

Niederreiter Cryptosystem

Algorithm 3: Decryption algorithm for the Niederreiter cryptosystem.

Input : Ciphertext c, secret key $(g(x), (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}))$.

Output: Plaintext e.

1 Compute the double-size $2t \times n$ parity check matrix

$$H^{(2)} = \begin{bmatrix} 1/g^2(\alpha_0) & 1/g^2(\alpha_1) & \cdots & 1/g^2(\alpha_{n-1}) \\ \alpha_0/g^2(\alpha_0) & \alpha_1/g^2(\alpha_1) & \cdots & \alpha_{n-1}/g^2(\alpha_{n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_0^{2t-1}/g^2(\alpha_0) & \alpha_1^{2t-1}/g^2(\alpha_1) & \cdots & \alpha_{n-1}^{2t-1}/g^2(\alpha_{n-1}) \end{bmatrix}.$$

Efficient decoding algorithm.

2 Transform $H^{(2)}$ to a $2mt \times n$ binary parity check matrix $H'^{(2)}$.

3 Compute the double-size syndrome: $S^{(2)} = H'^{(2)} \times (c|0)$.

4 Compute the error-locator polynomial $\sigma(x)$ from $S^{(2)}$.

5 Evaluate the error-locator polynomial $\sigma(x)$ at $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$.

Niederreiter Cryptosystem

Efficient decoding algorithm:

- **Option 1:** Patterson algorithm.
  - Not constant time,
  - side-channel attacks can be used to decode messages.

Niederreiter Cryptosystem

Efficient decoding algorithm:

- **Option 1:** Patterson algorithm.
  - Not constant time,
  - side-channel attacks can be used to decode messages.
- **Option 2:** Berlekamp-Massey algorithm.
  - Constant time.

Niederreiter Cryptosystem

Required Modules:

- Finite field arithmetic in $\mathbb{F}(2^m)$.

Niederreiter Cryptosystem

Required Modules:

- Finite field arithmetic in $\mathbb{F}(2^m)$.
- Polynomial arithmetic in $\mathbb{F}(2^m)[x]/f$.

Niederreiter Cryptosystem

Required Modules:

- Finite field arithmetic in $\mathbb{F}(2^m)$.
- Polynomial arithmetic in $\mathbb{F}(2^m)[x]/f$.
- Merge-sort for generating a permutation.

Niederreiter Cryptosystem

Required Modules:

- Finite field arithmetic in $\mathbb{F}(2^m)$.
- Polynomial arithmetic in $\mathbb{F}(2^m)[x]/f$.
- Merge-sort for generating a permutation.
- Additive FFT for polynomial evaluation.

Niederreiter Cryptosystem

Required Modules:

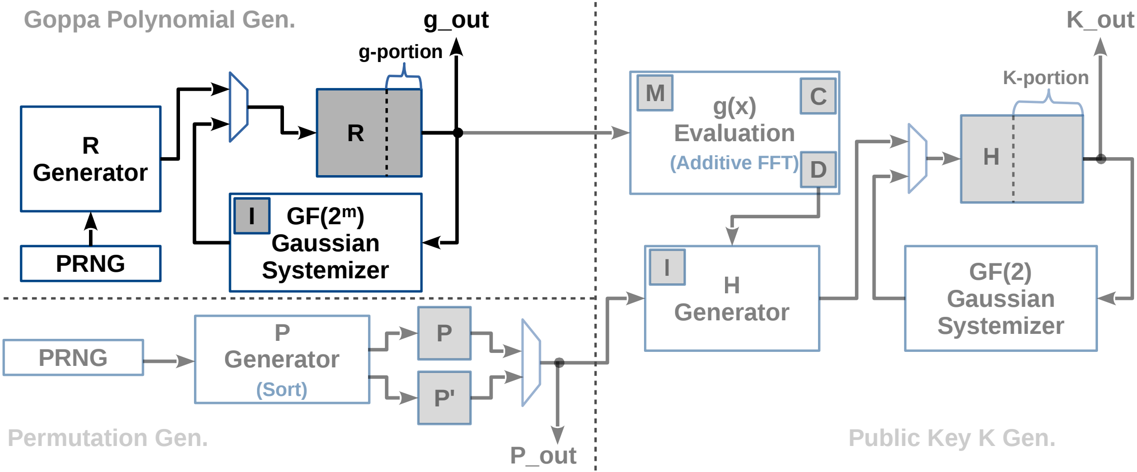
- Finite field arithmetic in $\mathbb{F}(2^m)$.
- Polynomial arithmetic in $\mathbb{F}(2^m)[x]/f$.
- Merge-sort for generating a permutation.
- Additive FFT for polynomial evaluation.
- Gaussian elimination.

Niederreiter Cryptosystem

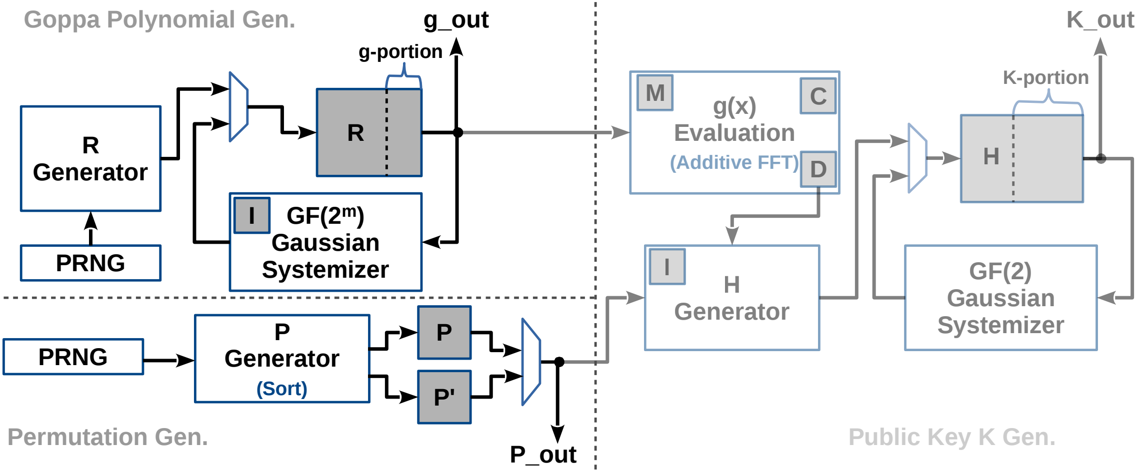
Required Modules:

- Finite field arithmetic in $\mathbb{F}(2^m)$.
- Polynomial arithmetic in $\mathbb{F}(2^m)[x]/f$.
- Merge-sort for generating a permutation.
- Additive FFT for polynomial evaluation.
- Gaussian elimination.
- Berlekamp Massey.

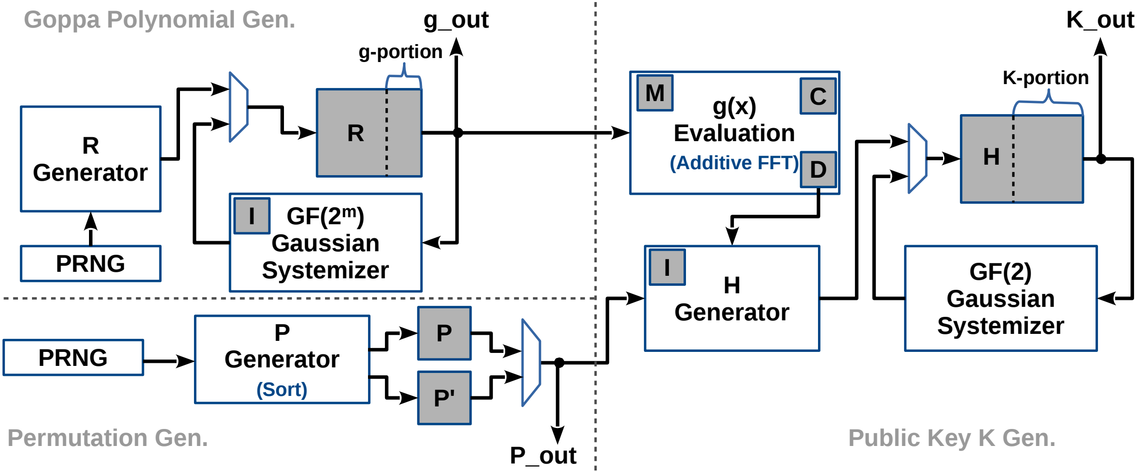
Design
Key Generation



Design
Key Generation



Design
Key Generation



Design

Algorithm 1: Key-generation algorithm for the Niederreiter cryptosystem.

Input : System parameters: m, t, and n.

Output: Private key $(g(x), (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}))$ and public key K.

1 Choose random sequence $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \in \mathbb{F}(2^m)^n$ of distinct elements.

2 Choose a random irreducible polynomial $g(x)$ of degree t.

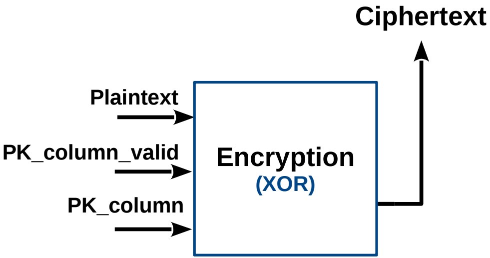
3 Compute the $t \times n$ parity check matrix

$$H = \begin{bmatrix} 1/g(\alpha_0) & 1/g(\alpha_1) & \cdots & 1/g(\alpha_{n-1}) \\ \alpha_0/g(\alpha_0) & \alpha_1/g(\alpha_1) & \cdots & \alpha_{n-1}/g(\alpha_{n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_0^{t-1}/g(\alpha_0) & \alpha_1^{t-1}/g(\alpha_1) & \cdots & \alpha_{n-1}^{t-1}/g(\alpha_{n-1}) \end{bmatrix}.$$

4 Transform H to a $mt \times n$ binary parity check matrix H'.

5 Transform H' into its systematic form $[\mathbb{I}_{mt}|K]$.

Design
Encryption

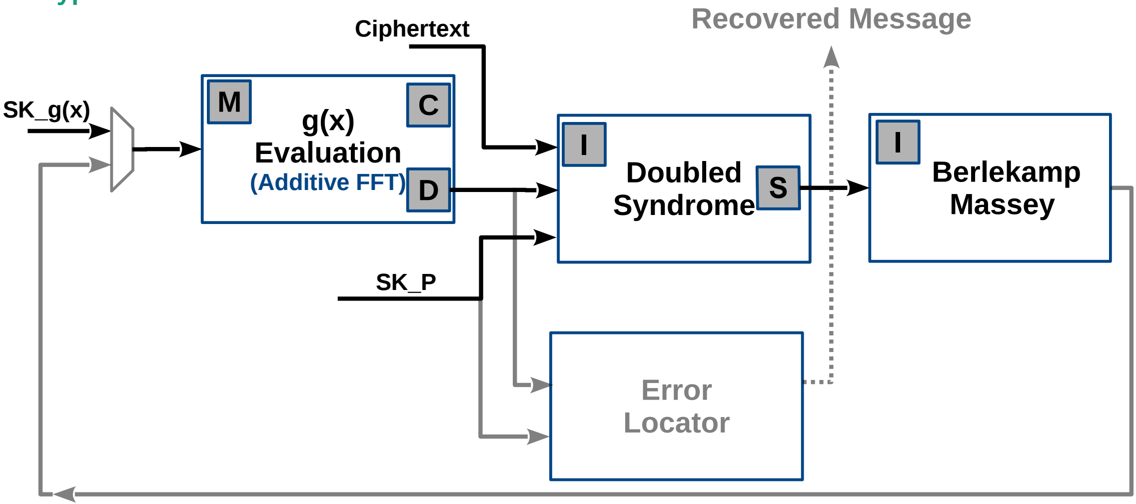


Algorithm 4: Encryption algorithm for the Niederreiter cryptosystem.

Input : Plaintext e, public key K.
Output: Ciphertext c.

1 Compute $c = [\mathbb{I}_{mt}|K] \times e$.
2 Return the ciphertext c.

Design
Decryption



Design

Algorithm 3: Decryption algorithm for the Niederreiter cryptosystem.

Input : Ciphertext c, secret key $(g(x), (\alpha_0, \alpha_1, \dots, \alpha_{n-1}))$.

Output: Plaintext e.

1 Compute the double-size $2t \times n$ parity check matrix

$$H^{(2)} = \begin{bmatrix} 1/g^2(\alpha_0) & 1/g^2(\alpha_1) & \cdots & 1/g^2(\alpha_{n-1}) \\ \alpha_0/g^2(\alpha_0) & \alpha_1/g^2(\alpha_1) & \cdots & \alpha_{n-1}/g^2(\alpha_{n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_0^{2t-1}/g^2(\alpha_0) & \alpha_1^{2t-1}/g^2(\alpha_1) & \cdots & \alpha_{n-1}^{2t-1}/g^2(\alpha_{n-1}) \end{bmatrix}.$$

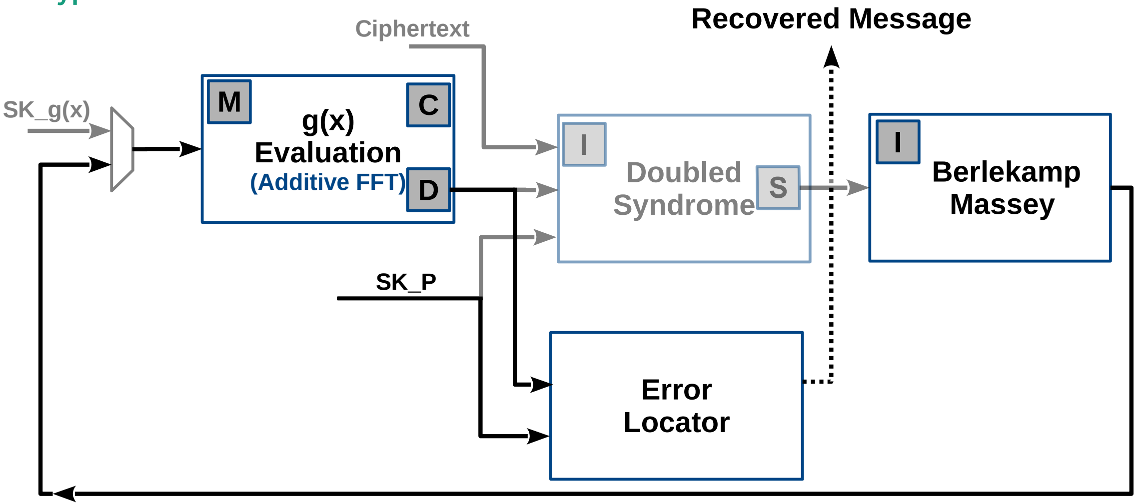
2 Transform $H^{(2)}$ to a $2mt \times n$ binary parity check matrix $H'^{(2)}$.

3 Compute the double-size syndrome: $S^{(2)} = H'^{(2)} \times (c|0)$.

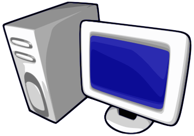
4 Compute the error-locator polynomial $\sigma(x)$ from $S^{(2)}$.

5 Evaluate the error-locator polynomial $\sigma(x)$ at $(\alpha_0, \alpha_1, \dots, \alpha_{n-1})$.

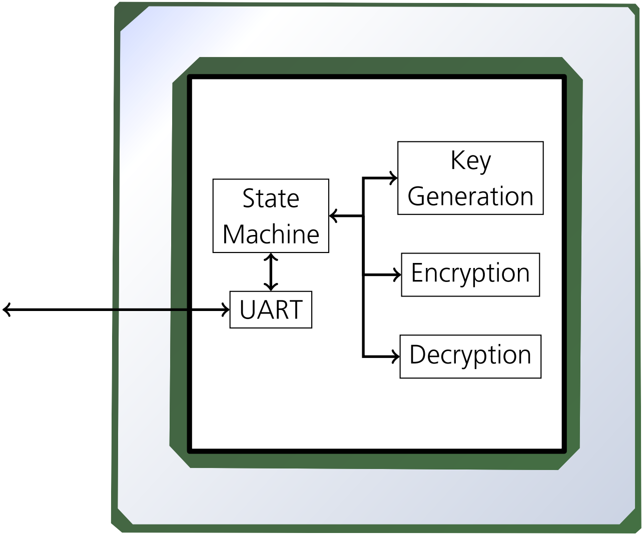
Design
Decryption



Design
Setup



verification
of results



Parameters

Code generation and module parameters:

- All system parameters (m, t, n) can be freely chosen.

Parameters

Code generation and module parameters:

- All system parameters (m, t, n) can be freely chosen.
- Performance parameters for controlling parallelism:

Parameters

Code generation and module parameters:

- All system parameters (m, t, n) can be freely chosen.
- Performance parameters for controlling parallelism:
  - Compact, low-area design for SmartCards, embedded systems, ...

Parameters

Code generation and module parameters:

- All system parameters (m, t, n) can be freely chosen.
- Performance parameters for controlling parallelism:
  - Compact, low-area design for SmartCards, embedded systems, ...
  - Large, high-performance design for server accelerator, ...

Parameters

Code generation and module parameters:

- All system parameters (m, t, n) can be freely chosen.
- Performance parameters for controlling parallelism:
  - Compact, low-area design for SmartCards, embedded systems, ...
  - Large, high-performance design for server accelerator, ...

Recommended system parameters (for 266-bit security):

- finite field 2^m: $m = 13$
- number of errors: $t = 119$
- code length: $n = 6960$

Performance

	Cycles					
Case	Key-Gen	Dec.	Logic	Mem.	Reg.	Fmax
area	11,121,214	34,492	53,447 (23%)	907 (35%)	118,243	245 MHz
bal.	3,062,936	22,768	70,478 (30%)	915 (36%)	146,648	251 MHz
time	966,400	17,055	121,806 (52%)	961 (38%)	223,232	248 MHz

Table: Performance for the entire Niederreiter cryptosystem (i.e., key generation, encryption, and decryption) including the serial IO interface when synthesized for the Stratix V (5SGXEA7N) FPGA.

Performance
Comparison

	Cycles			Logic	Freq.	Mem.	Time (ms)		
	Gen.	Dec.	Enc.		(MHz)		Gen.	Dec.	Enc.
			$m = 11, t = 50, n = 2048$, Virtex 5 LX110						
Shoufan et al.	14,670,000	210,300	81,500	14,537 (84%)	163	75	90.00	1.29	0.50
This design	1,503,927	5,864	1,498	6,660 (38%)	180	68	8.35	0.03	0.01
			$m = 13, t = 128, n = 8192$, Haswell vs. Stratix V						
Chou	1,236,054,840	343,344	289,152	—	4,000	—	309.01	0.09	0.07
This design	1,173,750	17,140	6,528	129,059 (54%)	231	1,126	5.08	0.07	0.07

Table: Comparison with related work. Logic is given in “Slices” for Xilinx Virtex FPGAs and in “ALMs” for Altera Stratix FPGAs.

Thank you for your attention!

Image Credits

Title page:

CC0 Creative Commons

`https://pixabay.com/en/boy-device-headphones-63777/`

Contact Information

Dr. Ruben Niederhagen

Cyber-Physical System Security

Fraunhofer-Institute for
Secure Information Technology

Address: Rheinstraße 75
64295 Darmstadt
Germany
Internet: http://www.sit.fraunhofer.de

Phone: +49 6151 869-135
Fax: +49 6151 869-224
E-Mail: ruben.niederhagen@sit.fraunhofer.de