

Ouroboros: a simple, secure and efficient key exchange protocol based on coding theory

Jean-Christophe Deneuille

[<jean-christophe.deneuille@xlim.fr>](mailto:jean-christophe.deneuille@xlim.fr)

April the 27th, 2017
Journées C2 – La Bresse



Joint work with:

P. Gaborit G. Zémor
University of Limoges University of Bordeaux

Outline

- 1 Preliminaries
- 2 Presentation of the Ouroboros protocol
- 3 Security
- 4 Parameters

Outline

- 1 Preliminaries
 - Encryption, Key Exchange, KEM
 - HQC
- 2 Presentation of the Ouroboros protocol
- 3 Security
- 4 Parameters

Primitives

(Public Key) Encryption

Allows to securely exchange information between two parties. Security model: IND-CPA/CCA/CCA2
examples: RSA, ElGamal, McEliece, HQC, ...

(Authenticated) Key Exchange

Allows two *authenticated* parties to agree on a common secret securely. Security model: CK, eCK; PAK
examples: Diffie-Hellman, HMQV, ...

Key Encapsulation Mechanism (KEM)

An *ephemeral* session key is encrypted with a PKE. Security model: at least IND-CPA
example: BCNS, Ding/Peikert, NewHope, ...

Primitives

(Public Key) Encryption

Allows to securely exchange information between two parties. Security model: IND-CPA/CCA/CCA2
examples: RSA, ElGamal, McEliece, HQC, ...

(Authenticated) Key Exchange

Allows two *authenticated* parties to agree on a common secret securely. Security model: CK, eCK; PAK
examples: Diffie-Hellman, HMQV, ...

Key Encapsulation Mechanism (KEM)

An *ephemeral* session key is encrypted with a PKE. Security model: at least IND-CPA
example: BCNS, Ding/Peikert, NewHope, ...

Primitives

(Public Key) Encryption

Allows to securely exchange information between two parties. Security model: IND-CPA/CCA/CCA2
examples: RSA, ElGamal, McEliece, HQC, ...

(Authenticated) Key Exchange

Allows two *authenticated* parties to agree on a common secret securely. Security model: CK, eCK; PAK
examples: Diffie-Hellman, HMQR, ...

Key Encapsulation Mechanism (KEM)

An *ephemeral* session key is encrypted with a PKE. Security model: at least IND-CPA
example: BCNS, Ding/Peikert, NewHope, ...

Outline

- 1 Preliminaries
 - Encryption, Key Exchange, KEM
 - HQC
- 2 Presentation of the Ouroboros protocol
- 3 Security
- 4 Parameters

Introduction

Encryption scheme based on **Hamming Quasi-Cyclic** codes [ABD⁺16]

- Derives from Alekhnovich's scheme [Ale03],
- Inherits its security (IND-CPA under the decisional version of the SD problem on QC codes),
- Features a thorough analysis of the Decryption Failure Rate,
- Efficient decoding for the proposed codes,
- Moderate key sizes: 7kB for 256 bits of security (against 1.25kB for MDPC [MTSB13]).

Intuition

Encryption

- Message is encoded through a code \mathcal{C}
- An error term is added to this coding using **Public Key**

Decryption

- **Secret Key** used to remove errors
- Code \mathcal{C} used for decoding back to the message

Notation

→ **Secret data** - **Public data** - **One-time Randomness**

Introduction

Encryption scheme based on **Hamming Quasi-Cyclic** codes [ABD⁺16]

- Derives from Alekhnovich's scheme [Ale03],
- Inherits its security (IND-CPA under the decisional version of the SD problem on QC codes),
- Features a thorough analysis of the Decryption Failure Rate,
- Efficient decoding for the proposed codes,
- Moderate key sizes: 7kB for 256 bits of security (against 1.25kB for MDPC [MTSB13]).

Intuition

Encryption

- Message is encoded through a code \mathcal{C}
- An error term is added to this coding using **Public Key**

Decryption

- **Secret Key** used to remove errors
- Code \mathcal{C} used for decoding back to the message

Notation

→ **Secret data** - **Public data** - **One-time Randomness**

Introduction

Encryption scheme based on **H**amming **Q**uasi-**C**yclic codes [ABD⁺16]

- Derives from Alekhnovich's scheme [Ale03],
- Inherits its security (IND-CPA under the decisional version of the SD problem on QC codes),
- Features a thorough analysis of the Decryption Failure Rate,
- Efficient decoding for the proposed codes,
- Moderate key sizes: 7kB for 256 bits of security (against 1.25kB for MDPC [MTSB13]).

Intuition

Encryption

- Message is encoded through a code \mathcal{C}
- An error term is added to this coding using **Public Key**

Decryption

- **Secret Key** used to remove errors
- Code \mathcal{C} used for decoding back to the message

Notation

→ **Secret data** - **Public data** - **One-time Randomness**

Introduction

Encryption scheme based on **Hamming Quasi-Cyclic** codes [ABD⁺16]

- Derives from Alekhovich's scheme [Ale03],
- Inherits its security (IND-CPA under the decisional version of the SD problem on QC codes),
- Features a thorough analysis of the Decryption Failure Rate,
- Efficient decoding for the proposed codes,
- Moderate key sizes: 7kB for 256 bits of security (against 1.25kB for MDPC [MTSB13]).

Intuition

Encryption

- Message is encoded through a code \mathcal{C}
- An error term is added to this coding using **Public Key**

Decryption

- **Secret Key** used to remove errors
- Code \mathcal{C} used for decoding back to the message

Notation

→ **Secret data** - **Public data** - **One-time Randomness**

Introduction

Encryption scheme based on **H**amming **Q**uasi-**C**yclic codes [ABD⁺16]

- Derives from Alekhnovich's scheme [Ale03],
- Inherits its security (IND-CPA under the decisional version of the SD problem on QC codes),
- Features a thorough analysis of the Decryption Failure Rate,
- Efficient decoding for the proposed codes,
- Moderate key sizes: 7kB for 256 bits of security (against 1.25kB for MDPC [MTSB13]).

Intuition

Encryption

- Message is encoded through a code \mathcal{C}
- An error term is added to this coding using **Public Key**

Decryption

- **Secret Key** used to remove errors
- Code \mathcal{C} used for decoding back to the message

Notation

→ **Secret data** - **Public data** - **One-time Randomness**

Introduction

Encryption scheme based on **H**amming **Q**uasi-**C**yclic codes [ABD⁺16]

- Derives from Alekhnovich's scheme [Ale03],
- Inherits its security (IND-CPA under the decisional version of the SD problem on QC codes),
- Features a thorough analysis of the Decryption Failure Rate,
- Efficient decoding for the proposed codes,
- Moderate key sizes: 7kB for 256 bits of security (against 1.25kB for MDPC [MTSB13]).

Intuition

Encryption

- Message is encoded through a code \mathcal{C}
- An error term is added to this coding using **Public Key**

Decryption

- **Secret Key** used to remove errors
- Code \mathcal{C} used for decoding back to the message

Notation

→ **Secret data** - **Public data** - **One-time Randomness**

Introduction

Encryption scheme based on **H**amming **Q**uasi-**C**yclic codes [ABD⁺16]

- Derives from Alekhnovich's scheme [Ale03],
- Inherits its security (IND-CPA under the decisional version of the SD problem on QC codes),
- Features a thorough analysis of the Decryption Failure Rate,
- Efficient decoding for the proposed codes,
- Moderate key sizes: 7kB for 256 bits of security (against 1.25kB for MDPC [MTSB13]).

Intuition

Encryption

- Message is encoded through a code \mathcal{C}
- An error term is added to this coding using **Public Key**

Decryption

- **Secret Key** used to remove errors
- Code \mathcal{C} used for decoding back to the message

Notation

→ **Secret data** - **Public data** - **One-time Randomness**

Presentation

- $\text{Setup}(1^\lambda)$: generates $n = n(\lambda)$, $k = k(\lambda)$, $\delta = \delta(\lambda)$, and $w = w(\lambda)$. Plaintext space is \mathbb{F}_2^k .
param = (n, k, δ, w) .
- $\text{KeyGen}(\text{param})$: generates $\mathbf{q}_r \xleftarrow{\$} \mathcal{V}$, the parity check matrix $\mathbf{Q} = (\mathbf{I}_n \mid \text{rot}(\mathbf{q}_r))$, and the generator matrix $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ of some code \mathcal{C} . $\mathbf{sk} = (\mathbf{x}, \mathbf{y}) \xleftarrow{\$} \mathcal{V}^2$ such that $\omega(\mathbf{x}), \omega(\mathbf{y}) \leq w$, sets $\text{pk} = (\mathbf{G}, \mathbf{Q}, \mathbf{s} = \mathbf{sk} \cdot \mathbf{Q}^\top, w)$, and returns (pk, sk) .
- $\text{Encrypt}(\text{pk} = (\mathbf{G}, \mathbf{Q}, \mathbf{s}), \mu, \theta)$: uses randomness θ to generate $\epsilon \xleftarrow{\$} \mathcal{V}$, $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2) \xleftarrow{\$} \mathcal{V}^2$ such that $\omega(\epsilon), \omega(\mathbf{r}_1), \omega(\mathbf{r}_2) \leq w$, sets $\mathbf{v}^\top = \mathbf{Q}\mathbf{r}^\top$ and $\rho = \mu\mathbf{G} + \mathbf{s} \cdot \mathbf{r}_2 + \epsilon$. It finally returns $\mathbf{c} = (\mathbf{v}, \rho)$, an encryption of μ under pk .
- $\text{Decrypt}(\text{sk} = (\mathbf{x}, \mathbf{y}), \mathbf{c} = (\mathbf{v}, \rho))$: returns $\mathcal{C}.\text{Decode}(\rho - \mathbf{v} \cdot \mathbf{y})$.

Presentation

- $\text{Setup}(1^\lambda)$: generates $n = n(\lambda)$, $k = k(\lambda)$, $\delta = \delta(\lambda)$, and $w = w(\lambda)$. Plaintext space is \mathbb{F}_2^k .
param = (n, k, δ, w) .
- $\text{KeyGen}(\text{param})$: generates $\mathbf{q}_r \xleftarrow{\$} \mathcal{V}$, the parity check matrix $\mathbf{Q} = (\mathbf{I}_n \mid \mathbf{rot}(\mathbf{q}_r))$, and the generator matrix $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ of some code \mathcal{C} . $\mathbf{sk} = (\mathbf{x}, \mathbf{y}) \xleftarrow{\$} \mathcal{V}^2$ such that $\omega(\mathbf{x}), \omega(\mathbf{y}) \leq w$, sets $\mathbf{pk} = (\mathbf{G}, \mathbf{Q}, \mathbf{s} = \mathbf{sk} \cdot \mathbf{Q}^\top, w)$, and returns $(\mathbf{pk}, \mathbf{sk})$.
- $\text{Encrypt}(\mathbf{pk} = (\mathbf{G}, \mathbf{Q}, \mathbf{s}), \mu, \theta)$: uses randomness θ to generate $\epsilon \xleftarrow{\$} \mathcal{V}$, $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2) \xleftarrow{\$} \mathcal{V}^2$ such that $\omega(\epsilon), \omega(\mathbf{r}_1), \omega(\mathbf{r}_2) \leq w$, sets $\mathbf{v}^\top = \mathbf{Q}\mathbf{r}^\top$ and $\rho = \mu\mathbf{G} + \mathbf{s} \cdot \mathbf{r}_2 + \epsilon$. It finally returns $\mathbf{c} = (\mathbf{v}, \rho)$, an encryption of μ under \mathbf{pk} .
- $\text{Decrypt}(\mathbf{sk} = (\mathbf{x}, \mathbf{y}), \mathbf{c} = (\mathbf{v}, \rho))$: returns $\mathcal{C}.\text{Decode}(\rho - \mathbf{v} \cdot \mathbf{y})$.

Presentation

- $\text{Setup}(1^\lambda)$: generates $n = n(\lambda)$, $k = k(\lambda)$, $\delta = \delta(\lambda)$, and $w = w(\lambda)$. Plaintext space is \mathbb{F}_2^k .
param = (n, k, δ, w) .
- $\text{KeyGen}(\text{param})$: generates $\mathbf{q}_r \xleftarrow{\$} \mathcal{V}$, the parity check matrix $\mathbf{Q} = (\mathbf{I}_n \mid \mathbf{rot}(\mathbf{q}_r))$, and the generator matrix $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ of some code \mathcal{C} . $\mathbf{sk} = (\mathbf{x}, \mathbf{y}) \xleftarrow{\$} \mathcal{V}^2$ such that $\omega(\mathbf{x}), \omega(\mathbf{y}) \leq w$, sets $\mathbf{pk} = (\mathbf{G}, \mathbf{Q}, \mathbf{s} = \mathbf{sk} \cdot \mathbf{Q}^\top, w)$, and returns $(\mathbf{pk}, \mathbf{sk})$.
- $\text{Encrypt}(\mathbf{pk} = (\mathbf{G}, \mathbf{Q}, \mathbf{s}), \boldsymbol{\mu}, \theta)$: uses randomness θ to generate $\boldsymbol{\epsilon} \xleftarrow{\$} \mathcal{V}$, $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2) \xleftarrow{\$} \mathcal{V}^2$ such that $\omega(\boldsymbol{\epsilon}), \omega(\mathbf{r}_1), \omega(\mathbf{r}_2) \leq w$, sets $\mathbf{v}^\top = \mathbf{Q}\mathbf{r}^\top$ and $\boldsymbol{\rho} = \boldsymbol{\mu}\mathbf{G} + \mathbf{s} \cdot \mathbf{r}_2 + \boldsymbol{\epsilon}$. It finally returns $\mathbf{c} = (\mathbf{v}, \boldsymbol{\rho})$, an encryption of $\boldsymbol{\mu}$ under \mathbf{pk} .
- $\text{Decrypt}(\mathbf{sk} = (\mathbf{x}, \mathbf{y}), \mathbf{c} = (\mathbf{v}, \boldsymbol{\rho}))$: returns $\mathcal{C}.\text{Decode}(\boldsymbol{\rho} - \mathbf{v} \cdot \mathbf{y})$.

Presentation

- $\text{Setup}(1^\lambda)$: generates $n = n(\lambda)$, $k = k(\lambda)$, $\delta = \delta(\lambda)$, and $w = w(\lambda)$. Plaintext space is \mathbb{F}_2^k .
param = (n, k, δ, w) .
- $\text{KeyGen}(\text{param})$: generates $\mathbf{q}_r \xleftarrow{\$} \mathcal{V}$, the parity check matrix $\mathbf{Q} = (\mathbf{I}_n \mid \mathbf{rot}(\mathbf{q}_r))$, and the generator matrix $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ of some code \mathcal{C} . $\mathbf{sk} = (\mathbf{x}, \mathbf{y}) \xleftarrow{\$} \mathcal{V}^2$ such that $\omega(\mathbf{x}), \omega(\mathbf{y}) \leq w$, sets $\mathbf{pk} = (\mathbf{G}, \mathbf{Q}, \mathbf{s} = \mathbf{sk} \cdot \mathbf{Q}^\top, w)$, and returns $(\mathbf{pk}, \mathbf{sk})$.
- $\text{Encrypt}(\mathbf{pk} = (\mathbf{G}, \mathbf{Q}, \mathbf{s}), \boldsymbol{\mu}, \theta)$: uses randomness θ to generate $\boldsymbol{\epsilon} \xleftarrow{\$} \mathcal{V}$, $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2) \xleftarrow{\$} \mathcal{V}^2$ such that $\omega(\boldsymbol{\epsilon}), \omega(\mathbf{r}_1), \omega(\mathbf{r}_2) \leq w$, sets $\mathbf{v}^\top = \mathbf{Q}\mathbf{r}^\top$ and $\boldsymbol{\rho} = \boldsymbol{\mu}\mathbf{G} + \mathbf{s} \cdot \mathbf{r}_2 + \boldsymbol{\epsilon}$. It finally returns $\mathbf{c} = (\mathbf{v}, \boldsymbol{\rho})$, an encryption of $\boldsymbol{\mu}$ under \mathbf{pk} .
- $\text{Decrypt}(\mathbf{sk} = (\mathbf{x}, \mathbf{y}), \mathbf{c} = (\mathbf{v}, \boldsymbol{\rho}))$: returns $\mathcal{C}.\text{Decode}(\boldsymbol{\rho} - \mathbf{v} \cdot \mathbf{y})$.

Correctness

Correctness Property

$$\text{Decrypt}(\text{sk}, \text{Encrypt}(\text{pk}, \mu, \theta)) = \mu$$

\mathcal{C} . Decode correctly decodes $\rho - \mathbf{v} \cdot \mathbf{y}$ whenever

the error term is **not too big**

$$\omega(\mathbf{s} \cdot \mathbf{r}_2 - \mathbf{v} \cdot \mathbf{y} + \epsilon) \leq \delta$$

$$\omega((\mathbf{x} + \mathbf{q}_r \cdot \mathbf{y}) \cdot \mathbf{r}_2 - (\mathbf{r}_1 + \mathbf{q}_r \cdot \mathbf{r}_2) \cdot \mathbf{y} + \epsilon) \leq \delta$$

$$\omega(\mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y} + \epsilon) \leq \delta$$

Error distribution analysis \rightarrow Decryption failure probability better understood

Correctness

Correctness Property

$$\text{Decrypt}(\text{sk}, \text{Encrypt}(\text{pk}, \mu, \theta)) = \mu$$

\mathcal{C} .Decode correctly decodes $\rho - \mathbf{v} \cdot \mathbf{y}$ whenever

the error term is **not too big**

$$\omega(\mathbf{s} \cdot \mathbf{r}_2 - \mathbf{v} \cdot \mathbf{y} + \epsilon) \leq \delta$$

$$\omega((\mathbf{x} + \mathbf{q}_r \cdot \mathbf{y}) \cdot \mathbf{r}_2 - (\mathbf{r}_1 + \mathbf{q}_r \cdot \mathbf{r}_2) \cdot \mathbf{y} + \epsilon) \leq \delta$$

$$\omega(\mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y} + \epsilon) \leq \delta$$

Error distribution analysis → Decryption failure probability better understood

Correctness

Correctness Property

$$\text{Decrypt}(\text{sk}, \text{Encrypt}(\text{pk}, \mu, \theta)) = \mu$$

\mathcal{C} .Decode correctly decodes $\rho - \mathbf{v} \cdot \mathbf{y}$ whenever

the error term is **not too big**

$$\omega(\mathbf{s} \cdot \mathbf{r}_2 - \mathbf{v} \cdot \mathbf{y} + \epsilon) \leq \delta$$

$$\omega((\mathbf{x} + \mathbf{q}_r \cdot \mathbf{y}) \cdot \mathbf{r}_2 - (\mathbf{r}_1 + \mathbf{q}_r \cdot \mathbf{r}_2) \cdot \mathbf{y} + \epsilon) \leq \delta$$

$$\omega(\mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y} + \epsilon) \leq \delta$$

Error distribution analysis → **Decryption failure probability better understood**

Outline

- 1 Preliminaries
- 2 Presentation of the Ouroboros protocol
 - Cyclic Error Decoding
 - BitFlip algorithm
 - Description of the protocol
- 3 Security
- 4 Parameters

A particular decoding

- HQC requires $\mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y} + \epsilon$ to be “small” to correctly decode
- Ouroboros further exploits the shape of the error

Cyclic Error Decoding (CED) Problem

- Let $\mathbf{x}, \mathbf{y}, \mathbf{r}_1, \mathbf{r}_2 \leftarrow \mathcal{S}_w^n(\mathbb{F}_2)$ with $w = \mathcal{O}(\sqrt{n})$, and $\mathbf{e} \leftarrow \mathcal{S}_{cw}^n(\mathbb{F}_2)$ a random error vector.
- Given $(\mathbf{x}, \mathbf{y}) \in (\mathcal{S}_w^n(\mathbb{F}_2))^2$ and $\mathbf{e}_c \leftarrow \mathbf{x}\mathbf{r}_2 - \mathbf{y}\mathbf{r}_1 + \mathbf{e}$ such that $\omega(\mathbf{r}_1) = \omega(\mathbf{r}_2) = w$, find $(\mathbf{r}_1, \mathbf{r}_2)$.

- This is essentially a *noisy* SD problem



A particular decoding

- HQC requires $\mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y} + \epsilon$ to be “small” to correctly decode
- Ouroboros further exploits the shape of the error

Cyclic Error Decoding (CED) Problem

- Let $\mathbf{x}, \mathbf{y}, \mathbf{r}_1, \mathbf{r}_2 \stackrel{\$}{\leftarrow} \mathcal{S}_w^n(\mathbb{F}_2)$ with $w = \mathcal{O}(\sqrt{n})$, and $\mathbf{e} \stackrel{\$}{\leftarrow} \mathcal{S}_{cw}^n(\mathbb{F}_2)$ a random error vector.
- Given $(\mathbf{x}, \mathbf{y}) \in (\mathcal{S}_w^n(\mathbb{F}_2))^2$ and $\mathbf{e}_c \leftarrow \mathbf{x}\mathbf{r}_2 - \mathbf{y}\mathbf{r}_1 + \mathbf{e}$ such that $\omega(\mathbf{r}_1) = \omega(\mathbf{r}_2) = w$, find $(\mathbf{r}_1, \mathbf{r}_2)$.

- This is essentially a *noisy* SD problem



A particular decoding

- HQC requires $\mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y} + \epsilon$ to be “small” to correctly decode
- Ouroboros further exploits the shape of the error

Cyclic Error Decoding (CED) Problem

- Let $\mathbf{x}, \mathbf{y}, \mathbf{r}_1, \mathbf{r}_2 \stackrel{\$}{\leftarrow} \mathcal{S}_w^n(\mathbb{F}_2)$ with $w = \mathcal{O}(\sqrt{n})$, and $\mathbf{e} \stackrel{\$}{\leftarrow} \mathcal{S}_{cw}^n(\mathbb{F}_2)$ a random error vector.
- Given $(\mathbf{x}, \mathbf{y}) \in (\mathcal{S}_w^n(\mathbb{F}_2))^2$ and $\mathbf{e}_c \leftarrow \mathbf{x}\mathbf{r}_2 - \mathbf{y}\mathbf{r}_1 + \mathbf{e}$ such that $\omega(\mathbf{r}_1) = \omega(\mathbf{r}_2) = w$, find $(\mathbf{r}_1, \mathbf{r}_2)$.
- This is essentially a *noisy* SD problem



A particular decoding

- HQC requires $\mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y} + \epsilon$ to be “small” to correctly decode
- Ouroboros further exploits the shape of the error

Cyclic Error Decoding (CED) Problem

- Let $\mathbf{x}, \mathbf{y}, \mathbf{r}_1, \mathbf{r}_2 \stackrel{\$}{\leftarrow} \mathcal{S}_w^n(\mathbb{F}_2)$ with $w = \mathcal{O}(\sqrt{n})$, and $\mathbf{e} \stackrel{\$}{\leftarrow} \mathcal{S}_{cw}^n(\mathbb{F}_2)$ a random error vector.
- Given $(\mathbf{x}, \mathbf{y}) \in (\mathcal{S}_w^n(\mathbb{F}_2))^2$ and $\mathbf{e}_c \leftarrow \mathbf{x}\mathbf{r}_2 - \mathbf{y}\mathbf{r}_1 + \mathbf{e}$ such that $\omega(\mathbf{r}_1) = \omega(\mathbf{r}_2) = w$, find $(\mathbf{r}_1, \mathbf{r}_2)$.
- This is essentially a *noisy* SD problem

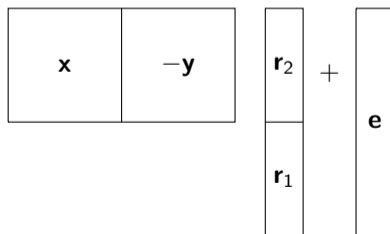


A particular decoding

- HQC requires $\mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y} + \epsilon$ to be “small” to correctly decode
- Ouroboros further exploits the shape of the error

Cyclic Error Decoding (CED) Problem

- Let $\mathbf{x}, \mathbf{y}, \mathbf{r}_1, \mathbf{r}_2 \stackrel{\$}{\leftarrow} \mathcal{S}_w^n(\mathbb{F}_2)$ with $w = \mathcal{O}(\sqrt{n})$, and $\mathbf{e} \stackrel{\$}{\leftarrow} \mathcal{S}_{cw}^n(\mathbb{F}_2)$ a random error vector.
- Given $(\mathbf{x}, \mathbf{y}) \in (\mathcal{S}_w^n(\mathbb{F}_2))^2$ and $\mathbf{e}_c \leftarrow \mathbf{x}\mathbf{r}_2 - \mathbf{y}\mathbf{r}_1 + \mathbf{e}$ such that $\omega(\mathbf{r}_1) = \omega(\mathbf{r}_2) = w$, find $(\mathbf{r}_1, \mathbf{r}_2)$.
- This is essentially a *noisy* SD problem



Outline

- 1 Preliminaries
- 2 Presentation of the Ouroboros protocol**
 - Cyclic Error Decoding
 - BitFlip algorithm**
 - Description of the protocol
- 3 Security
- 4 Parameters

Hard Decision Decoding: BitFlip

- Introduced by Gallager in 1962
- Iterative decoding for **Low Density Parity Check** codes
- Decoding capacity increase linearly with the code length

Intuition

- 1 Compute the number of unsatisfied parity-check equations for each bit of the message
- 2 If this number is greater than some *threshold*, flip the bit and go to 1.
- 3 Stop when the syndrome is null (or after a certain number of iterations).

- Easy to understand
- Easy to implement
- Pretty efficient
- The threshold value is crucial [CS16]

Hard Decision Decoding: BitFlip

- Introduced by Gallager in 1962
- Iterative decoding for **Low Density Parity Check** codes
- Decoding capacity increase linearly with the code length

Intuition

- 1 Compute the number of unsatisfied parity-check equations for each bit of the message
- 2 If this number is greater than some *threshold*, flip the bit and go to 1.
- 3 Stop when the syndrome is null (or after a certain number of iterations).

- Easy to understand
- Easy to implement
- Pretty efficient
- The threshold value is crucial [CS16]

Hard Decision Decoding: BitFlip

- Introduced by Gallager in 1962
- Iterative decoding for **L**ow **D**ensity **P**arity **C**heck codes
- Decoding capacity increase linearly with the code length

Intuition

- 1 Compute the number of unsatisfied parity-check equations for each bit of the message
- 2 If this number is greater than some *threshold*, flip the bit and go to 1.
- 3 Stop when the syndrome is null (or after a certain number of iterations).

- Easy to understand
- Easy to implement
- Pretty efficient
- The threshold value is crucial [CS16]

Hard Decision Decoding: BitFlip

- Introduced by Gallager in 1962
- Iterative decoding for **L**ow **D**ensity **P**arity **C**heck codes
- Decoding capacity increase linearly with the code length

Intuition

- 1 Compute the number of unsatisfied parity-check equations for each bit of the message
 - 2 If this number is greater than some *threshold*, flip the bit and go to 1.
 - 3 Stop when the syndrome is null (or after a certain number of iterations).
- Easy to understand
 - Easy to implement
 - Pretty efficient
 - The threshold value is crucial [CS16]

Hard Decision Decoding: BitFlip

- Introduced by Gallager in 1962
- Iterative decoding for **L**ow **D**ensity **P**arity **C**heck codes
- Decoding capacity increase linearly with the code length

Intuition

- 1 Compute the number of unsatisfied parity-check equations for each bit of the message
- 2 If this number is greater than some *threshold*, flip the bit and go to 1.
- 3 Stop when the syndrome is null (or after a certain number of iterations).

- Easy to understand
- Easy to implement
- Pretty efficient
- The threshold value is crucial [CS16]

Hard Decision Decoding: BitFlip

- Introduced by Gallager in 1962
- Iterative decoding for **L**ow **D**ensity **P**arity **C**heck codes
- Decoding capacity increase linearly with the code length

Intuition

- 1 Compute the number of unsatisfied parity-check equations for each bit of the message
- 2 If this number is greater than some *threshold*, flip the bit and go to 1.
- 3 Stop when the syndrome is null (or after a certain number of iterations).

- Easy to understand
- Easy to implement
- Pretty efficient
- The threshold value is crucial [CS16]

Hard Decision Decoding: BitFlip

- Introduced by Gallager in 1962
- Iterative decoding for **L**ow **D**ensity **P**arity **C**heck codes
- Decoding capacity increase linearly with the code length

Intuition

- 1 Compute the number of unsatisfied parity-check equations for each bit of the message
- 2 If this number is greater than some *threshold*, flip the bit and go to 1.
- 3 Stop when the syndrome is null (or after a certain number of iterations).

- Easy to understand
- Easy to implement
- Pretty efficient
- The threshold value is crucial [CS16]

Hard Decision Decoding: BitFlip

- Introduced by Gallager in 1962
- Iterative decoding for **L**ow **D**ensity **P**arity **C**heck codes
- Decoding capacity increase linearly with the code length

Intuition

- 1 Compute the number of unsatisfied parity-check equations for each bit of the message
- 2 If this number is greater than some *threshold*, flip the bit and go to 1.
- 3 Stop when the syndrome is null (or after a certain number of iterations).

- Easy to understand
- Easy to implement
- Pretty efficient
- The threshold value is crucial [CS16]

Hard Decision Decoding: BitFlip

- Introduced by Gallager in 1962
- Iterative decoding for **L**ow **D**ensity **P**arity **C**heck codes
- Decoding capacity increase linearly with the code length

Intuition

- 1 Compute the number of unsatisfied parity-check equations for each bit of the message
- 2 If this number is greater than some *threshold*, flip the bit and go to 1.
- 3 Stop when the syndrome is null (or after a certain number of iterations).

- Easy to understand
- Easy to implement
- Pretty efficient
- The threshold value is crucial [CS16]

Hard Decision Decoding: BitFlip

- Introduced by Gallager in 1962
- Iterative decoding for **L**ow **D**ensity **P**arity **C**heck codes
- Decoding capacity increase linearly with the code length

Intuition

- 1 Compute the number of unsatisfied parity-check equations for each bit of the message
- 2 If this number is greater than some *threshold*, flip the bit and go to 1.
- 3 Stop when the syndrome is null (or after a certain number of iterations).

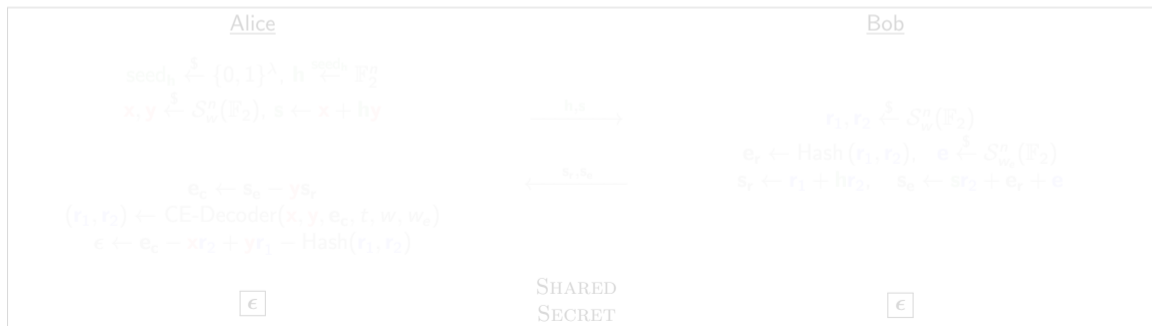
- Easy to understand
- Easy to implement
- Pretty efficient
- **The threshold value is crucial [CS16]**

Outline

- 1 Preliminaries
- 2 Presentation of the Ouroboros protocol
 - Cyclic Error Decoding
 - BitFlip algorithm
 - Description of the protocol
- 3 Security
- 4 Parameters

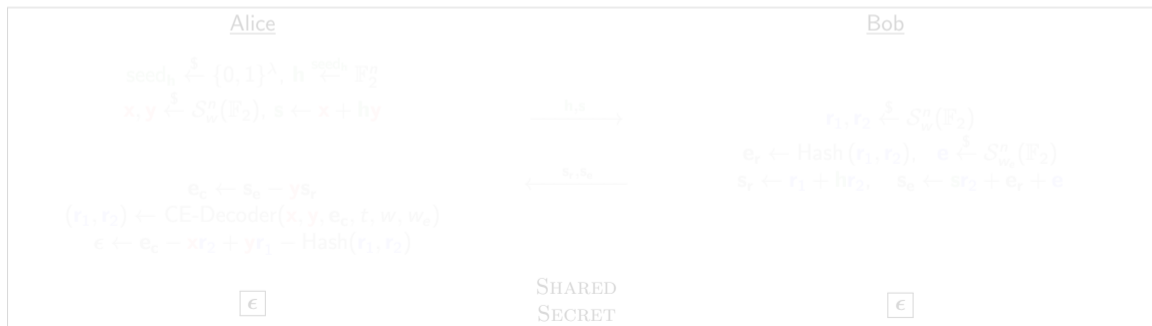
Ouroboros

- Requires a hash function $\text{Hash} : \{0, 1\}^* \rightarrow \mathcal{S}_{cw}^n(\mathbb{F}_2)$ [Sen05]
- ϵ of HQC plays the role of the exchanged secret in Ouroboros
- CE-Decoder is a modified BitFlip algorithm to solve the CED problem



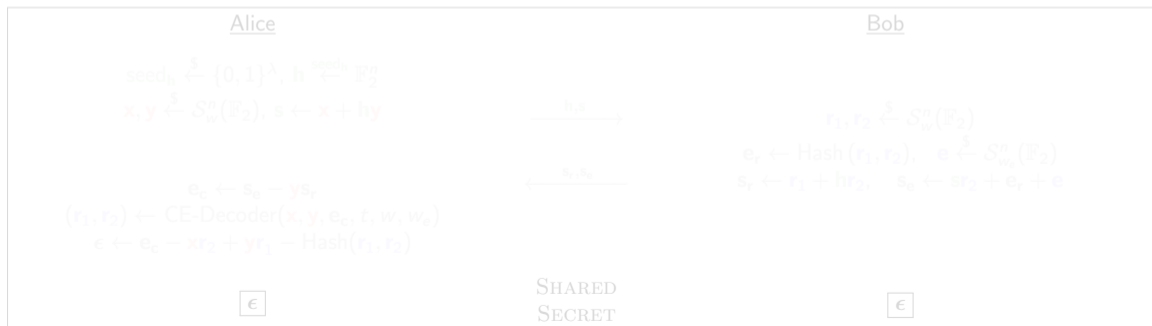
Ouroboros

- Requires a hash function $\text{Hash} : \{0, 1\}^* \rightarrow \mathcal{S}_{cw}^n(\mathbb{F}_2)$ [Sen05]
- ϵ of HQC plays the role of the exchanged secret in Ouroboros
- CE-Decoder is a modified BitFlip algorithm to solve the CED problem



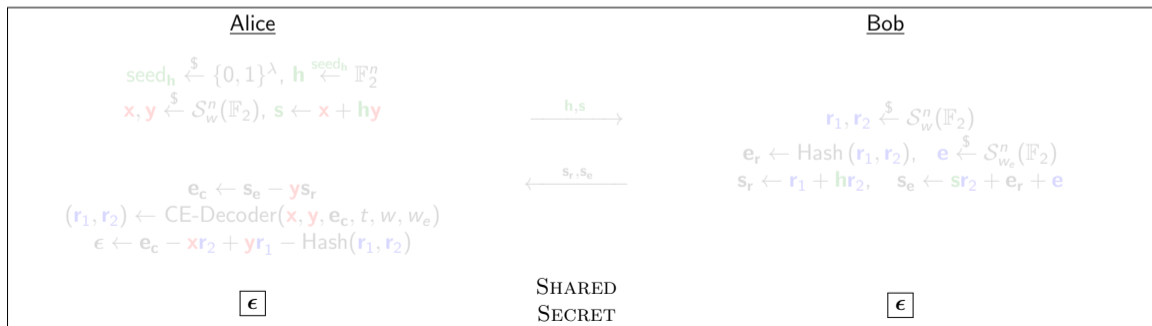
Ouroboros

- Requires a hash function $\text{Hash} : \{0, 1\}^* \rightarrow \mathcal{S}_{cw}^n(\mathbb{F}_2)$ [Sen05]
- ϵ of HQC plays the role of the exchanged secret in Ouroboros
- CE-Decoder is a modified BitFlip algorithm to solve the CED problem



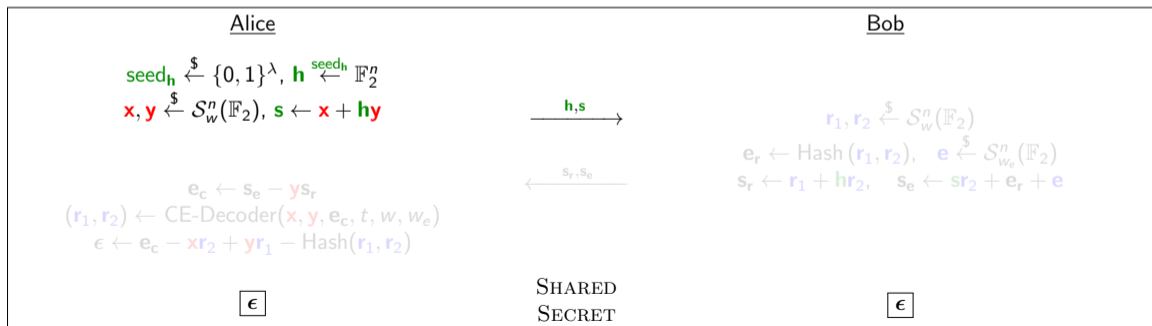
Ouroboros

- Requires a hash function $\text{Hash} : \{0, 1\}^* \rightarrow \mathcal{S}_{cw}^n(\mathbb{F}_2)$ [Sen05]
- ϵ of HQC plays the role of the exchanged secret in Ouroboros
- CE-Decoder is a modified BitFlip algorithm to solve the CED problem



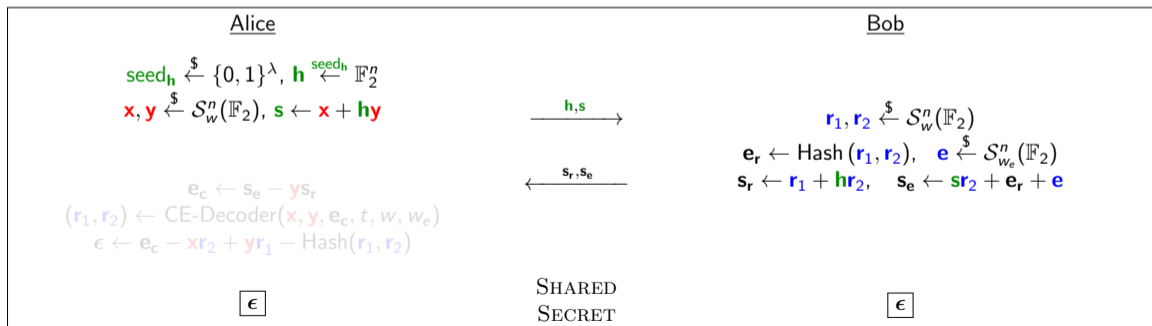
Ouroboros

- Requires a hash function $\text{Hash} : \{0, 1\}^* \rightarrow \mathcal{S}_{cw}^n(\mathbb{F}_2)$ [Sen05]
- ϵ of HQC plays the role of the exchanged secret in Ouroboros
- CE-Decoder is a modified BitFlip algorithm to solve the CED problem



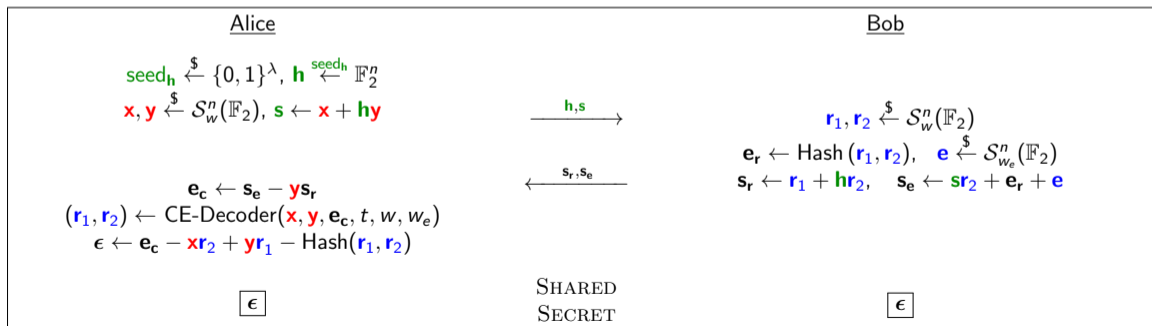
Ouroboros

- Requires a hash function $\text{Hash} : \{0, 1\}^* \rightarrow \mathcal{S}_{cw}^n(\mathbb{F}_2)$ [Sen05]
- ϵ of HQC plays the role of the exchanged secret in Ouroboros
- CE-Decoder is a modified BitFlip algorithm to solve the CED problem



Ouroboros

- Requires a hash function $\text{Hash} : \{0, 1\}^* \rightarrow \mathcal{S}_{cw}^n(\mathbb{F}_2)$ [Sen05]
- ϵ of HQC plays the role of the exchanged secret in Ouroboros
- CE-Decoder is a modified BitFlip algorithm to solve the CED problem



Outline

- 1 Preliminaries
- 2 Presentation of the Ouroboros protocol
- 3 **Security**
 - Security Model and Hybrid Argument
 - Sketch of proof
- 4 Parameters

Security Model and Hybrid Argument

- Key exchange as an encryption scheme
 - Same as Ding *et al.* [Din12, DXL12], Peikert's [Pei14], BCNS [BCNS15] and NEWHOPE [ADPS16]
 - Usual game:

$\text{Exp}_{\mathcal{E}, \mathcal{A}}^{\text{ind}-b}(\lambda)$

1. $\text{param} \leftarrow \text{Setup}(1^\lambda)$
2. $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{param})$
3. $(\mu_0, \mu_1) \leftarrow \mathcal{A}(\text{FIND} : \text{pk})$
4. $\mathbf{c}^* \leftarrow \text{Encrypt}(\text{pk}, \mu_b, \theta)$
5. $b' \leftarrow \mathcal{A}(\text{GUESS} : \mathbf{c}^*)$
6. RETURN b'

- Hybrid argument:
 - 1 Construct a sequence of games transitioning from $\text{Enc}(\mu_0)$ to $\text{Enc}(\mu_1)$
 - 2 Prove they are indistinguishable one from another

Security Model and Hybrid Argument

- Key exchange as an encryption scheme
- Same as Ding *et al.* [Din12, DXL12], Peikert's [Pei14], BCNS [BCNS15] and NEWHOPE [ADPS16]
- Usual game:

$\text{Exp}_{\mathcal{E}, \mathcal{A}}^{\text{ind}-b}(\lambda)$

1. $\text{param} \leftarrow \text{Setup}(1^\lambda)$
2. $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{param})$
3. $(\mu_0, \mu_1) \leftarrow \mathcal{A}(\text{FIND} : \text{pk})$
4. $\mathbf{c}^* \leftarrow \text{Encrypt}(\text{pk}, \mu_b, \theta)$
5. $b' \leftarrow \mathcal{A}(\text{GUESS} : \mathbf{c}^*)$
6. RETURN b'

- Hybrid argument:
 - 1 Construct a sequence of games transitioning from $\text{Enc}(\mu_0)$ to $\text{Enc}(\mu_1)$
 - 2 Prove they are indistinguishable one from another

Security Model and Hybrid Argument

- Key exchange as an encryption scheme
- Same as Ding *et al.* [Din12, DXL12], Peikert's [Pei14], BCNS [BCNS15] and NEWHOPE [ADPS16]
- Usual game:

$\text{Exp}_{\mathcal{E}, \mathcal{A}}^{\text{ind}-b}(\lambda)$

1. $\text{param} \leftarrow \text{Setup}(1^\lambda)$
2. $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{param})$
3. $(\mu_0, \mu_1) \leftarrow \mathcal{A}(\text{FIND} : \text{pk})$
4. $\mathbf{c}^* \leftarrow \text{Encrypt}(\text{pk}, \mu_b, \theta)$
5. $b' \leftarrow \mathcal{A}(\text{GUESS} : \mathbf{c}^*)$
6. RETURN b'

- Hybrid argument:
 - 1 Construct a sequence of games transitioning from $\text{Enc}(\mu_0)$ to $\text{Enc}(\mu_1)$
 - 2 Prove they are indistinguishable one from another

Security Model and Hybrid Argument

- Key exchange as an encryption scheme
- Same as Ding *et al.* [Din12, DXL12], Peikert's [Pei14], BCNS [BCNS15] and NEWHOPE [ADPS16]
- Usual game:

$\text{Exp}_{\mathcal{E}, \mathcal{A}}^{\text{ind}-b}(\lambda)$

1. $\text{param} \leftarrow \text{Setup}(1^\lambda)$
2. $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{param})$
3. $(\mu_0, \mu_1) \leftarrow \mathcal{A}(\text{FIND} : \text{pk})$
4. $\mathbf{c}^* \leftarrow \text{Encrypt}(\text{pk}, \mu_b, \theta)$
5. $b' \leftarrow \mathcal{A}(\text{GUESS} : \mathbf{c}^*)$
6. RETURN b'

- Hybrid argument:
 - 1 Construct a sequence of games transitioning from $\text{Enc}(\mu_0)$ to $\text{Enc}(\mu_1)$
 - 2 Prove they are indistinguishable one from another

Security

Definition (SD Distribution)

For positive integers, n , k , and w , the $SD(n, k, w)$ Distribution chooses $\mathbf{H} \xleftarrow{\$} \mathbb{F}^{(n-k) \times n}$ and $\mathbf{x} \xleftarrow{\$} \mathbb{F}^n$ such that $\omega(\mathbf{x}) = w$, and outputs $(\mathbf{H}, \mathbf{H}\mathbf{x}^\top)$.

Definition (Decisional s -QCSD Problem)

For positive integers n , k , w , s , a random parity check matrix \mathbf{H} of a QC code \mathcal{C} and $\mathbf{y} \xleftarrow{\$} \mathbb{F}^n$, the *Decisional s -Quasi-Cyclic SD Problem* s -DQCSD(n, k, w) asks to decide with non-negligible advantage whether $(\mathbf{H}, \mathbf{y}^\top)$ came from the s -QCSD(n, k, w) distribution or the uniform distribution over $\mathbb{F}^{(n-k) \times n} \times \mathbb{F}^{n-k}$.

Theorem

The scheme presented above is IND-CPA under the 2-DQCSD and 3-DQCSD assumptions.

Security

Definition (SD Distribution)

For positive integers, n , k , and w , the $SD(n, k, w)$ Distribution chooses $\mathbf{H} \xleftarrow{\$} \mathbb{F}^{(n-k) \times n}$ and $\mathbf{x} \xleftarrow{\$} \mathbb{F}^n$ such that $\omega(\mathbf{x}) = w$, and outputs $(\mathbf{H}, \mathbf{H}\mathbf{x}^\top)$.

Definition (Decisional s -QCSD Problem)

For positive integers n , k , w , s , a random parity check matrix \mathbf{H} of a QC code \mathcal{C} and $\mathbf{y} \xleftarrow{\$} \mathbb{F}^n$, the *Decisional s -Quasi-Cyclic SD Problem* s -DQCSD(n, k, w) asks to decide with non-negligible advantage whether $(\mathbf{H}, \mathbf{y}^\top)$ came from the s -QCSD(n, k, w) distribution or the uniform distribution over $\mathbb{F}^{(n-k) \times n} \times \mathbb{F}^{n-k}$.

Theorem

The scheme presented above is IND-CPA under the 2-DQCSD and 3-DQCSD assumptions.

Security

Definition (SD Distribution)

For positive integers, n , k , and w , the $SD(n, k, w)$ Distribution chooses $\mathbf{H} \xleftarrow{\$} \mathbb{F}^{(n-k) \times n}$ and $\mathbf{x} \xleftarrow{\$} \mathbb{F}^n$ such that $\omega(\mathbf{x}) = w$, and outputs $(\mathbf{H}, \mathbf{H}\mathbf{x}^\top)$.

Definition (Decisional s -QCSD Problem)

For positive integers n , k , w , s , a random parity check matrix \mathbf{H} of a QC code \mathcal{C} and $\mathbf{y} \xleftarrow{\$} \mathbb{F}^n$, the *Decisional s -Quasi-Cyclic SD Problem s -DQCSD(n, k, w)* asks to decide with non-negligible advantage whether $(\mathbf{H}, \mathbf{y}^\top)$ came from the s -QCSD(n, k, w) distribution or the uniform distribution over $\mathbb{F}^{(n-k) \times n} \times \mathbb{F}^{n-k}$.

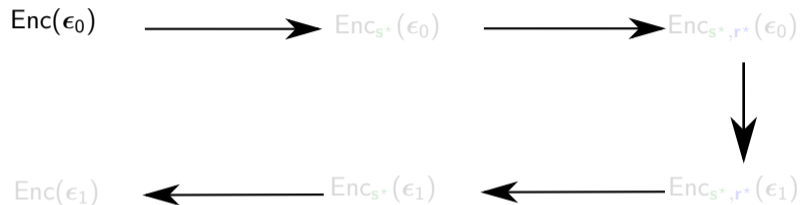
Theorem

The scheme presented above is IND-CPA under the 2-DQCSD and 3-DQCSD assumptions.

Outline

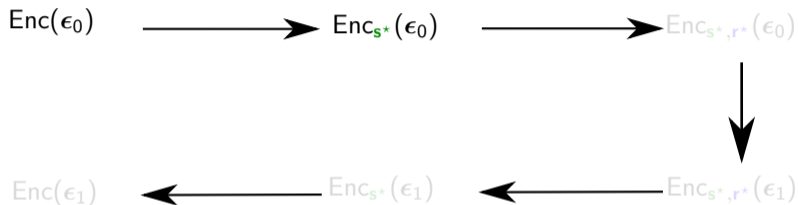
- 1 Preliminaries
- 2 Presentation of the Ouroboros protocol
- 3 Security**
 - Security Model and Hybrid Argument
 - Sketch of proof
- 4 Parameters

Sequence of games from $\text{Enc}(\epsilon_0)$ to $\text{Enc}(\epsilon_1)$



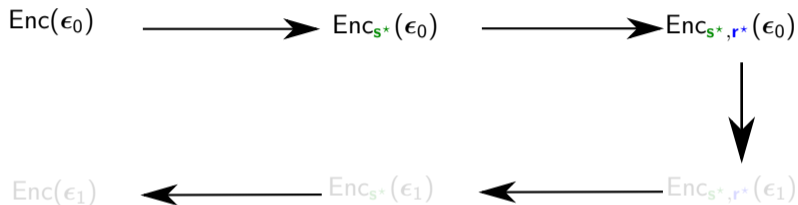
$$\text{Adv}_{\mathcal{E}, \mathcal{A}}^{\text{ind}}(\lambda) \leq 2 \cdot \left(\text{Adv}^{2\text{-DQCSD}}(\lambda) + \text{Adv}^{3\text{-DQCSD}}(\lambda) \right)$$

Sequence of games from $\text{Enc}(\epsilon_0)$ to $\text{Enc}(\epsilon_1)$



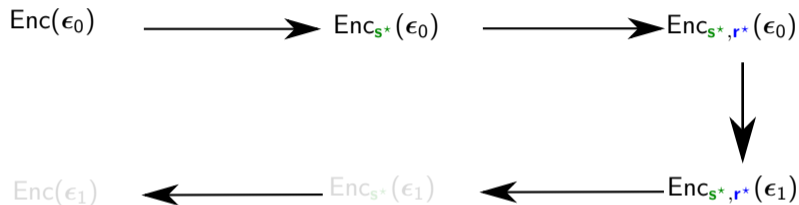
$$\text{Adv}_{\mathcal{E}, \mathcal{A}}^{\text{ind}}(\lambda) \leq 2 \cdot \left(\text{Adv}^{2\text{-DQCSD}}(\lambda) + \text{Adv}^{3\text{-DQCSD}}(\lambda) \right)$$

Sequence of games from $\text{Enc}(\epsilon_0)$ to $\text{Enc}(\epsilon_1)$



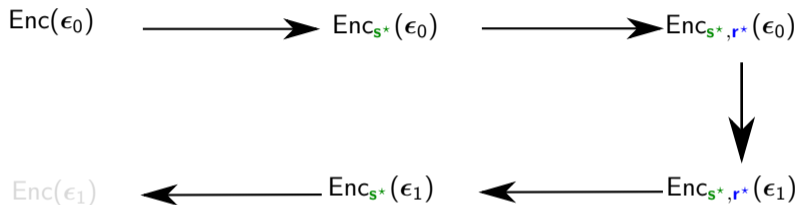
$$\text{Adv}_{\mathcal{E}, \mathcal{A}}^{\text{ind}}(\lambda) \leq 2 \cdot \left(\text{Adv}^{2\text{-DQCSD}}(\lambda) + \text{Adv}^{3\text{-DQCSD}}(\lambda) \right)$$

Sequence of games from $\text{Enc}(\epsilon_0)$ to $\text{Enc}(\epsilon_1)$



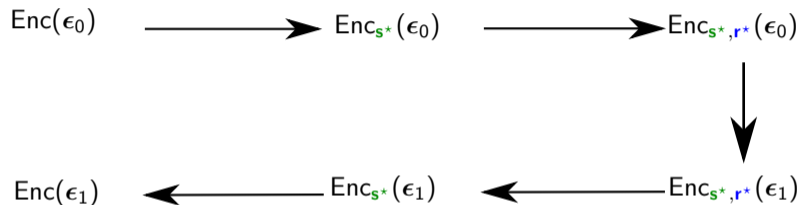
$$\text{Adv}_{\mathcal{E}, \mathcal{A}}^{\text{ind}}(\lambda) \leq 2 \cdot \left(\text{Adv}^{2\text{-DQCSD}}(\lambda) + \text{Adv}^{3\text{-DQCSD}}(\lambda) \right)$$

Sequence of games from $\text{Enc}(\epsilon_0)$ to $\text{Enc}(\epsilon_1)$



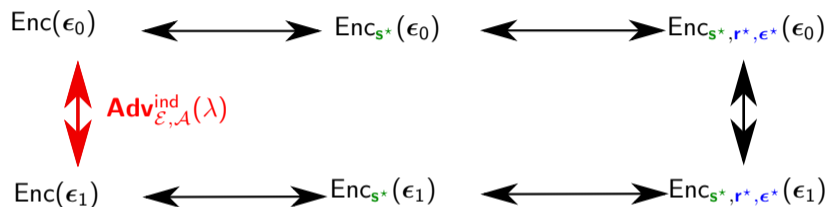
$$\text{Adv}_{\mathcal{E}, \mathcal{A}}^{\text{ind}}(\lambda) \leq 2 \cdot \left(\text{Adv}^{2\text{-DQCSD}}(\lambda) + \text{Adv}^{3\text{-DQCSD}}(\lambda) \right)$$

Sequence of games from $\text{Enc}(\epsilon_0)$ to $\text{Enc}(\epsilon_1)$



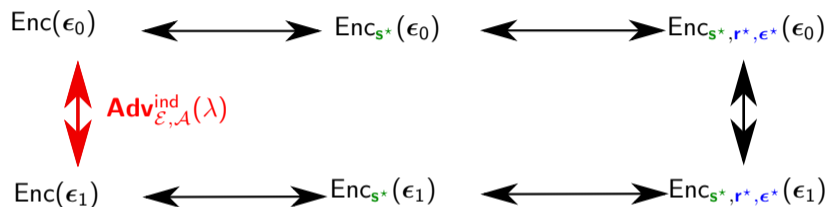
$$\text{Adv}_{\mathcal{E}, \mathcal{A}}^{\text{ind}}(\lambda) \leq 2 \cdot \left(\text{Adv}^{2\text{-DQCSD}}(\lambda) + \text{Adv}^{3\text{-DQCSD}}(\lambda) \right)$$

Sequence of games from $\text{Enc}(\epsilon_0)$ to $\text{Enc}(\epsilon_1)$



$$\text{Adv}_{\mathcal{E}, \mathcal{A}}^{\text{ind}}(\lambda) \leq 2 \cdot \left(\text{Adv}^{2\text{-DQCSD}}(\lambda) + \text{Adv}^{3\text{-DQCSD}}(\lambda) \right)$$

Sequence of games from $\text{Enc}(\epsilon_0)$ to $\text{Enc}(\epsilon_1)$



$$\text{Adv}_{\mathcal{E}, \mathcal{A}}^{\text{ind}}(\lambda) \leq 2 \cdot \left(\text{Adv}^{2\text{-DQCSD}}(\lambda) + \text{Adv}^{3\text{-DQCSD}}(\lambda) \right)$$

Outline

- 1 Preliminaries
- 2 Presentation of the Ouroboros protocol
- 3 Security
- 4 Parameters
 - Reduction Compliant
 - Best Known Attacks

Reduction Compliant Parameters

Instance	Ouroboros Parameters					
	n	w	w_e	threshold	security	DFR
Low-I	5,851	47	94	30	80	$0.92 \cdot 10^{-5}$
Low-II	5,923	47	94	30	80	$2.3 \cdot 10^{-6}$
Medium-I	13,691	75	150	45	128	$0.96 \cdot 10^{-5}$
Medium-II	14,243	75	150	45	128	$1.09 \cdot 10^{-6}$
Strong-I	40,013	147	294	85	256	$4.20 \cdot 10^{-5}$
Strong-II	40,973	147	294	85	256	$< 10^{-6}$

Table : Parameter sets for Ouroboros

Outline

- 1 Preliminaries
- 2 Presentation of the Ouroboros protocol
- 3 Security
- 4 Parameters
 - Reduction Compliant
 - Best Known Attacks

Parameters wrt Best Know Attacks

Instance	Ouroboros Optimized Parameters					
	n	w	w_e	threshold	security	DFR
Low-I	4,813	41	123	27	80	$2.23 \cdot 10^{-5}$
Low-II	5,003	41	123	27	80	$2.60 \cdot 10^{-6}$
Medium-I	10,301	67	201	42	128	$1.01 \cdot 10^{-4}$
Medium-II	10,837	67	201	42	128	$< 10^{-7}$
Strong-I	32,771	131	393	77	256	$< 10^{-4}$
Strong-II	33,997	131	393	77	256	$< 10^{-7}$

Table : Optimized parameter sets for Ouroboros in Hamming metric

Conclusion

In this talk

- Ouroboros: a *secure, simple, and efficient* code-based key exchange protocol
- Efficient decoding through BitFlip
- Competitive parameters

Further Improvements

Conclusion

In this talk

- Ouroboros: a *secure, simple, and efficient* code-based key exchange protocol
- Efficient decoding through BitFlip
- Competitive parameters

Further Improvements

Conclusion

In this talk

- Ouroboros: a *secure, simple, and efficient* code-based key exchange protocol
- Efficient decoding through BitFlip
- Competitive parameters

Further Improvements

Conclusion

In this talk

- Ouroboros: a *secure, simple, and efficient* code-based key exchange protocol
- Efficient decoding through BitFlip
- Competitive parameters

Further Improvements

- Improve BitFlip threshold [CS16]
- Switch to Rank metric → *interlude*
- Optimize implementation
- OpenSSL TLS integration

Conclusion

In this talk

- Ouroboros: a *secure, simple, and efficient* code-based key exchange protocol
- Efficient decoding through BitFlip
- Competitive parameters

Further Improvements

- Improve BitFlip threshold [CS16]
- Switch to Rank metric → *interlude*
- Optimize implementation
- OpenSSL TLS integration

Conclusion

In this talk

- Ouroboros: a *secure, simple, and efficient* code-based key exchange protocol
- Efficient decoding through BitFlip
- Competitive parameters

Further Improvements

- Improve BitFlip threshold [CS16]
- Switch to Rank metric → **interlude**
- Optimize implementation
- OpenSSL TLS integration

Rank Metric Interlude (1/2)

Rank metric defined over (finite) extensions of finite fields

- \mathbb{F}_q a finite field with q a power of a prime.
- \mathbb{F}_{q^m} an extension of degree m of \mathbb{F}_q .
- \mathbb{F}_{q^m} can be seen as a vector space on \mathbb{F}_q .
- $\mathcal{B} = (b_1, \dots, b_m)$ a basis of \mathbb{F}_{q^m} over \mathbb{F}_q .

Let $\mathbf{v} = (v_1, \dots, v_n)$ be a word of length n in \mathbb{F}_{q^m} .

Any coordinate $v_j = \sum_{i=1}^m v_{ij} b_i$ with $v_{ij} \in \mathbb{F}_q$.

$$\mathbf{v} = (v_1, \dots, v_n) \rightarrow \mathbf{V} = \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \dots & v_{mn} \end{pmatrix}$$

Rank weight of word

\mathbf{v} has rank $r = \text{rank}(\mathbf{v})$ iff the rank of $\mathbf{V} = (v_{ij})_{ij}$ is r .

Equivalently $\text{rank}(\mathbf{v}) = r \Leftrightarrow v_j \in V_r \subset \mathbb{F}_{q^m}^n$ with $\dim(V_r) = r$.

Rank Metric Interlude (1/2)

Rank metric defined over (finite) extensions of finite fields

- \mathbb{F}_q a finite field with q a power of a prime.
- \mathbb{F}_{q^m} an extension of degree m of \mathbb{F}_q .
- \mathbb{F}_{q^m} can be seen as a vector space on \mathbb{F}_q .
- $\mathcal{B} = (b_1, \dots, b_m)$ a basis of \mathbb{F}_{q^m} over \mathbb{F}_q .

Let $\mathbf{v} = (v_1, \dots, v_n)$ be a word of length n in \mathbb{F}_{q^m} .

Any coordinate $v_j = \sum_{i=1}^m v_{ij} b_i$ with $v_{ij} \in \mathbb{F}_q$.

$$\mathbf{v} = (v_1, \dots, v_n) \rightarrow \mathbf{V} = \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \dots & v_{mn} \end{pmatrix}$$

Rank weight of word

\mathbf{v} has rank $r = \text{rank}(\mathbf{v})$ iff the rank of $\mathbf{V} = (v_{ij})_{ij}$ is r .

Equivalently $\text{rank}(\mathbf{v}) = r \Leftrightarrow v_j \in V_r \subset \mathbb{F}_{q^m}^n$ with $\dim(V_r) = r$.

Rank Metric Interlude (1/2)

Rank metric defined over (finite) extensions of finite fields

- \mathbb{F}_q a finite field with q a power of a prime.
- \mathbb{F}_{q^m} an extension of degree m of \mathbb{F}_q .
- \mathbb{F}_{q^m} can be seen as a vector space on \mathbb{F}_q .
- $\mathcal{B} = (b_1, \dots, b_m)$ a basis of \mathbb{F}_{q^m} over \mathbb{F}_q .

Let $\mathbf{v} = (v_1, \dots, v_n)$ be a word of length n in \mathbb{F}_{q^m} .

Any coordinate $v_j = \sum_{i=1}^m v_{ij} b_i$ with $v_{ij} \in \mathbb{F}_q$.

$$\mathbf{v} = (v_1, \dots, v_n) \rightarrow \mathbf{V} = \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \dots & v_{mn} \end{pmatrix}$$

Rank weight of word

\mathbf{v} has rank $r = \text{rank}(\mathbf{v})$ iff the rank of $\mathbf{V} = (v_{ij})_{ij}$ is r .

Equivalently $\text{rank}(\mathbf{v}) = r \Leftrightarrow v_j \in V_r \subset \mathbb{F}_{q^m}^n$ with $\dim(V_r) = r$.

Rank Metric Interlude (1/2)

Rank metric defined over (finite) extensions of finite fields

- \mathbb{F}_q a finite field with q a power of a prime.
- \mathbb{F}_{q^m} an extension of degree m of \mathbb{F}_q .
- \mathbb{F}_{q^m} can be seen as a vector space on \mathbb{F}_q .
- $\mathcal{B} = (b_1, \dots, b_m)$ a basis of \mathbb{F}_{q^m} over \mathbb{F}_q .

Let $\mathbf{v} = (v_1, \dots, v_n)$ be a word of length n in \mathbb{F}_{q^m} .

Any coordinate $v_j = \sum_{i=1}^m v_{ij} b_i$ with $v_{ij} \in \mathbb{F}_q$.

$$\mathbf{v} = (v_1, \dots, v_n) \rightarrow \mathbf{V} = \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \dots & v_{mn} \end{pmatrix}$$

Rank weight of word

\mathbf{v} has rank $r = \text{rank}(\mathbf{v})$ iff the rank of $\mathbf{V} = (v_{ij})_{ij}$ is r .

Equivalently $\text{rank}(\mathbf{v}) = r \Leftrightarrow v_j \in V_r \subset \mathbb{F}_{q^m}^n$ with $\dim(V_r) = r$.

Rank Metric Interlude (1/2)

Rank metric defined over (finite) extensions of finite fields

- \mathbb{F}_q a finite field with q a power of a prime.
- \mathbb{F}_{q^m} an extension of degree m of \mathbb{F}_q .
- \mathbb{F}_{q^m} can be seen as a vector space on \mathbb{F}_q .
- $\mathcal{B} = (b_1, \dots, b_m)$ a basis of \mathbb{F}_{q^m} over \mathbb{F}_q .

Let $\mathbf{v} = (v_1, \dots, v_n)$ be a word of length n in \mathbb{F}_{q^m} .

Any coordinate $v_j = \sum_{i=1}^m v_{ij} b_i$ with $v_{ij} \in \mathbb{F}_q$.

$$\mathbf{v} = (v_1, \dots, v_n) \rightarrow \mathbf{V} = \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \dots & v_{mn} \end{pmatrix}$$

Rank weight of word

\mathbf{v} has rank $r = \text{rank}(\mathbf{v})$ iff the rank of $\mathbf{V} = (v_{ij})_{ij}$ is r .

Equivalently $\text{rank}(\mathbf{v}) = r \Leftrightarrow v_j \in V_r \subset \mathbb{F}_{q^m}^n$ with $\dim(V_r) = r$.

Rank Metric Interlude (1/2)

Rank metric defined over (finite) extensions of finite fields

- \mathbb{F}_q a finite field with q a power of a prime.
- \mathbb{F}_{q^m} an extension of degree m of \mathbb{F}_q .
- \mathbb{F}_{q^m} can be seen as a vector space on \mathbb{F}_q .
- $\mathcal{B} = (b_1, \dots, b_m)$ a basis of \mathbb{F}_{q^m} over \mathbb{F}_q .

Let $\mathbf{v} = (v_1, \dots, v_n)$ be a word of length n in \mathbb{F}_{q^m} .

Any coordinate $v_j = \sum_{i=1}^m v_{ij} b_i$ with $v_{ij} \in \mathbb{F}_q$.

$$\mathbf{v} = (v_1, \dots, v_n) \rightarrow \mathbf{V} = \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \dots & v_{mn} \end{pmatrix}$$

Rank weight of word

\mathbf{v} has rank $r = \text{rank}(\mathbf{v})$ iff the rank of $\mathbf{V} = (v_{ij})_{ij}$ is r .

Equivalently $\text{rank}(\mathbf{v}) = r \Leftrightarrow v_j \in V_r \subset \mathbb{F}_{q^m}^n$ with $\dim(V_r) = r$.

Rank Metric Interlude (1/2)

Rank metric defined over (finite) extensions of finite fields

- \mathbb{F}_q a finite field with q a power of a prime.
- \mathbb{F}_{q^m} an extension of degree m of \mathbb{F}_q .
- \mathbb{F}_{q^m} can be seen as a vector space on \mathbb{F}_q .
- $\mathcal{B} = (b_1, \dots, b_m)$ a basis of \mathbb{F}_{q^m} over \mathbb{F}_q .

Let $\mathbf{v} = (v_1, \dots, v_n)$ be a word of length n in \mathbb{F}_{q^m} .

Any coordinate $v_j = \sum_{i=1}^m v_{ij} b_i$ with $v_{ij} \in \mathbb{F}_q$.

$$\mathbf{v} = (v_1, \dots, v_n) \rightarrow \mathbf{V} = \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \dots & v_{mn} \end{pmatrix}$$

Rank weight of word

\mathbf{v} has rank $r = \text{rank}(\mathbf{v})$ iff the rank of $\mathbf{V} = (v_{ij})_{ij}$ is r .

Equivalently $\text{rank}(\mathbf{v}) = r \Leftrightarrow v_j \in V_r \subset \mathbb{F}_{q^m}^n$ with $\dim(V_r) = r$.

Rank Metric Interlude (1/2)

Rank metric defined over (finite) extensions of finite fields

- \mathbb{F}_q a finite field with q a power of a prime.
- \mathbb{F}_{q^m} an extension of degree m of \mathbb{F}_q .
- \mathbb{F}_{q^m} can be seen as a vector space on \mathbb{F}_q .
- $\mathcal{B} = (b_1, \dots, b_m)$ a basis of \mathbb{F}_{q^m} over \mathbb{F}_q .

Let $\mathbf{v} = (v_1, \dots, v_n)$ be a word of length n in \mathbb{F}_{q^m} .

Any coordinate $v_j = \sum_{i=1}^m v_{ij} b_i$ with $v_{ij} \in \mathbb{F}_q$.

$$\mathbf{v} = (v_1, \dots, v_n) \rightarrow \mathbf{V} = \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \dots & v_{mn} \end{pmatrix}$$

Rank weight of word

\mathbf{v} has rank $r = \text{rank}(\mathbf{v})$ iff the rank of $\mathbf{V} = (v_{ij})_{ij}$ is r .

Equivalently $\text{rank}(\mathbf{v}) = r \Leftrightarrow v_j \in V_r \subset \mathbb{F}_{q^m}^n$ with $\dim(V_r) = r$.

Rank Metric Interlude (2/2)

- Best Known Attacks have worse complexity in rank metric ($2^{\mathcal{O}(n^2)}$) than in Hamming metric ($2^{\mathcal{O}(n)}$)
- Consequence: worse attacks \Rightarrow better parameters

Instance	Ouroboros-R Parameters						
	key size (bits)	n	m	q	w	security	decoding failure
Ouroboros-R-I	1,591	37	43	2	5	100	10^{-4}
Ouroboros-R-II	2,809	53	53	2	5	128	10^{-8}
Ouroboros-R-III	3,953	59	67	2	6	192	10^{-7}
Ouroboros-R-IV	5,293	67	79	2	7	256	10^{-5}
Ouroboros-R-V	5,618	53	53	4	6	256	10^{-10}

Table : Parameter sets for Ouroboros-R in rank metric.

Rank Metric Interlude (2/2)

- Best Known Attacks have worse complexity in rank metric ($2^{\mathcal{O}(n^2)}$) than in Hamming metric ($2^{\mathcal{O}(n)}$)
- Consequence: worse attacks \Rightarrow better parameters

Instance	Ouroboros-R Parameters						
	key size (bits)	n	m	q	w	security	decoding failure
Ouroboros-R-I	1,591	37	43	2	5	100	10^{-4}
Ouroboros-R-II	2,809	53	53	2	5	128	10^{-8}
Ouroboros-R-III	3,953	59	67	2	6	192	10^{-7}
Ouroboros-R-IV	5,293	67	79	2	7	256	10^{-5}
Ouroboros-R-V	5,618	53	53	4	6	256	10^{-10}

Table : Parameter sets for Ouroboros-R in rank metric.

Rank Metric Interlude (2/2)

- Best Known Attacks have worse complexity in rank metric ($2^{\mathcal{O}(n^2)}$) than in Hamming metric ($2^{\mathcal{O}(n)}$)
- Consequence: worse attacks \Rightarrow better parameters

Instance	Ouroboros-R Parameters						
	key size (bits)	n	m	q	w	security	decoding failure
Ouroboros-R-I	1,591	37	43	2	5	100	10^{-4}
Ouroboros-R-II	2,809	53	53	2	5	128	10^{-8}
Ouroboros-R-III	3,953	59	67	2	6	192	10^{-7}
Ouroboros-R-IV	5,293	67	79	2	7	256	10^{-5}
Ouroboros-R-V	5,618	53	53	4	6	256	10^{-10}

Table : Parameter sets for Ouroboros-R in rank metric.

Conclusion

In this talk

- Ouroboros: a *secure, simple, and efficient* code-based key exchange protocol
- Efficient decoding through BitFlip
- Competitive parameters

Further Improvements

- Improve BitFlip threshold [CS16]
- Switch to Rank metric → **interlude**
- Optimize implementation
- OpenSSL TLS integration

Conclusion

In this talk

- Ouroboros: a *secure, simple, and efficient* code-based key exchange protocol
- Efficient decoding through BitFlip
- Competitive parameters

Further Improvements

- Improve BitFlip threshold [CS16]
- Switch to Rank metric → [interlude](#)
- Optimize implementation
- OpenSSL TLS integration

Conclusion

In this talk

- Ouroboros: a *secure, simple, and efficient* code-based key exchange protocol
- Efficient decoding through BitFlip
- Competitive parameters

Further Improvements

- Improve BitFlip threshold [CS16]
- Switch to Rank metric → [interlude](#)
- Optimize implementation
- OpenSSL TLS integration

Thanks!



Thanks!



Carlos Aguilar Melchor, Olivier Blazy, Jean Christophe Deneuville, Philippe Gaborit, and Gilles Zémor.
Efficient encryption from random quasi-cyclic codes.
CoRR, abs/1612.05572, 2016.



Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe.
Post-quantum key exchange - A new hope.
In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 327–343. USENIX Association, 2016.



Michael Alekhnovich.
More on average case vs approximation complexity.
In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 298–307, 2003.



Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila.
Post-quantum key exchange for the TLS protocol from the ring learning with errors problem.
In *2015 IEEE Symposium on Security and Privacy*, pages 553–570. IEEE Computer Society Press, May 2015.



Julia Chaudet and Nicolas Sendrier.
Worst case qc-mdpc decoder for mceliece cryptosystem.
In *Information Theory (ISIT), 2016 IEEE International Symposium on*, pages 1366–1370. IEEE, 2016.



Jintai Ding.
New cryptographic constructions using generalized learning with errors problem.
Cryptology ePrint Archive, Report 2012/387, 2012.



Jintai Ding, Xiang Xie, and Xiaodong Lin.
A simple provably secure key exchange scheme based on the learning with errors problem.
Cryptology ePrint Archive, Report 2012/688, 2012.



Javier Herranz, Dennis Hofheinz, and Eike Kiltz.
KEM/DEM: Necessary and sufficient conditions for secure hybrid encryption.
Cryptology ePrint Archive, Report 2006/265, 2006.



Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo SLM Barreto.
Mdpc-mceliece: New mceliece variants from moderate density parity-check codes.

In *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, pages 2069–2073. IEEE, 2013.



NIST – National Institute of Standards and Technology.
Submission requirements and evaluation criteria for the post-quantum cryptography standardization process (call for proposal), December 2016.



Chris Peikert.
Lattice cryptography for the internet.
In Michele Mosca, editor, *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings*, volume 8772 of *Lecture Notes in Computer Science*, pages 197–219. Springer, 2014.



Nicolas Sendrier.
Encoding information into constant weight words.
In *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on*, pages 435–438. IEEE, 2005.

