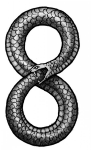
Ouroboros: a simple, secure and efficient key exchange protocol based on coding theory

Jean-Christophe Deneuville
<jean-christophe.deneuville@xlim.fr>

April the 27^{th}, 2017
Journées C2 – La Bresse

Pré-GDR Sécurité Informatique



Joint work with:

P. Gaborit
University of Limoges

G. Zémor
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; PACK
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; PACK
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; PACK
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, ...

Outline

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

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 $\rightarrow$ 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 $\rightarrow$ 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 $\rightarrow$ 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 $\rightarrow$ 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 $\rightarrow$ 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 $\rightarrow$ 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 $\rightarrow$ Secret data - Public data - One-time Randomness

Presentation

- Setup(1^λ): generates $n = n(\lambda)$, $k = k(\lambda)$, $\delta = \delta(\lambda)$, and $w = w(\lambda)$. Plaintext space is $\mathbb{F}_2^k$. param$= (n, k, \delta, w)$.
- KeyGen(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 $\mathcal{C}$. $\mathsf{sk} = (\mathbf{x}, \mathbf{y}) \xleftarrow{\$} \mathcal{V}^2$ such that $\omega(\mathbf{x}), \omega(\mathbf{y}) \leq w$, sets $\mathsf{pk} = \left(\mathbf{G}, \mathbf{Q}, \mathbf{s} = \mathsf{sk} \cdot \mathbf{Q}^\top, w\right)$, and returns $(\mathsf{pk}, \mathsf{sk})$.
- Encrypt($\mathsf{pk} = (\mathbf{G}, \mathbf{Q}, \mathbf{s}), \mu, \theta$): 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 pk.
- Decrypt($\mathsf{sk} = (\mathbf{x}, \mathbf{y}), \mathbf{c} = (\mathbf{v}, \boldsymbol{\rho})$): returns $\mathcal{C}$.Decode$(\boldsymbol{\rho} - \mathbf{v} \cdot \mathbf{y})$.

Presentation

- Setup(1^λ): generates $n = n(\lambda)$, $k = k(\lambda)$, $\delta = \delta(\lambda)$, and $w = w(\lambda)$. Plaintext space is $\mathbb{F}_2^k$. param$= (n, k, \delta, w)$.
- KeyGen(param): generates $\mathbf{q}_r \stackrel{\$}{\leftarrow} \mathcal{V}$, 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 $\mathcal{C}$. $\mathsf{sk} = (\mathbf{x}, \mathbf{y}) \stackrel{\$}{\leftarrow} \mathcal{V}^2$ such that $\omega(\mathbf{x}), \omega(\mathbf{y}) \leq w$, sets $\mathsf{pk} = \left(\mathbf{G}, \mathbf{Q}, \mathbf{s} = \mathsf{sk} \cdot \mathbf{Q}^\top, w\right)$, and returns $(\mathsf{pk}, \mathsf{sk})$.
- Encrypt($\mathsf{pk} = (\mathbf{G}, \mathbf{Q}, \mathbf{s}), \mu, \theta$): uses randomness θ to generate $\boldsymbol{\epsilon} \stackrel{\$}{\leftarrow} \mathcal{V}$, $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2) \stackrel{\$}{\leftarrow} \mathcal{V}^2$ 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 pk.
- Decrypt($\mathsf{sk} = (\mathbf{x}, \mathbf{y}), \mathbf{c} = (\mathbf{v}, \boldsymbol{\rho})$): returns $\mathcal{C}.\mathsf{Decode}(\boldsymbol{\rho} - \mathbf{v} \cdot \mathbf{y})$.

Presentation

- Setup(1^λ): generates $n = n(\lambda)$, $k = k(\lambda)$, $\delta = \delta(\lambda)$, and $w = w(\lambda)$. Plaintext space is $\mathbb{F}_2^k$. param$= (n, k, \delta, w)$.
- KeyGen(param): generates $\mathbf{q}_r \stackrel{\$}{\leftarrow} \mathcal{V}$, 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 $\mathcal{C}$. $\mathsf{sk} = (\mathbf{x}, \mathbf{y}) \stackrel{\$}{\leftarrow} \mathcal{V}^2$ such that $\omega(\mathbf{x}), \omega(\mathbf{y}) \leq w$, sets $\mathsf{pk} = \left(\mathbf{G}, \mathbf{Q}, \mathbf{s} = \mathsf{sk} \cdot \mathbf{Q}^\top, w\right)$, and returns $(\mathsf{pk}, \mathsf{sk})$.
- Encrypt($\mathsf{pk} = (\mathbf{G}, \mathbf{Q}, \mathbf{s}), \boldsymbol{\mu}, \theta$): uses randomness θ to generate $\boldsymbol{\epsilon} \stackrel{\$}{\leftarrow} \mathcal{V}$, $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2) \stackrel{\$}{\leftarrow} \mathcal{V}^2$ 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 pk.
- Decrypt($\mathsf{sk} = (\mathbf{x}, \mathbf{y}), \mathbf{c} = (\mathbf{v}, \boldsymbol{\rho})$): returns $\mathcal{C}$.Decode($\boldsymbol{\rho} - \mathbf{v} \cdot \mathbf{y}$).

Presentation

- Setup(1^λ): generates $n = n(\lambda)$, $k = k(\lambda)$, $\delta = \delta(\lambda)$, and $w = w(\lambda)$. Plaintext space is $\mathbb{F}_2^k$. param$= (n, k, \delta, w)$.
- KeyGen(param): generates $\mathbf{q}_r \stackrel{\$}{\leftarrow} \mathcal{V}$, 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 $\mathcal{C}$. $\mathsf{sk} = (\mathbf{x}, \mathbf{y}) \stackrel{\$}{\leftarrow} \mathcal{V}^2$ such that $\omega(\mathbf{x}), \omega(\mathbf{y}) \le w$, sets $\mathsf{pk} = \left(\mathbf{G}, \mathbf{Q}, \mathbf{s} = \mathsf{sk} \cdot \mathbf{Q}^\top, w\right)$, and returns $(\mathsf{pk}, \mathsf{sk})$.
- Encrypt($\mathsf{pk} = (\mathbf{G}, \mathbf{Q}, \mathbf{s}), \boldsymbol{\mu}, \theta$): uses randomness θ to generate $\boldsymbol{\epsilon} \stackrel{\$}{\leftarrow} \mathcal{V}$, $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2) \stackrel{\$}{\leftarrow} \mathcal{V}^2$ such that $\omega(\boldsymbol{\epsilon}), \omega(\mathbf{r}_1), \omega(\mathbf{r}_2) \le w$, 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 pk.
- Decrypt($\mathsf{sk} = (\mathbf{x}, \mathbf{y}), \mathbf{c} = (\mathbf{v}, \boldsymbol{\rho})$): returns $\mathcal{C}$.Decode$(\boldsymbol{\rho} - \mathbf{v} \cdot \mathbf{y})$.

Correctness

Correctness Property

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

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

the error term is **not too big**

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

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

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

Error distribution analysis $\rightarrow$ Decryption failure probability better understood

Correctness

Correctness Property

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

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

the error term is **not too big**

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

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

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

Error distribution analysis $\rightarrow$ Decryption failure probability better understood

Correctness

Correctness Property

$$\text{Decrypt}\left(\mathsf{sk}, \text{Encrypt}\left(\mathsf{pk}, \boldsymbol{\mu}, \theta\right)\right) = \boldsymbol{\mu}$$

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

the error term is **not too big**

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

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

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

Error distribution analysis $\rightarrow$ 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 $\mathbf{x}, \mathbf{y}, \mathbf{r}_1, \mathbf{r}_2 \xleftarrow{\$} \mathcal{S}_w^n(\mathbb{F}_2)$ with $w = \mathcal{O}(\sqrt{n})$, and $\mathbf{e} \xleftarrow{\$} \mathcal{S}_{cw}^n(\mathbb{F}_2)$ 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

$$\begin{pmatrix} \mathbf{x} & -\mathbf{y} \end{pmatrix} \begin{pmatrix} \mathbf{r}_2 \\ \mathbf{r}_1 \end{pmatrix} + \mathbf{e}$$

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 $\mathbf{x}, \mathbf{y}, \mathbf{r}_1, \mathbf{r}_2 \xleftarrow{\$} \mathcal{S}_w^n(\mathbb{F}_2)$ with $w = \mathcal{O}(\sqrt{n})$, and $\mathbf{e} \xleftarrow{\$} \mathcal{S}_{cw}^n(\mathbb{F}_2)$ 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

$$\begin{pmatrix} \mathbf{x} & -\mathbf{y} \end{pmatrix} \begin{pmatrix} \mathbf{r}_2 \\ \mathbf{r}_1 \end{pmatrix} + \mathbf{e}$$

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 $\mathbf{x}, \mathbf{y}, \mathbf{r}_1, \mathbf{r}_2 \xleftarrow{\$} \mathcal{S}_w^n(\mathbb{F}_2)$ with $w = \mathcal{O}(\sqrt{n})$, and $\mathbf{e} \xleftarrow{\$} \mathcal{S}_{cw}^n(\mathbb{F}_2)$ 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 $\mathbf{x}, \mathbf{y}, \mathbf{r}_1, \mathbf{r}_2 \xleftarrow{\$} \mathcal{S}_w^n(\mathbb{F}_2)$ with $w = \mathcal{O}(\sqrt{n})$, and $\mathbf{e} \xleftarrow{\$} \mathcal{S}_{cw}^n(\mathbb{F}_2)$ 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

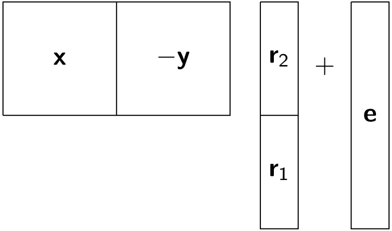
A particular decoding

- 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 $\mathbf{x}, \mathbf{y}, \mathbf{r}_1, \mathbf{r}_2 \xleftarrow{\$} \mathcal{S}_w^n(\mathbb{F}_2)$ with $w = \mathcal{O}(\sqrt{n})$, and $\mathbf{e} \xleftarrow{\$} \mathcal{S}_{cw}^n(\mathbb{F}_2)$ 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 **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]

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 Hash : $\{0,1\}^* \longrightarrow \mathcal{S}^n_{cw}(\mathbb{F}_2)$ [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

Alice		Bob
$\text{seed}_{\mathbf{h}} \xleftarrow{\$} \{0,1\}^\lambda$, $\mathbf{h} \xleftarrow{\text{seed}_{\mathbf{h}}} \mathbb{F}_2^n$		
$\mathbf{x}, \mathbf{y} \xleftarrow{\$} \mathcal{S}^n_w(\mathbb{F}_2)$, $\mathbf{s} \leftarrow \mathbf{x} + \mathbf{h}\mathbf{y}$	$\xrightarrow{\mathbf{h},\mathbf{s}}$	$\mathbf{r}_1, \mathbf{r}_2 \xleftarrow{\$} \mathcal{S}^n_w(\mathbb{F}_2)$
		$\mathbf{e}_\mathbf{r} \leftarrow \text{Hash}\left(\mathbf{r}_1, \mathbf{r}_2\right)$, $\mathbf{e} \xleftarrow{\$} \mathcal{S}^n_{w_e}(\mathbb{F}_2)$
	$\xleftarrow{\mathbf{s}_\mathbf{r}, \mathbf{s}_\mathbf{e}}$	$\mathbf{s}_\mathbf{r} \leftarrow \mathbf{r}_1 + \mathbf{h}\mathbf{r}_2$, $\mathbf{s}_\mathbf{e} \leftarrow \mathbf{s}\mathbf{r}_2 + \mathbf{e}_\mathbf{r} + \mathbf{e}$
$\mathbf{e}_\mathbf{c} \leftarrow \mathbf{s}_\mathbf{e} - \mathbf{y}\mathbf{s}_\mathbf{r}$		
$(\mathbf{r}_1, \mathbf{r}_2) \leftarrow \text{CE-Decoder}(\mathbf{x}, \mathbf{y}, \mathbf{e}_\mathbf{c}, t, w, w_e)$		
$\epsilon \leftarrow \mathbf{e}_\mathbf{c} - \mathbf{x}\mathbf{r}_2 + \mathbf{y}\mathbf{r}_1 - \text{Hash}(\mathbf{r}_1, \mathbf{r}_2)$		
$\boxed{\epsilon}$	SHARED SECRET	$\boxed{\epsilon}$

Ouroboros

- Requires a hash function Hash : $\{0,1\}^* \longrightarrow \mathcal{S}^n_{cw}(\mathbb{F}_2)$ [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

Alice		Bob
$\text{seed}_h \xleftarrow{\$} \{0,1\}^\lambda$, $h \xleftarrow{\text{seed}_h} \mathbb{F}_2^n$ $x, y \xleftarrow{\$} \mathcal{S}^n_w(\mathbb{F}_2)$, $s \leftarrow x + hy$	$\xrightarrow{h,s}$	
		$r_1, r_2 \xleftarrow{\$} \mathcal{S}^n_w(\mathbb{F}_2)$ $e_r \leftarrow \text{Hash}(r_1, r_2)$, $e \xleftarrow{\$} \mathcal{S}^n_{w_e}(\mathbb{F}_2)$ $s_r \leftarrow r_1 + hr_2$, $s_e \leftarrow sr_2 + e_r + e$
$e_c \leftarrow s_e - ys_r$ $(r_1, r_2) \leftarrow \text{CE-Decoder}(x, y, e_c, t, w, w_e)$ $\epsilon \leftarrow e_c - xr_2 + yr_1 - \text{Hash}(r_1, r_2)$	$\xleftarrow{s_r, s_e}$	
$\boxed{\epsilon}$	SHARED SECRET	$\boxed{\epsilon}$

Ouroboros

- Requires a hash function Hash : $\{0,1\}^* \longrightarrow \mathcal{S}^n_{cw}(\mathbb{F}_2)$ [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

Alice		Bob
$\mathsf{seed}_{\mathbf{h}} \xleftarrow{\$} \{0,1\}^\lambda$, $\mathbf{h} \xleftarrow{\mathsf{seed}_{\mathbf{h}}} \mathbb{F}_2^n$ $\mathbf{x}, \mathbf{y} \xleftarrow{\$} \mathcal{S}^n_w(\mathbb{F}_2)$, $\mathbf{s} \leftarrow \mathbf{x} + \mathbf{h}\mathbf{y}$	$\xrightarrow{\mathbf{h}, \mathbf{s}}$	$\mathbf{r}_1, \mathbf{r}_2 \xleftarrow{\$} \mathcal{S}^n_w(\mathbb{F}_2)$
	$\xleftarrow{\mathbf{s}_r, \mathbf{s}_e}$	$\mathbf{e}_r \leftarrow \mathrm{Hash}(\mathbf{r}_1, \mathbf{r}_2)$, $\mathbf{e} \xleftarrow{\$} \mathcal{S}^n_{w_e}(\mathbb{F}_2)$ $\mathbf{s}_r \leftarrow \mathbf{r}_1 + \mathbf{h}\mathbf{r}_2$, $\mathbf{s}_e \leftarrow \mathbf{s}\mathbf{r}_2 + \mathbf{e}_r + \mathbf{e}$
$\mathbf{e}_c \leftarrow \mathbf{s}_e - \mathbf{y}\mathbf{s}_r$ $(\mathbf{r}_1, \mathbf{r}_2) \leftarrow \text{CE-Decoder}(\mathbf{x}, \mathbf{y}, \mathbf{e}_c, t, w, w_e)$ $\epsilon \leftarrow \mathbf{e}_c - \mathbf{x}\mathbf{r}_2 + \mathbf{y}\mathbf{r}_1 - \mathrm{Hash}(\mathbf{r}_1, \mathbf{r}_2)$		
$\boxed{\epsilon}$	SHARED SECRET	$\boxed{\epsilon}$

Ouroboros

- Requires a hash function Hash : $\{0,1\}^* \longrightarrow \mathcal{S}^n_{cw}(\mathbb{F}_2)$ [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

Alice		Bob
$\mathbf{seed_h} \overset{\$}{\leftarrow} \{0,1\}^\lambda$, $\mathbf{h} \overset{\mathbf{seed_h}}{\leftarrow} \mathbb{F}_2^n$		
$\mathbf{x}, \mathbf{y} \overset{\$}{\leftarrow} \mathcal{S}^n_w(\mathbb{F}_2)$, $\mathbf{s} \leftarrow \mathbf{x} + \mathbf{h}\mathbf{y}$	$\xrightarrow{\mathbf{h},\mathbf{s}}$	$\mathbf{r}_1, \mathbf{r}_2 \overset{\$}{\leftarrow} \mathcal{S}^n_w(\mathbb{F}_2)$
		$\mathbf{e_r} \leftarrow \mathrm{Hash}(\mathbf{r}_1, \mathbf{r}_2)$, $\mathbf{e} \overset{\$}{\leftarrow} \mathcal{S}^n_{w_e}(\mathbb{F}_2)$
	$\xleftarrow{\mathbf{s_r}, \mathbf{s_e}}$	$\mathbf{s_r} \leftarrow \mathbf{r}_1 + \mathbf{h}\mathbf{r}_2$, $\mathbf{s_e} \leftarrow \mathbf{s}\mathbf{r}_2 + \mathbf{e_r} + \mathbf{e}$
$\mathbf{e_c} \leftarrow \mathbf{s_e} - \mathbf{y}\mathbf{s_r}$		
$(\mathbf{r}_1, \mathbf{r}_2) \leftarrow \text{CE-Decoder}(\mathbf{x}, \mathbf{y}, \mathbf{e_c}, t, w, w_e)$		
$\epsilon \leftarrow \mathbf{e_c} - \mathbf{x}\mathbf{r}_2 + \mathbf{y}\mathbf{r}_1 - \mathrm{Hash}(\mathbf{r}_1, \mathbf{r}_2)$		
$\boxed{\epsilon}$	SHARED SECRET	$\boxed{\epsilon}$

Ouroboros

- Requires a hash function Hash : $\{0,1\}^* \longrightarrow \mathcal{S}^n_{cw}(\mathbb{F}_2)$ [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

Alice		Bob
$\mathsf{seed}_{\mathbf{h}} \xleftarrow{\$} \{0,1\}^\lambda$, $\mathbf{h} \xleftarrow{\mathsf{seed}_{\mathbf{h}}} \mathbb{F}_2^n$ $\mathbf{x}, \mathbf{y} \xleftarrow{\$} \mathcal{S}^n_w(\mathbb{F}_2)$, $\mathbf{s} \leftarrow \mathbf{x} + \mathbf{h}\mathbf{y}$	$\xrightarrow{\mathbf{h},\mathbf{s}}$	$\mathbf{r}_1, \mathbf{r}_2 \xleftarrow{\$} \mathcal{S}^n_w(\mathbb{F}_2)$
		$\mathbf{e}_r \leftarrow \mathrm{Hash}(\mathbf{r}_1, \mathbf{r}_2)$, $\mathbf{e} \xleftarrow{\$} \mathcal{S}^n_{w_e}(\mathbb{F}_2)$ $\mathbf{s}_r \leftarrow \mathbf{r}_1 + \mathbf{h}\mathbf{r}_2$, $\mathbf{s}_e \leftarrow \mathbf{s}\mathbf{r}_2 + \mathbf{e}_r + \mathbf{e}$
$\mathbf{e}_c \leftarrow \mathbf{s}_e - \mathbf{y}\mathbf{s}_r$ $(\mathbf{r}_1, \mathbf{r}_2) \leftarrow \text{CE-Decoder}(\mathbf{x}, \mathbf{y}, \mathbf{e}_c, t, w, w_e)$ $\epsilon \leftarrow \mathbf{e}_c - \mathbf{x}\mathbf{r}_2 + \mathbf{y}\mathbf{r}_1 - \mathrm{Hash}(\mathbf{r}_1, \mathbf{r}_2)$	$\xleftarrow{\mathbf{s}_r, \mathbf{s}_e}$	
$\boxed{\epsilon}$	SHARED SECRET	$\boxed{\epsilon}$

Ouroboros

- Requires a hash function Hash : $\{0,1\}^* \longrightarrow \mathcal{S}^n_{cw}(\mathbb{F}_2)$ [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

Alice		Bob
$\mathsf{seed_h} \xleftarrow{\$} \{0,1\}^\lambda$, $\mathbf{h} \xleftarrow{\mathsf{seed_h}} \mathbb{F}_2^n$		
$\mathbf{x},\mathbf{y} \xleftarrow{\$} \mathcal{S}^n_w(\mathbb{F}_2)$, $\mathbf{s} \leftarrow \mathbf{x} + \mathbf{hy}$	$\xrightarrow{\mathbf{h},\mathbf{s}}$	$\mathbf{r}_1,\mathbf{r}_2 \xleftarrow{\$} \mathcal{S}^n_w(\mathbb{F}_2)$
		$\mathbf{e_r} \leftarrow \mathrm{Hash}\left(\mathbf{r}_1,\mathbf{r}_2\right)$, $\mathbf{e} \xleftarrow{\$} \mathcal{S}^n_{w_e}(\mathbb{F}_2)$
	$\xleftarrow{\mathbf{s_r},\mathbf{s_e}}$	$\mathbf{s_r} \leftarrow \mathbf{r}_1 + \mathbf{hr}_2$, $\mathbf{s_e} \leftarrow \mathbf{sr}_2 + \mathbf{e_r} + \mathbf{e}$
$\mathbf{e_c} \leftarrow \mathbf{s_e} - \mathbf{ys_r}$		
$(\mathbf{r}_1,\mathbf{r}_2) \leftarrow \text{CE-Decoder}(\mathbf{x},\mathbf{y},\mathbf{e_c},t,w,w_e)$		
$\epsilon \leftarrow \mathbf{e_c} - \mathbf{xr}_2 + \mathbf{yr}_1 - \mathrm{Hash}(\mathbf{r}_1,\mathbf{r}_2)$		
$\boxed{\epsilon}$	SHARED SECRET	$\boxed{\epsilon}$

Ouroboros

- Requires a hash function Hash : $\{0,1\}^* \longrightarrow \mathcal{S}^n_{cw}(\mathbb{F}_2)$ [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

Alice		Bob
$\mathsf{seed_h} \xleftarrow{\$} \{0,1\}^\lambda$, $\mathbf{h} \xleftarrow{\mathsf{seed_h}} \mathbb{F}_2^n$		
$\mathbf{x}, \mathbf{y} \xleftarrow{\$} \mathcal{S}^n_w(\mathbb{F}_2)$, $\mathbf{s} \leftarrow \mathbf{x} + \mathbf{hy}$	$\xrightarrow{\mathbf{h}, \mathbf{s}}$	$\mathbf{r}_1, \mathbf{r}_2 \xleftarrow{\$} \mathcal{S}^n_w(\mathbb{F}_2)$
		$\mathbf{e_r} \leftarrow \mathrm{Hash}(\mathbf{r}_1, \mathbf{r}_2)$, $\mathbf{e} \xleftarrow{\$} \mathcal{S}^n_{w_e}(\mathbb{F}_2)$
	$\xleftarrow{\mathbf{s_r}, \mathbf{s_e}}$	$\mathbf{s_r} \leftarrow \mathbf{r}_1 + \mathbf{hr}_2$, $\mathbf{s_e} \leftarrow \mathbf{sr}_2 + \mathbf{e_r} + \mathbf{e}$
$\mathbf{e_c} \leftarrow \mathbf{s_e} - \mathbf{ys_r}$		
$(\mathbf{r}_1, \mathbf{r}_2) \leftarrow \text{CE-Decoder}(\mathbf{x}, \mathbf{y}, \mathbf{e_c}, t, w, w_e)$		
$\epsilon \leftarrow \mathbf{e_c} - \mathbf{xr}_2 + \mathbf{yr}_1 - \mathrm{Hash}(\mathbf{r}_1, \mathbf{r}_2)$		
$\boxed{\epsilon}$	SHARED SECRET	$\boxed{\epsilon}$

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:

$\mathbf{Exp}_{\mathcal{E},\mathcal{A}}^{\mathsf{ind}-b}(\lambda)$
1. param $\leftarrow$ Setup(1^λ)
2. $(\mathsf{pk}, \mathsf{sk}) \leftarrow$ KeyGen(param)
3. $(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{pk})$
4. $\mathbf{c}^* \leftarrow$ Encrypt$(\mathsf{pk}, \boldsymbol{\mu}_b, \theta)$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : \mathbf{c}^*)$
6. RETURN b'

- Hybrid argument:
  1. Construct a sequence of games transitioning from Enc(μ_0) to Enc(μ_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:

$\mathbf{Exp}_{\mathcal{E},\mathcal{A}}^{\mathsf{ind}-b}(\lambda)$
1. param $\leftarrow$ Setup(1^λ)
2. $(\mathsf{pk}, \mathsf{sk}) \leftarrow$ KeyGen(param)
3. $(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1) \leftarrow \mathcal{A}(\mathtt{FIND} : \mathsf{pk})$
4. $\mathbf{c}^* \leftarrow$ Encrypt$(\mathsf{pk}, \boldsymbol{\mu}_b, \theta)$
5. $b' \leftarrow \mathcal{A}(\mathtt{GUESS} : \mathbf{c}^*)$
6. RETURN b'

- Hybrid argument:
  1. Construct a sequence of games transitioning from Enc(μ_0) to Enc(μ_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:

$\mathbf{Exp}^{\mathsf{ind}-b}_{\mathcal{E},\mathcal{A}}(\lambda)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1) \leftarrow \mathcal{A}(\mathtt{FIND} : \mathsf{pk})$
4. $\mathbf{c}^* \leftarrow \mathsf{Encrypt}(\mathsf{pk}, \boldsymbol{\mu}_b, \theta)$
5. $b' \leftarrow \mathcal{A}(\mathtt{GUESS} : \mathbf{c}^*)$
6. $\mathtt{RETURN}\ b'$

- Hybrid argument:
  1. Construct a sequence of games transitioning from $\mathsf{Enc}(\boldsymbol{\mu}_0)$ to $\mathsf{Enc}(\boldsymbol{\mu}_1)$
  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:

$\mathbf{Exp}_{\mathcal{E},\mathcal{A}}^{\mathsf{ind}-b}(\lambda)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1) \leftarrow \mathcal{A}(\mathtt{FIND} : \mathsf{pk})$
4. $\mathbf{c}^* \leftarrow \mathsf{Encrypt}(\mathsf{pk}, \boldsymbol{\mu}_b, \theta)$
5. $b' \leftarrow \mathcal{A}(\mathtt{GUESS} : \mathbf{c}^*)$
6. $\mathtt{RETURN}\ \ b'$

- Hybrid argument:
  1. Construct a sequence of games transitioning from $\mathsf{Enc}(\boldsymbol{\mu}_0)$ to $\mathsf{Enc}(\boldsymbol{\mu}_1)$
  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 $\mathbf{H} \stackrel{\$}{\leftarrow} \mathbb{F}^{(n-k)\times n}$ and $\mathbf{x} \stackrel{\$}{\leftarrow} \mathbb{F}^n$ 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 $\mathbf{y} \stackrel{\$}{\leftarrow} \mathbb{F}^n$, 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 $(\mathbf{H}, \mathbf{Hx}^{\top})$.

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 $\mathsf{Enc}(\epsilon_0)$ to $\mathsf{Enc}(\epsilon_1)$

$$\begin{tikzcd}
\mathsf{Enc}(\epsilon_0) \arrow[r] & \mathsf{Enc}_{\mathbf{s}^\star}(\epsilon_0) \arrow[r] & \mathsf{Enc}_{\mathbf{s}^\star,\mathbf{r}^\star}(\epsilon_0) \arrow[d] \\
\mathsf{Enc}(\epsilon_1) & \mathsf{Enc}_{\mathbf{s}^\star}(\epsilon_1) \arrow[l] & \mathsf{Enc}_{\mathbf{s}^\star,\mathbf{r}^\star}(\epsilon_1) \arrow[l]
\end{tikzcd}$$

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

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

$$\begin{tikzcd}
\mathsf{Enc}(\epsilon_0) \arrow[r] & \mathsf{Enc}_{\mathbf{s}^\star}(\epsilon_0) \arrow[r] & \mathsf{Enc}_{\mathbf{s}^\star,\mathbf{r}^\star}(\epsilon_0) \arrow[d] \\
\mathsf{Enc}(\epsilon_1) & \mathsf{Enc}_{\mathbf{s}^\star}(\epsilon_1) \arrow[l] & \mathsf{Enc}_{\mathbf{s}^\star,\mathbf{r}^\star}(\epsilon_1) \arrow[l]
\end{tikzcd}$$

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

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

$$\begin{tikzcd}
\mathsf{Enc}(\epsilon_0) \arrow[r] & \mathsf{Enc}_{\mathbf{s}^\star}(\epsilon_0) \arrow[r] & \mathsf{Enc}_{\mathbf{s}^\star,\mathbf{r}^\star}(\epsilon_0) \arrow[d] \\
\mathsf{Enc}(\epsilon_1) & \mathsf{Enc}_{\mathbf{s}^\star}(\epsilon_1) \arrow[l] & \mathsf{Enc}_{\mathbf{s}^\star,\mathbf{r}^\star}(\epsilon_1) \arrow[l]
\end{tikzcd}$$

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

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

$$\begin{tikzcd}
\mathrm{Enc}(\epsilon_0) \arrow[r] & \mathrm{Enc}_{\mathbf{s}^\star}(\epsilon_0) \arrow[r] & \mathrm{Enc}_{\mathbf{s}^\star,\mathbf{r}^\star}(\epsilon_0) \arrow[d] \\
\mathrm{Enc}(\epsilon_1) & \mathrm{Enc}_{\mathbf{s}^\star}(\epsilon_1) \arrow[l] & \mathrm{Enc}_{\mathbf{s}^\star,\mathbf{r}^\star}(\epsilon_1) \arrow[l]
\end{tikzcd}$$

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

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

$$\begin{tikzcd}
\mathsf{Enc}(\epsilon_0) \arrow[r] & \mathsf{Enc}_{\mathbf{s}^\star}(\epsilon_0) \arrow[r] & \mathsf{Enc}_{\mathbf{s}^\star,\mathbf{r}^\star}(\epsilon_0) \arrow[d] \\
\mathsf{Enc}(\epsilon_1) & \mathsf{Enc}_{\mathbf{s}^\star}(\epsilon_1) \arrow[l] & \mathsf{Enc}_{\mathbf{s}^\star,\mathbf{r}^\star}(\epsilon_1) \arrow[l]
\end{tikzcd}$$

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

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

$$\begin{tikzcd}
\mathsf{Enc}(\epsilon_0) \arrow[r] & \mathsf{Enc}_{\mathbf{s}^\star}(\epsilon_0) \arrow[r] & \mathsf{Enc}_{\mathbf{s}^\star,\mathbf{r}^\star}(\epsilon_0) \arrow[d] \\
\mathsf{Enc}(\epsilon_1) & \mathsf{Enc}_{\mathbf{s}^\star}(\epsilon_1) \arrow[l] & \mathsf{Enc}_{\mathbf{s}^\star,\mathbf{r}^\star}(\epsilon_1) \arrow[l]
\end{tikzcd}$$

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

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

$$\begin{tikzcd}
\mathsf{Enc}(\epsilon_0) \arrow[r, leftrightarrow] \arrow[d, leftrightarrow, "\mathbf{Adv}^{\mathsf{ind}}_{\mathcal{E},\mathcal{A}}(\lambda)"] & \mathsf{Enc}_{\mathbf{s}^\star}(\epsilon_0) \arrow[r, leftrightarrow] & \mathsf{Enc}_{\mathbf{s}^\star,\mathbf{r}^\star,\boldsymbol{\epsilon}^\star}(\epsilon_0) \arrow[d, leftrightarrow] \\
\mathsf{Enc}(\epsilon_1) \arrow[r, leftrightarrow] & \mathsf{Enc}_{\mathbf{s}^\star}(\epsilon_1) \arrow[r, leftrightarrow] & \mathsf{Enc}_{\mathbf{s}^\star,\mathbf{r}^\star,\boldsymbol{\epsilon}^\star}(\epsilon_1)
\end{tikzcd}$$

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

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

$$\begin{tikzcd}
\mathsf{Enc}(\epsilon_0) \arrow[r, leftrightarrow] \arrow[d, leftrightarrow, "\mathbf{Adv}^{\mathrm{ind}}_{\mathcal{E},\mathcal{A}}(\lambda)"] & \mathsf{Enc}_{\mathbf{s}^\star}(\epsilon_0) \arrow[r, leftrightarrow] & \mathsf{Enc}_{\mathbf{s}^\star,\mathbf{r}^\star,\boldsymbol{\epsilon}^\star}(\epsilon_0) \arrow[d, leftrightarrow] \\
\mathsf{Enc}(\epsilon_1) \arrow[r, leftrightarrow] & \mathsf{Enc}_{\mathbf{s}^\star}(\epsilon_1) \arrow[r, leftrightarrow] & \mathsf{Enc}_{\mathbf{s}^\star,\mathbf{r}^\star,\boldsymbol{\epsilon}^\star}(\epsilon_1)
\end{tikzcd}$$

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

Outline

1 Preliminaries

2 Presentation of the Ouroboros protocol

3 Security

4 Parameters
- Reduction Compliant
- Best Known Attacks

Reduction Compliant Parameters

	Ouroboros Parameters					
Instance	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

	Ouroboros Optimized Parameters					
Instance	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 $\rightarrow$ 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 $\mathbb{F}_q$.
- $\mathcal{B} = (b_1, ..., b_m)$ a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$.

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

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

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

Rank weight of word

$\mathbf{v}$ has rank $r = \mathrm{rank}(\mathbf{v})$ iff the rank of $\mathbf{V} = (v_{ij})_{ij}$ is r.
Equivalently $\mathrm{rank}(\mathbf{v}) = r \Leftrightarrow v_j \in V_r \subset \mathbb{F}_{q^m}^n$ 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 $r = \mathrm{rank}(\mathbf{v})$ iff the rank of $\mathbf{V} = (v_{ij})_{ij}$ is r.
Equivalently $\mathrm{rank}(\mathbf{v}) = r \Leftrightarrow v_j \in V_r \subset \mathbb{F}_{q^m}^n$ 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 $\mathbb{F}_q$.
- $\mathcal{B} = (b_1, ..., b_m)$ a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$.

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

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

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

Rank weight of word

$\mathbf{v}$ has rank $r = \mathrm{rank}(\mathbf{v})$ iff the rank of $\mathbf{V} = (v_{ij})_{ij}$ is r.
Equivalently $\mathrm{rank}(\mathbf{v}) = r \Leftrightarrow v_j \in V_r \subset \mathbb{F}_{q^m}^n$ 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 $\mathbb{F}_q$.
- $\mathcal{B} = (b_1, ..., b_m)$ a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$.

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

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

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

Rank weight of word

$\mathbf{v}$ has rank $r = \mathrm{rank}(\mathbf{v})$ iff the rank of $\mathbf{V} = (v_{ij})_{ij}$ is r.
Equivalently $\mathrm{rank}(\mathbf{v}) = r \Leftrightarrow v_j \in V_r \subset \mathbb{F}_{q^m}^n$ 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 $\mathbb{F}_q$.
- $\mathcal{B} = (b_1, ..., b_m)$ a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$.

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

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

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

Rank weight of word

$\mathbf{v}$ has rank $r = \mathrm{rank}(\mathbf{v})$ iff the rank of $\mathbf{V} = (v_{ij})_{ij}$ is r.
Equivalently $\mathrm{rank}(\mathbf{v}) = r \Leftrightarrow v_j \in V_r \subset \mathbb{F}_{q^m}^n$ 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 $\mathbb{F}_q$.
- $\mathcal{B} = (b_1, ..., b_m)$ a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$.

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

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

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

Rank weight of word

$\mathbf{v}$ has rank $r = \mathrm{rank}(\mathbf{v})$ iff the rank of $\mathbf{V} = (v_{ij})_{ij}$ is r.
Equivalently $\mathrm{rank}(\mathbf{v}) = r \Leftrightarrow v_j \in V_r \subset \mathbb{F}_{q^m}^n$ 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 $\mathbb{F}_q$.
- $\mathcal{B} = (b_1, ..., b_m)$ 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 $v_{ij} \in \mathbb{F}_q$.

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

Rank weight of word

$\mathbf{v}$ has rank $r = \mathrm{rank}(\mathbf{v})$ iff the rank of $\mathbf{V} = (v_{ij})_{ij}$ is r.
Equivalently $\mathrm{rank}(\mathbf{v}) = r \Leftrightarrow v_j \in V_r \subset \mathbb{F}_{q^m}^n$ 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 $\mathbb{F}_q$.
- $\mathcal{B} = (b_1, ..., b_m)$ a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$.

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

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

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

Rank weight of word

$\mathbf{v}$ has rank $r = \mathrm{rank}(\mathbf{v})$ iff the rank of $\mathbf{V} = (v_{ij})_{ij}$ is r.
Equivalently $\mathrm{rank}(\mathbf{v}) = r \Leftrightarrow v_j \in V_r \subset \mathbb{F}_{q^m}^n$ 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

	Ouroboros-R Parameters						
Instance	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

	Ouroboros-R Parameters						
Instance	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 Chaulet 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.