Code-based Cryptography: State of the Art

Part I

Edoardo Persichetti

18 March 2019

Charles E. Schmidt College of Science
Florida Atlantic University

In This Talk

- Motivation
- Intro: a bit of Background
- Conservative Code-Based Cryptography
- Considerations

Part I

MOTIVATION

POST-QUANTUM CRYPTOGRAPHY

In a few years time large-scale quantum computers might be reality.

POST-QUANTUM CRYPTOGRAPHY

In a few years time large-scale quantum computers might be reality.
But then (Shor, '95):

Post-Quantum Cryptography

In a few years time large-scale quantum computers might be reality.
But then (Shor, '95):

- RSA
- DSA
- ECC
- Diffie-Hellman key exchange
- and many others ... **not secure** !

POST-QUANTUM CRYPTOGRAPHY

In a few years time large-scale quantum computers might be reality.
But then (Shor, '95):

- RSA
- DSA
- ECC
- Diffie-Hellman key exchange
- and many others ... **not secure** !

→ NIST's Post-Quantum Cryptography Standardization Call

Post-Quantum Cryptography

In a few years time large-scale quantum computers might be reality.
But then (Shor, ’95):

- RSA
- DSA
- ECC
- Diffie-Hellman key exchange
- and many others ... **not secure** !

$\rightarrow$ NIST’s Post-Quantum Cryptography Standardization Call

Main areas of research:

- Lattice-based cryptography.
- Hash-based cryptography.
- Code-based cryptography (McEliece, Niederreiter).
- Multivariate cryptography.
- Isogeny-based cryptography.

Part II

INTRO: A BIT OF BACKGROUND

What is Code-Based Cryptography?

WHAT IS CODE-BASED CRYPTOGRAPHY?

The family of cryptographic primitives based on the following.

What is Code-Based Cryptography?

The family of cryptographic primitives based on the following.

Problem (Computational Syndrome Decoding)

Given: $H \in \mathbb{F}_q^{(n-k)\times n}$, $y \in \mathbb{F}_q^{(n-k)}$ and $t \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ with $wt(e) \leq t$ such that $He^T = y$.

What is Code-Based Cryptography?

The family of cryptographic primitives based on the following.

Problem (Computational Syndrome Decoding)

Given: $H \in \mathbb{F}_q^{(n-k)\times n}$, $y \in \mathbb{F}_q^{(n-k)}$ *and* $t \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ *with* $wt(e) \leq t$ *such that* $He^T = y$.

Decisional version: NP-Complete (Berlekamp, McEliece and van Tilborg, 1978).

What is Code-Based Cryptography?

The family of cryptographic primitives based on the following.

Problem (Computational Syndrome Decoding)

Given: $H \in \mathbb{F}_q^{(n-k)\times n}$, $y \in \mathbb{F}_q^{(n-k)}$ *and* $t \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ *with* $wt(e) \leq t$ *such that* $He^T = y$.

Decisional version: NP-Complete (Berlekamp, McEliece and van Tilborg, 1978).

Unique solution when t is below a certain threshold.

What is Code-Based Cryptography?

The family of cryptographic primitives based on the following.

Problem (Computational Syndrome Decoding)

Given: $H \in \mathbb{F}_q^{(n-k)\times n}$, $y \in \mathbb{F}_q^{(n-k)}$ *and* $t \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ *with* $wt(e) \leq t$ *such that* $He^T = y$.

Decisional version: NP-Complete (Berlekamp, McEliece and van Tilborg, 1978).

Unique solution when t is below a certain threshold.

GV Bound

For a given finite field $\mathbb{F}_q$ and integers n, k, the Gilbert-Varshamov (GV) distance is the largest integer d_0 such that

$$|\mathcal{B}(0, d_0 - 1)| \leq q^{n-k}$$

where $\mathcal{B}(x, r) = \{y \in \mathbb{F}_q^n \mid d(x, y) \leq r\}$ is the n-dimensional ball of radius r centered in x.

Error-Correcting Codes

$[n, k]$ linear code over $\mathbb{F}_q$

A subspace of dimension k of $\mathbb{F}_q^n$.
t-error correcting: $\exists$ algorithm that corrects up to t errors.

$[n, k]$ LINEAR CODE OVER $\mathbb{F}_q$

A subspace of dimension k of $\mathbb{F}_q^n$.
t-error correcting: $\exists$ algorithm that corrects up to t errors.

HAMMING METRIC

$wt(x) = |\{i : x_i \neq 0, 1 \leq i \leq n\}|$, $d(x, y) = wt(x - y)$.

Minimum distance (of $\mathcal{C}$): $\min\{d(x, y) : x, y \in \mathcal{C}\}$.

Error-Correcting Codes

$[n, k]$ linear code over $\mathbb{F}_q$

A subspace of dimension k of $\mathbb{F}_q^n$.
t-error correcting: $\exists$ algorithm that corrects up to t errors.

Hamming metric

$wt(x) = |\{i : x_i \neq 0, 1 \leq i \leq n\}|$, $d(x, y) = wt(x - y)$.
Minimum distance (of $\mathcal{C}$): $\min\{d(x, y) : x, y \in \mathcal{C}\}$.

Generator matrix

$G \in \mathbb{F}_q^{k \times n}$ defines the code as follows: $x \in \mathcal{C}_G \iff x = \mu G$ for $\mu \in \mathbb{F}_q^k$.
Systematic form: $(I_k | M)$.

Error-Correcting Codes

$[n, k]$ linear code over $\mathbb{F}_q$

A subspace of dimension k of $\mathbb{F}_q^n$.
t-error correcting: $\exists$ algorithm that corrects up to t errors.

Hamming metric

$wt(x) = |\{i : x_i \neq 0, 1 \leq i \leq n\}|$, $d(x, y) = wt(x - y)$.
Minimum distance (of $\mathcal{C}$): $\min\{d(x, y) : x, y \in \mathcal{C}\}$.

Generator matrix

$G \in \mathbb{F}_q^{k \times n}$ defines the code as follows: $x \in \mathcal{C}_G \iff x = \mu G$ for $\mu \in \mathbb{F}_q^k$.
Systematic form: $(I_k | M)$.

Parity-check matrix

$H \in \mathbb{F}_q^{(n-k) \times n}$ defines the code as follows: $x \in \mathcal{C}_H \iff Hx^T = 0$.
Systematic form: $(M^T | I_{n-k})$.

(DE)CODING PROBLEMS

In general, it is hard to decode random codes.

(De)Coding Problems

In general, it is hard to decode random codes.

Problem (General Decoding)

Given: $G \in \mathbb{F}_q^{k \times n}$, $y \in \mathbb{F}_q^n$ and $t \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ with $wt(e) \leq t$ such that $y - e \in \mathcal{C}_G$.

(DE)CODING PROBLEMS

In general, it is hard to decode random codes.

PROBLEM (GENERAL DECODING)

Given: $G \in \mathbb{F}_q^{k \times n}$, $y \in \mathbb{F}_q^n$ *and* $t \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ *with* $wt(e) \leq t$ *such that* $y - e \in \mathcal{C}_G$.

Easy to see the two problems are equivalent.

(DE)CODING PROBLEMS

In general, it is hard to decode random codes.

PROBLEM (GENERAL DECODING)

Given: $G \in \mathbb{F}_q^{k\times n}$, $y \in \mathbb{F}_q^n$ *and* $t \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ *with* $wt(e) \leq t$ *such that* $y - e \in \mathcal{C}_G$.

Easy to see the two problems are equivalent.

To get trapdoor, need one more ingredient.

(De)Coding Problems

In general, it is hard to decode random codes.

Problem (General Decoding)

Given: $G \in \mathbb{F}_q^{k \times n}$, $y \in \mathbb{F}_q^n$ *and* $t \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ *with* $wt(e) \leq t$ *such that* $y - e \in \mathcal{C}_G$.

Easy to see the two problems are equivalent.

To get trapdoor, need one more ingredient.

Assumption (Code Indistinguishability)

Let M be a matrix defining a code. Then M is indistinguishable from a randomly generated matrix of the same size.

(De)Coding Problems

In general, it is hard to decode random codes.

Problem (General Decoding)

Given: $G \in \mathbb{F}_q^{k\times n}$, $y \in \mathbb{F}_q^n$ *and* $t \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ *with* $wt(e) \leq t$ *such that* $y - e \in \mathcal{C}_G$.

Easy to see the two problems are equivalent.

To get trapdoor, need one more ingredient.

Assumption (Code Indistinguishability)

Let M be a matrix defining a code. Then M is indistinguishable from a randomly generated matrix of the same size.

Choose a code family with efficient decoding algorithm associated to description Δ and hide the structure.

(De)Coding Problems

In general, it is hard to decode random codes.

Problem (General Decoding)

Given: $G \in \mathbb{F}_q^{k \times n}$, $y \in \mathbb{F}_q^n$ *and* $t \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ *with* $wt(e) \leq t$ *such that* $y - e \in \mathcal{C}_G$.

Easy to see the two problems are equivalent.

To get trapdoor, need one more ingredient.

Assumption (Code Indistinguishability)

Let M be a matrix defining a code. Then M is indistinguishable from a randomly generated matrix of the same size.

Choose a code family with efficient decoding algorithm associated to description Δ and hide the structure.

Hardness of assumption depends on chosen code family.

Code-based Cryptosystems

McEliece: first proposal (1978), based on GDP.

McEliece: first proposal (1978), based on GDP.

Chosen code family: binary Goppa codes.

CODE-BASED CRYPTOSYSTEMS

McEliece: first proposal (1978), based on GDP.

Chosen code family: binary Goppa codes.

KeyGen chooses generator matrix G and forms public key as SGP.

Code-based Cryptosystems

McEliece: first proposal (1978), based on GDP.

Chosen code family: binary Goppa codes.

KeyGen chooses generator matrix G and forms public key as SGP.

Plaintext is encrypted as noisy codeword (scheme is probabilistic).

McEliece: first proposal (1978), based on GDP.

Chosen code family: binary Goppa codes.

KeyGen chooses generator matrix G and forms public key as SGP.

Plaintext is encrypted as noisy codeword (scheme is probabilistic).

Niederreiter: “dual”/equivalent version (1985), based on SDP.

McEliece: first proposal (1978), based on GDP.

Chosen code family: binary Goppa codes.

KeyGen chooses generator matrix G and forms public key as SGP.

Plaintext is encrypted as noisy codeword (scheme is probabilistic).

Niederreiter: "dual"/equivalent version (1985), based on SDP.

Chosen code family: Generalized Reed-Solomon (GRS) codes.

Code-based Cryptosystems

McEliece: first proposal (1978), based on GDP.

Chosen code family: binary Goppa codes.

KeyGen chooses generator matrix G and forms public key as SGP.

Plaintext is encrypted as noisy codeword (scheme is probabilistic).

Niederreiter: "dual"/equivalent version (1985), based on SDP.

Chosen code family: Generalized Reed-Solomon (GRS) codes.

KeyGen chooses parity-check matrix H and forms public key as SHP.

McEliece: first proposal (1978), based on GDP.

Chosen code family: binary Goppa codes.

KeyGen chooses generator matrix G and forms public key as SGP.

Plaintext is encrypted as noisy codeword (scheme is probabilistic).

Niederreiter: "dual"/equivalent version (1985), based on SDP.

Chosen code family: Generalized Reed-Solomon (GRS) codes.

KeyGen chooses parity-check matrix H and forms public key as SHP.

Plaintext is encrypted as low-weight vector (scheme is deterministic).

McEliece PKE (Modern)

Key Generation

- Choose t-error correcting code $\mathcal{C}$.
- SK: code description Δ for $\mathcal{C}$.
- PK: generator matrix G in systematic form for $\mathcal{C}$.

McEliece PKE (Modern)

Key Generation

- Choose t-error correcting code $\mathcal{C}$.
- *SK*: code description Δ for $\mathcal{C}$.
- *PK*: generator matrix G in systematic form for $\mathcal{C}$.

Encryption

- Plaintext is a word $\mu \in \mathbb{F}_2^k$.
- Select random error vector $e \in \mathbb{F}_2^n$ of weight t.
- $c = \mu G + e$.

McEliece PKE (Modern)

Key Generation

- Choose t-error correcting code $\mathcal{C}$.
- SK: code description Δ for $\mathcal{C}$.
- PK: generator matrix G in systematic form for $\mathcal{C}$.

Encryption

- Plaintext is a word $\mu \in \mathbb{F}_2^k$.
- Select random error vector $e \in \mathbb{F}_2^n$ of weight t.
- $c = \mu G + e$.

Decryption

- Set $\mu = Decode_\Delta(c)$ and return μ.
- Return $\perp$ if decoding fails.

Niederreiter PKE (Modern)

Key Generation

- Choose t-error correcting code $\mathcal{C}$.
- SK: code description Δ for $\mathcal{C}$.
- PK: parity-check matrix H in systematic form for $\mathcal{C}$.

Niederreiter PKE (Modern)

Key Generation

- Choose t-error correcting code $\mathcal{C}$.
- SK: code description Δ for $\mathcal{C}$.
- PK: parity-check matrix H in systematic form for $\mathcal{C}$.

Encryption

- Plaintext is a word $e \in \mathbb{F}_2^n$ of weight t.
- $c = He^T$.

Niederreiter PKE (Modern)

Key Generation

- Choose t-error correcting code $\mathcal{C}$.
- SK: code description Δ for $\mathcal{C}$.
- PK: parity-check matrix H in systematic form for $\mathcal{C}$.

Encryption

- Plaintext is a word $e \in \mathbb{F}_2^n$ of weight t.
- $c = He^T$.

Decryption

- Set $e = Decode_\Delta(c)$ and return e.
- Return $\perp$ if decoding fails.

SECURITY

Both encryption schemes are only OW-CPA (OW-Passive) secure.

SECURITY

Both encryption schemes are only OW-CPA (OW-Passive) secure.

Given that assumption is true, best attack is generic search on random codes.

SECURITY

Both encryption schemes are only OW-CPA (OW-Passive) secure.

Given that assumption is true, best attack is generic search on random codes.

Paradigm: Information Set Decoding (ISD)(Prange,1962).

SECURITY

Both encryption schemes are only OW-CPA (OW-Passive) secure.

Given that assumption is true, best attack is generic search on random codes.

Paradigm: Information Set Decoding (ISD)(Prange,1962).

In a nutshell: look for Information Set (set of columns carrying the information symbols) which is error-free.

Both encryption schemes are only OW-CPA (OW-Passive) secure.

Given that assumption is true, best attack is generic search on random codes.

Paradigm: Information Set Decoding (ISD)(Prange,1962).

In a nutshell: look for Information Set (set of columns carrying the information symbols) which is error-free.

Several variants use Birthday Paradox and other tricks to obtain some speed-ups.

Both encryption schemes are only OW-CPA (OW-Passive) secure.

Given that assumption is true, best attack is generic search on random codes.

Paradigm: Information Set Decoding (ISD)(Prange,1962).

In a nutshell: look for Information Set (set of columns carrying the information symbols) which is error-free.

Several variants use Birthday Paradox and other tricks to obtain some speed-ups.

Complexity $2^{t(c+o(1))}$, constant c depending on algorithm, code and error rate.

SECURITY

Both encryption schemes are only OW-CPA (OW-Passive) secure.

Given that assumption is true, best attack is generic search on random codes.

Paradigm: Information Set Decoding (ISD)(Prange,1962).

In a nutshell: look for Information Set (set of columns carrying the information symbols) which is error-free.

Several variants use Birthday Paradox and other tricks to obtain some speed-ups.

Complexity $2^{t(c+o(1))}$, constant c depending on algorithm, code and error rate.

Use ISD as a tool to assess security level.

Part III

Conservative Code-Based Cryptography

Overall Strategy

1. Code family.

OVERALL STRATEGY

1. Code family.

Nearly every code choice has been shown to be insecure

- GRS
- Reed-Muller
- Concatenated
- Elliptic
- ...

Overall Strategy

1. Code family.

Nearly every code choice has been shown to be insecure

- GRS
- Reed-Muller
- Concatenated
- Elliptic
- ...

$\rightarrow$ Plain binary Goppa codes secure for 40 years.

Overall Strategy

1. Code family.

Nearly every code choice has been shown to be insecure

- GRS
- Reed-Muller
- Concatenated
- Elliptic
- ...

$\rightarrow$ Plain binary Goppa codes secure for 40 years.

2. Protocol.

Overall Strategy

1. Code family.

Nearly every code choice has been shown to be insecure

- GRS
- Reed-Muller
- Concatenated
- Elliptic
- ...

$\rightarrow$ Plain binary Goppa codes secure for 40 years.

2. Protocol.

Ideal use of PKC: exchange a key for symmetric cipher.

Overall Strategy

1. Code family.

Nearly every code choice has been shown to be insecure

- GRS
- Reed-Muller
- Concatenated
- Elliptic
- ...

$\rightarrow$ Plain binary Goppa codes secure for 40 years.

2. Protocol.

Ideal use of PKC: exchange a key for symmetric cipher.

$\rightarrow$ Focus on designing KEM.

Overall Strategy

1. Code family.

Nearly every code choice has been shown to be insecure

- GRS
- Reed-Muller
- Concatenated
- Elliptic
- ...

$\rightarrow$ Plain binary Goppa codes secure for 40 years.

2. Protocol.

Ideal use of PKC: exchange a key for symmetric cipher.

$\rightarrow$ Focus on designing KEM.

3. Framework.

Overall Strategy

1. Code family.

Nearly every code choice has been shown to be insecure

- GRS
- Reed-Muller
- Concatenated
- Elliptic
- ...

$\rightarrow$ Plain binary Goppa codes secure for 40 years.

2. Protocol.

Ideal use of PKC: exchange a key for symmetric cipher.

$\rightarrow$ Focus on designing KEM.

3. Framework.

Since we use a KEM, "plaintext" is randomly generated.

Overall Strategy

1. Code family.

Nearly every code choice has been shown to be insecure

- GRS
- Reed-Muller
- Concatenated
- Elliptic
- ...

$\rightarrow$ Plain binary Goppa codes secure for 40 years.

2. Protocol.

Ideal use of PKC: exchange a key for symmetric cipher.

$\rightarrow$ Focus on designing KEM.

3. Framework.

Since we use a KEM, “plaintext” is randomly generated.

$\rightarrow$ More practical to use Niederreiter.

Select hash functions $\mathbf{H}, \mathbf{K}$ (in practice, just use SHAKE-256).

Classic McEliece: a binary Goppa-based KEM

Select hash functions $\mathbf{H}, \mathbf{K}$ (in practice, just use SHAKE-256).

Key Generation

- Choose a Goppa code $\mathcal{C}$.
- SK: description $(g, \alpha_1, \ldots \alpha_n)$ for $\mathcal{C}$ plus random string s.
- PK: parity-check matrix H in systematic form for $\mathcal{C}$.

Classic McEliece: a binary Goppa-based KEM

Select hash functions $\mathbf{H}, \mathbf{K}$ (in practice, just use SHAKE-256).

Key Generation

- Choose a Goppa code $\mathcal{C}$.
- SK: description $(g, \alpha_1, \dots \alpha_n)$ for $\mathcal{C}$ plus random string s.
- PK: parity-check matrix H in systematic form for $\mathcal{C}$.

Encapsulation

- Sample a word $e \in \mathbb{F}_2^n$ of weight t.
- $c = (c_0, c_1)$ where $c_0 = He^T$, $c_1 = \mathbf{H}(e)$.
- $K = \mathbf{K}(c, e)$

Classic McEliece: a binary Goppa-based KEM

Select hash functions $\mathbf{H}, \mathbf{K}$ (in practice, just use SHAKE-256).

Key Generation

- Choose a Goppa code $\mathcal{C}$.
- SK: description $(g, \alpha_1, \ldots \alpha_n)$ for $\mathcal{C}$ plus random string s.
- PK: parity-check matrix H in systematic form for $\mathcal{C}$.

Encapsulation

- Sample a word $e \in \mathbb{F}_2^n$ of weight t.
- $c = (c_0, c_1)$ where $c_0 = He^T$, $c_1 = \mathbf{H}(e)$.
- $K = \mathbf{K}(c, e)$

Decryption

- Set $e' = Decode(c_0)$.
- $c' = (c'_0, c'_1)$ where $c'_0 = He'^T$, $c'_1 = \mathbf{H}(e')$.
- Return $K = \mathbf{K}(c', s)$ if decoding fails or $c \neq c'$.
- Else return $K = \mathbf{K}(c', e')$.

Alternative: NTS-KEM

Independent work, similar in spirit. Main differences:

ALTERNATIVE: NTS-KEM

Independent work, similar in spirit. Main differences:

1. Monic Squarefree Goppa poly (vs Irreducible).

ALTERNATIVE: NTS-KEM

Independent work, similar in spirit. Main differences:

1. Monic Squarefree Goppa poly (vs Irreducible).

No significant advantage either way, but irreducible is more “conservative”.

Alternative: NTS-KEM

Independent work, similar in spirit. Main differences:

1. Monic Squarefree Goppa poly (vs Irreducible).

No significant advantage either way, but irreducible is more "conservative".

2. Permuted systematic form during key generation (vs Unpermuted).

ALTERNATIVE: NTS-KEM

Independent work, similar in spirit. Main differences:

1. Monic Squarefree Goppa poly (vs Irreducible).

No significant advantage either way, but irreducible is more "conservative".

2. Permuted systematic form during key generation (vs Unpermuted).

100% success chance (vs 29%) but not constant time (would be slower).

Alternative: NTS-KEM

Independent work, similar in spirit. Main differences:

1. Monic Squarefree Goppa poly (vs Irreducible).

No significant advantage either way, but irreducible is more "conservative".

2. Permuted systematic form during key generation (vs Unpermuted).

100% success chance (vs 29%) but not constant time (would be slower).

Also, expanding seed for private key is more expensive.

Alternative: NTS-KEM

Independent work, similar in spirit. Main differences:

1. Monic Squarefree Goppa poly (vs Irreducible).

No significant advantage either way, but irreducible is more "conservative".

2. Permuted systematic form during key generation (vs Unpermuted).

100% success chance (vs 29%) but not constant time (would be slower).

Also, expanding seed for private key is more expensive.

3. Obfuscated ciphertext (vs traditional $(He^T, \mathbf{H}(e))$).

Alternative: NTS-KEM

Independent work, similar in spirit. Main differences:

1. Monic Squarefree Goppa poly (vs Irreducible).

No significant advantage either way, but irreducible is more "conservative".

2. Permuted systematic form during key generation (vs Unpermuted).

100% success chance (vs 29%) but not constant time (would be slower).

Also, expanding seed for private key is more expensive.

3. Obfuscated ciphertext (vs traditional $(He^T, \mathbf{H}(e))$).

Same length, more complicated description, no advantages.

ALTERNATIVE: NTS-KEM

Independent work, similar in spirit. Main differences:

1. Monic Squarefree Goppa poly (vs Irreducible).

No significant advantage either way, but irreducible is more "conservative".

2. Permuted systematic form during key generation (vs Unpermuted).

100% success chance (vs 29%) but not constant time (would be slower).

Also, expanding seed for private key is more expensive.

3. Obfuscated ciphertext (vs traditional $(He^T, \mathbf{H}(e))$).

Same length, more complicated description, no advantages.

In fact, obfuscated ciphertext is equivalent to traditional.

Obfuscating ciphertexts

Consider public matrix M, i.e. $H = (I_k|M)$.

Obfuscating ciphertexts

Consider public matrix M, i.e. $H = (I_k|M)$.

Generate $e = (e_c, e_a, e_b)$ of size $(n-k) + (k-256) + 256$ bits.

Obfuscating ciphertexts

Consider public matrix M, i.e. $H = (I_k|M)$.

Generate $e = (e_c, e_a, e_b)$ of size $(n-k)+(k-256)+256$ bits.

$c = (c_0, c_1)$ where $c_0 = e_c + M(e_a, \mathbf{H}(e))^T$, $c_1 = \mathbf{H}(e) + e_b$.

OBFUSCATING CIPHERTEXTS

Consider public matrix M, i.e. $H = (I_k|M)$.

Generate $e = (e_c, e_a, e_b)$ of size $(n-k)+(k-256)+256$ bits.

$c = (c_0, c_1)$ where $c_0 = e_c + M(e_a, \mathbf{H}(e))^T$, $c_1 = \mathbf{H}(e) + e_b$.

Generate e as usual and call e_c, e_a, e_b as above.

Obfuscating ciphertexts

Consider public matrix M, i.e. $H = (I_k|M)$.

Generate $e = (e_c, e_a, e_b)$ of size $(n-k)+(k-256)+256$ bits.

$c = (c_0, c_1)$ where $c_0 = e_c + M(e_a, \mathbf{H}(e))^T$, $c_1 = \mathbf{H}(e) + e_b$.

Generate e as usual and call e_c, e_a, e_b as above.

$c = (c_0, c_1)$ where $c_0 = e_c + M(e_a, e_b)^T$, $c_1 = \mathbf{H}(e)$.

Consider public matrix M, i.e. $H = (I_k|M)$.

Generate $e = (e_c, e_a, e_b)$ of size $(n-k)+(k-256)+256$ bits.

$c = (c_0, c_1)$ where $c_0 = e_c + M(e_a, \mathbf{H}(e))^T$, $c_1 = \mathbf{H}(e) + e_b$.

Generate e as usual and call e_c, e_a, e_b as above.

$c = (c_0, c_1)$ where $c_0 = e_c + M(e_a, e_b)^T$, $c_1 = \mathbf{H}(e)$.

Tweak c_1 by adding e_b: still uniform random hash.

Consider public matrix M, i.e. $H = (I_k|M)$.

Generate $e = (e_c, e_a, e_b)$ of size $(n-k) + (k-256) + 256$ bits.

$c = (c_0, c_1)$ where $c_0 = e_c + M(e_a, \mathbf{H}(e))^T$, $c_1 = \mathbf{H}(e) + e_b$.

Generate e as usual and call e_c, e_a, e_b as above.

$c = (c_0, c_1)$ where $c_0 = e_c + M(e_a, e_b)^T$, $c_1 = \mathbf{H}(e)$.

Tweak c_1 by adding e_b: still uniform random hash.

$c = (c_0, c_1)$ where $c_0 = e_c + M(e_a, e_b)^T$, $c_1 = \mathbf{H}(e) + e_b$.

Consider public matrix M, i.e. $H = (I_k|M)$.

Generate $e = (e_c, e_a, e_b)$ of size $(n-k)+(k-256)+256$ bits.

$c = (c_0, c_1)$ where $c_0 = e_c + M(e_a, \mathbf{H}(e))^T$, $c_1 = \mathbf{H}(e) + e_b$.

Generate e as usual and call e_c, e_a, e_b as above.

$c = (c_0, c_1)$ where $c_0 = e_c + M(e_a, e_b)^T$, $c_1 = \mathbf{H}(e)$.

Tweak c_1 by adding e_b: still uniform random hash.

$c = (c_0, c_1)$ where $c_0 = e_c + M(e_a, e_b)^T$, $c_1 = \mathbf{H}(e) + e_b$.

Define public function $Obfuscate(A, B) = (A + M(0, B)^T, B)$.

Obfuscating ciphertexts

Consider public matrix M, i.e. $H = (I_k|M)$.

Generate $e = (e_c, e_a, e_b)$ of size $(n-k) + (k-256) + 256$ bits.

$c = (c_0, c_1)$ where $c_0 = e_c + M(e_a, \mathbf{H}(e))^T$, $c_1 = \mathbf{H}(e) + e_b$.

Generate e as usual and call e_c, e_a, e_b as above.

$c = (c_0, c_1)$ where $c_0 = e_c + M(e_a, e_b)^T$, $c_1 = \mathbf{H}(e)$.

Tweak c_1 by adding e_b: still uniform random hash.

$c = (c_0, c_1)$ where $c_0 = e_c + M(e_a, e_b)^T$, $c_1 = \mathbf{H}(e) + e_b$.

Define public function $Obfuscate(A, B) = (A + M(0, B)^T, B)$.

Then $Obfuscate(c_0, c_1)$ is an NTS-KEM ciphertext.

Block Length

NTS-KEM requires $n = 2^m$, not true for Classic McEliece.

Block Length

NTS-KEM requires $n = 2^m$, not true for Classic McEliece.

$n < 2^m$ little extra implementation effort, but allows more flexibility.

Block Length

NTS-KEM requires $n = 2^m$, not true for Classic McEliece.

$n < 2^m$ little extra implementation effort, but allows more flexibility.

Possibility of tradeoff with data sizes.

Block Length

NTS-KEM requires $n = 2^m$, not true for Classic McEliece.

$n < 2^m$ little extra implementation effort, but allows more flexibility.

Possibility of tradeoff with data sizes.

NTS-KEM parameters (bytes):

m	n	t	PK Size	SK Size	Ciph Size	Security
13	8,192	136	1,419,704	19,890	253	5
13	8,192	80	929,760	17,524	162	3
12	4,096	64	319,488	9,216	128	1

Block Length

NTS-KEM requires $n = 2^m$, not true for Classic McEliece.

$n < 2^m$ little extra implementation effort, but allows more flexibility.

Possibility of tradeoff with data sizes.

NTS-KEM parameters (bytes):

m	n	t	PK Size	SK Size	Ciph Size	Security
13	8,192	136	1,419,704	19,890	253	5
13	8,192	80	929,760	17,524	162	3
12	4,096	64	319,488	9,216	128	1

Classic McEliece parameters (bytes):

m	n	t	PK Size	SK Size	Ciph Size	Security
13	8,192	128	1,357,824	14,080	240	5
13	6,960	119	1,046,739	13,908	226	5
13	6,688	128	1,044,992	13,892	240	5
13	4,608	96	524,160	13,568	188	3
12	3,488	64	261,120	6,452	128	1

Part IV

Final Considerations

Inherent Aspects of Conservative CBC

Very simple description (binary objects, low-weight XOR).

Inherent Aspects of Conservative CBC

Very simple description (binary objects, low-weight XOR).

Very fast implementation (encapsulation/decapsulation).

INHERENT ASPECTS OF CONSERVATIVE CBC

Very simple description (binary objects, low-weight XOR).

Very fast implementation (encapsulation/decapsulation).

Very small ciphertext size.

Inherent Aspects of Conservative CBC

Very simple description (binary objects, low-weight XOR).

Very fast implementation (encapsulation/decapsulation).

Very small ciphertext size.

No decryption failures.

Inherent Aspects of Conservative CBC

Very simple description (binary objects, low-weight XOR).

Very fast implementation (encapsulation/decapsulation).

Very small ciphertext size.

No decryption failures.

Long-term static keys + easy, tight IND-CCA reduction
(Bernstein, P., 2018).

Very simple description (binary objects, low-weight XOR).

Very fast implementation (encapsulation/decapsulation).

Very small ciphertext size.

No decryption failures.

Long-term static keys + easy, tight IND-CCA reduction
(Bernstein, P., 2018).

40 years of security history.

INHERENT ASPECTS OF CONSERVATIVE CBC

Very simple description (binary objects, low-weight XOR).

Very fast implementation (encapsulation/decapsulation).

Very small ciphertext size.

No decryption failures.

Long-term static keys + easy, tight IND-CCA reduction
(Bernstein, P., 2018).

40 years of security history.

Very large key and slow key generation.

What About Signatures?

WHAT ABOUT SIGNATURES?

Long time standing open problem.

What About Signatures?

Long time standing open problem.

4 NIST submissions, 0 survivors: all withdrawn/broken.

What About Signatures?

Long time standing open problem.

4 NIST submissions, 0 survivors: all withdrawn/broken.

Inherent problem with metric.

What About Signatures?

Long time standing open problem.

4 NIST submissions, 0 survivors: all withdrawn/broken.

Inherent problem with metric.

Hash-and-sign: disjoint “balls” don’t cover space.

What About Signatures?

Long time standing open problem.

4 NIST submissions, 0 survivors: all withdrawn/broken.

Inherent problem with metric.

Hash-and-sign: disjoint “balls” don’t cover space.

Fiat-Shamir: “sparse” masking vector doesn’t hide secret.

WHAT ABOUT SIGNATURES?

Long time standing open problem.

4 NIST submissions, 0 survivors: all withdrawn/broken.

Inherent problem with metric.

Hash-and-sign: disjoint “balls” don’t cover space.

Fiat-Shamir: “sparse” masking vector doesn’t hide secret.

Out of scope of these talks (but happy to discuss!).

See you tomorrow!