Post-Quantum

Cryptography Conference

Code-based Cryptography

Simona Samardjiska

Assistant Professor at Digital Security Group, ICIS, Radboud University

Code-based cryptography

Simona Samardjiska
Digital Security Group, Radboud University
simonas@cs.ru.nl

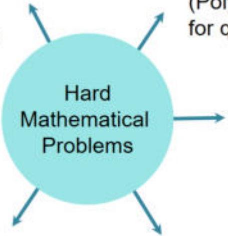
Post-Quantum Cryptography Conference, November 07, 2023

What is code-based cryptography?

Code-based cryptography - an area of post-quantum cryptography

Isogeny based cryptosystems –
KEMs/NIKEs/signatures
(Finding isogenies on supersingular elliptic curves)

Multivariate Quadratic cryptosystems – mainly signatures
(Polynomial System Solving –PoSSo, for quadratic polynomials - MQ problem)

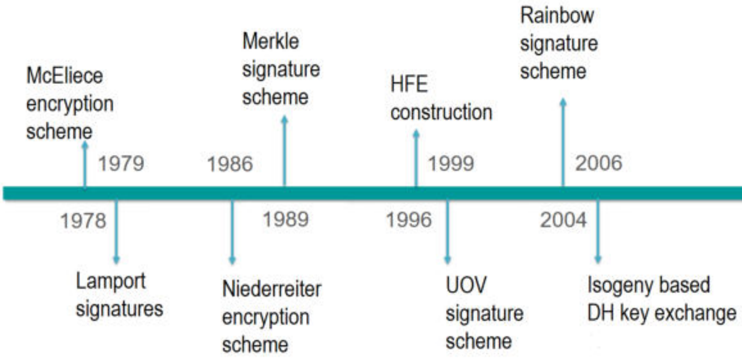


Code-based cryptosystems – mainly encryption/KEMs
(decoding random linear codes, equivalence)

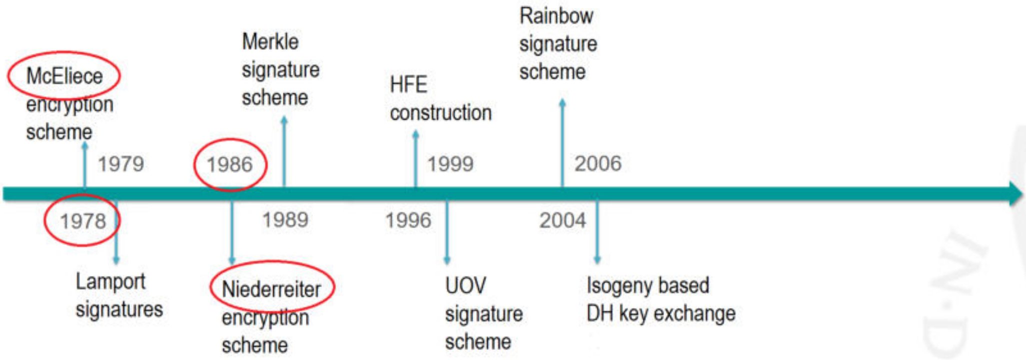
Hash-based signatures (only)
(only secure hash function needed)

Lattice-based cryptosystems – signatures/encryption/KEMs
(many different hard problems – SIS, SVP, LWE)

Yet–not new at all!

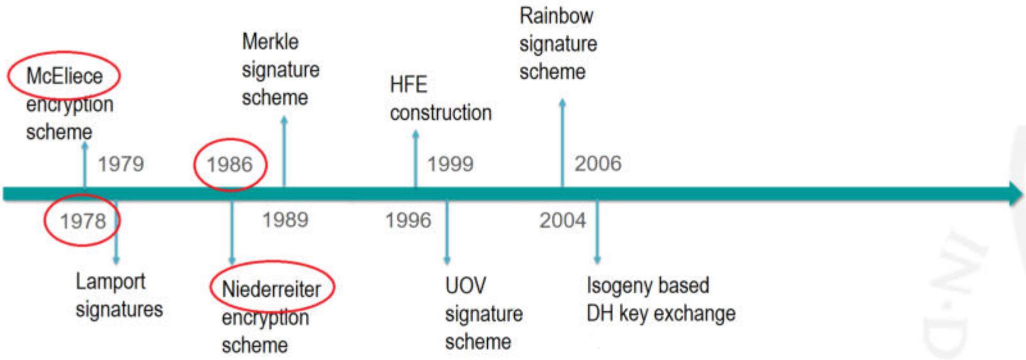


Yet–not new at all!



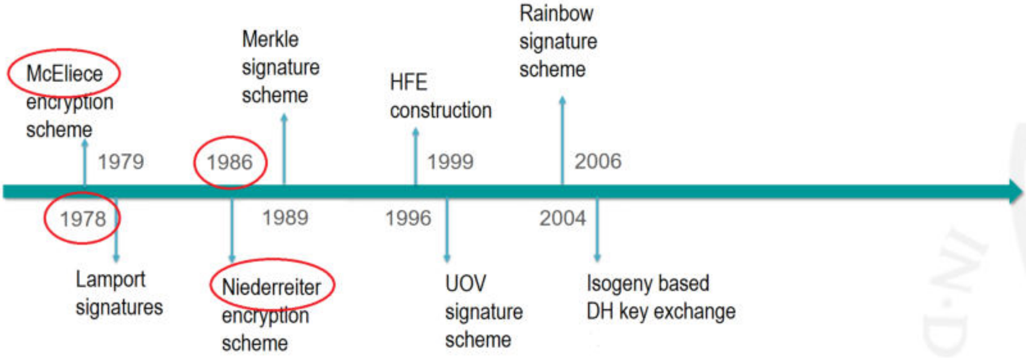
- McEliece cryptosystem is as old as RSA! . . .

Yet–not new at all!



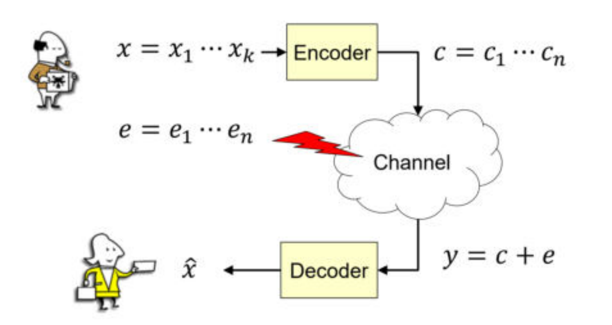
- McEliece cryptosystem is as old as RSA! . . .
- But, huge public key $\approx 70KB$ for 80 bits security

Yet–not new at all!



- McEliece cryptosystem is as old as RSA! ...
- But, huge public key $\approx 70KB$ for 80 bits security
- Classic McEliece $\approx 260KB$ for NIST level 1 security

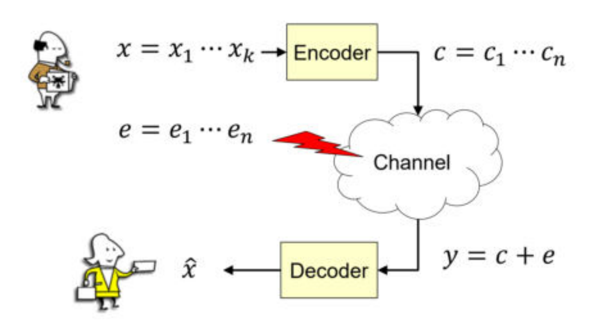
Basic principle – borrowed from noisy channel communication



Noisy channels introduce **errors** to transmitted digital messages

- satellite to earth communication, mobile phone data, humans typing on keyboard, ...

Basic principle – borrowed from noisy channel communication

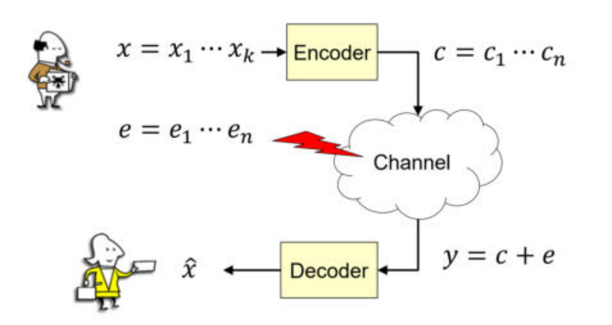


Noisy channels introduce **errors** to transmitted digital messages

- satellite to earth communication, mobile phone data, humans typing on keyboard, ...

Systems need a mechanism for **correction of errors** – They use **error correcting codes**

Basic principle – borrowed from noisy channel communication



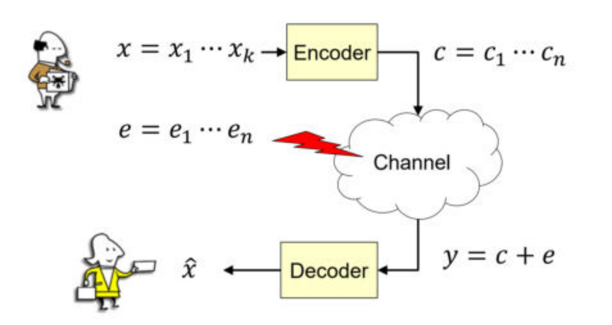
Noisy channels introduce **errors** to transmitted digital messages

- satellite to earth communication, mobile phone data, humans typing on keyboard, ...

Systems need a mechanism for **correction of errors** – They use **error correcting codes**

- Sender encodes the message by adding some redundancy

Basic principle – borrowed from noisy channel communication



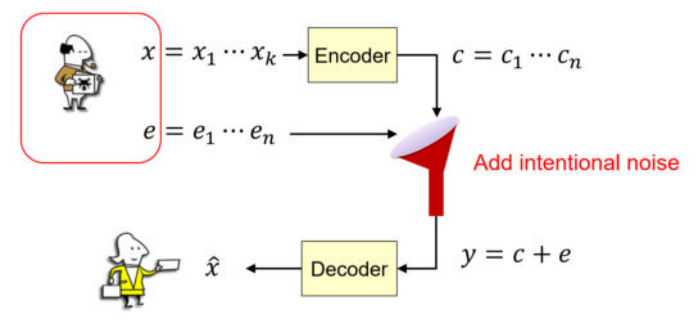
Noisy channels introduce **errors** to transmitted digital messages

- satellite to earth communication, mobile phone data, humans typing on keyboard, . . .

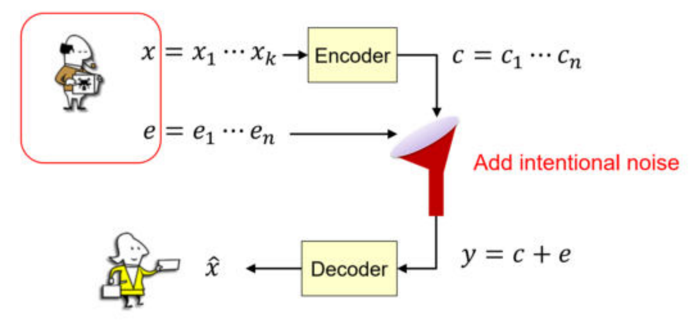
Systems need a mechanism for **correction of errors** – They use **error correcting codes**

- Sender encodes the message by adding some redundancy
- Receiver decodes the message to remove the error

How to use it in cryptography?

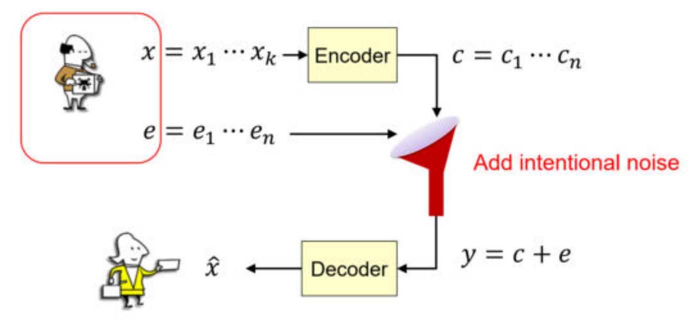


How to use it in cryptography?

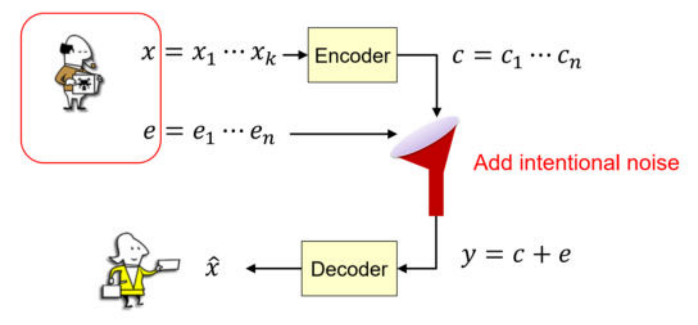


- Sender encodes the message by adding some redundancy + introduces error (**intentional noise**)
- Receiver decodes the message to remove the error with knowledge of efficient decoding method
- For some codes, we know how to correct errors efficiently, **bot not for all!**

How to use it in cryptography?



- Sender encodes the message by adding some redundancy + introduces error (**intentional noise**)
- Receiver decodes the message to remove the error with knowledge of efficient decoding method
- For some codes, we know how to correct errors efficiently, **bot not for all!**
- Decoding random codes is a **hard problem** $\Rightarrow$ we can use codes **in cryptography**!



- Sender encodes the message by adding some redundancy + introduces error (**intentional noise**)
- Receiver decodes the message to remove the error with knowledge of efficient decoding method
- For some codes, we know how to correct errors efficiently, **bot not for all!**
- Decoding random codes is a **hard problem** $\Rightarrow$ we can use codes **in cryptography**!

Binary $[n, k]$ linear code $\mathcal{C}$ of length n and dimension k is a subspace of $\mathbb{F}_2^n$ of dimension k

Linear codes basics

Binary $[n, k]$ linear code $\mathcal{C}$ of length n and dimension k is a subspace of $\mathbb{F}_2^n$ of dimension k

- Defined by basis matrix – $k \times n$ **generator matrix G**

$$\mathcal{C} = \{\mathbf{c} \mid \mathbf{c} = \mathbf{mG}, \mathbf{m} \in \mathbb{F}_2^k\}$$

Linear codes basics

Binary $[n, k]$ linear code $\mathcal{C}$ of length n and dimension k is a subspace of $\mathbb{F}_2^n$ of dimension k

- Defined by basis matrix – $k \times n$ **generator matrix G**

$$\mathcal{C} = \{\mathbf{c} \mid \mathbf{c} = \mathbf{m}\mathbf{G}, \mathbf{m} \in \mathbb{F}_2^k\}$$

- or by kernel matrix – $(n-k) \times n$ **Parity-Check matrix H**

$$\mathcal{C} = \{\mathbf{c} \mid \mathbf{c}\mathbf{H}^\top = \mathbf{0}\}$$

Linear codes basics

Binary $[n, k]$ linear code $\mathcal{C}$ of length n and dimension k is a subspace of $\mathbb{F}_2^n$ of dimension k

- Defined by basis matrix – $k \times n$ **generator matrix G**

$$\mathcal{C} = \{\mathbf{c} \mid \mathbf{c} = \mathbf{m}\mathbf{G}, \mathbf{m} \in \mathbb{F}_2^k\}$$

- or by kernel matrix – $(n-k) \times n$ **Parity-Check matrix H**

$$\mathcal{C} = \{\mathbf{c} \mid \mathbf{c}\mathbf{H}^\top = \mathbf{0}\}$$

- $\mathbf{G}\mathbf{H}^\top = \mathbf{0}$

Linear codes basics

Binary $[n, k]$ linear code $\mathcal{C}$ of length n and dimension k is a subspace of $\mathbb{F}_2^n$ of dimension k

- Defined by basis matrix – $k \times n$ **generator matrix G**

$$\mathcal{C} = \{\mathbf{c} \mid \mathbf{c} = \mathbf{mG}, \mathbf{m} \in \mathbb{F}_2^k\}$$

- or by kernel matrix – $(n - k) \times n$ **Parity-Check matrix H**

$$\mathcal{C} = \{\mathbf{c} \mid \mathbf{cH}^\top = \mathbf{0}\}$$

- $\mathbf{GH}^\top = \mathbf{0}$
- Systematic form of $\mathbf{G} = [\mathbf{I}_{k\times k}|\mathbf{T}] \quad \Rightarrow \quad \mathbf{H} = [\mathbf{T}^\top|\mathbf{I}_{(n-k)\times(n-k)}]$ (store only the redundant part $\mathbf{T}$)

Linear codes basics

Binary $[n, k]$ linear code $\mathcal{C}$ of length n and dimension k is a subspace of $\mathbb{F}_2^n$ of dimension k

- Defined by basis matrix – $k \times n$ **generator matrix G**

$$\mathcal{C} = \{\mathbf{c} \mid \mathbf{c} = \mathbf{mG}, \mathbf{m} \in \mathbb{F}_2^k\}$$

- or by kernel matrix – $(n-k) \times n$ **Parity-Check matrix H**

$$\mathcal{C} = \{\mathbf{c} \mid \mathbf{cH}^\top = \mathbf{0}\}$$

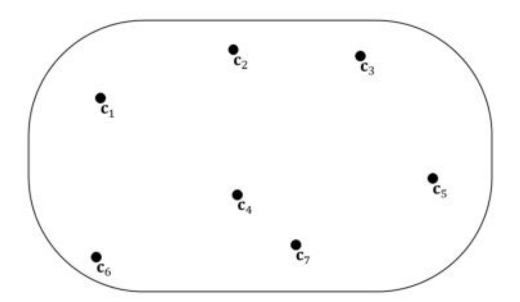
- $\mathbf{GH}^\top = \mathbf{0}$
- Systematic form of $\mathbf{G} = [\mathbf{I}_{k \times k} | \mathbf{T}] \quad \Rightarrow \quad \mathbf{H} = [\mathbf{T}^\top | \mathbf{I}_{(n-k)\times(n-k)}]$ (store only the redundant part $\mathbf{T}$)
- Elements of $\mathcal{C}$ are called **codewords** – notation $\mathbf{c}_1, \mathbf{c}_2, \ldots$

Linear codes + Hamming distance basics

- **Hamming weight** of $\mathbf{c}$ – number of non-zero coordinates of $\mathbf{c}$ - notation $hw(\mathbf{c})$

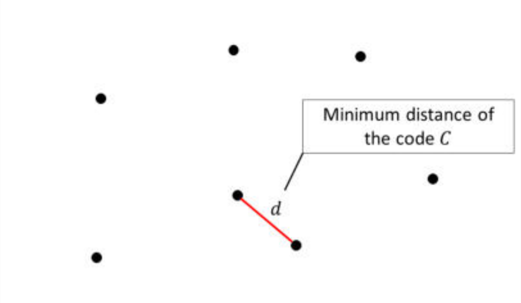
Linear codes + Hamming distance basics

- **Hamming weight** of $\mathbf{c}$ – number of non-zero coordinates of $\mathbf{c}$ - notation $hw(\mathbf{c})$

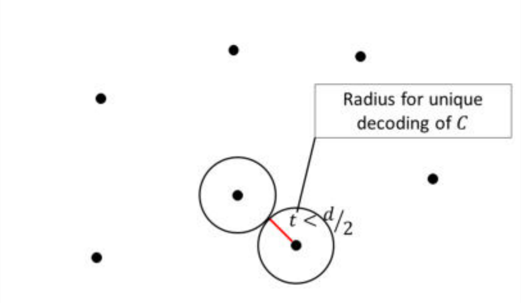


Linear codes + Hamming distance basics

- **Hamming weight** of $\mathbf{c}$ – number of non-zero coordinates of $\mathbf{c}$ - notation $hw(\mathbf{c})$
- **Minimum weight** $d(\mathcal{C}) = \min_{\mathbf{c} \neq \mathbf{0}}\{hw(\mathbf{c})\}$



- **Hamming weight** of $\mathbf{c}$ – number of non-zero coordinates of $\mathbf{c}$ - notation $hw(\mathbf{c})$
- **Minimum weight** $d(\mathcal{C}) = \min_{\mathbf{c} \neq \mathbf{0}} \{hw(\mathbf{c})\}$
- If $d(\mathcal{C}) > 2t$ - the code can correct t errors (t bit-flips during transmission)



Linear codes basics

Encoding of messages:

$$\mathbf{c} = \mathbf{m}\mathbf{G}$$

Transmission errors introduced:

$$\mathbf{y} = \mathbf{c} + \mathbf{e}$$

Decoding: A procedure *Decode*():

$$\text{Find } \mathbf{c}' \text{ s.t. } hw(\mathbf{c}' - \mathbf{y}) \leq t$$

Linear codes basics

Encoding of messages:

$$\mathbf{c} = \mathbf{mG}$$

Transmission errors introduced:

$$\mathbf{y} = \mathbf{c} + \mathbf{e}$$

Decoding: A procedure *Decode*():

$$\text{Find } \mathbf{c}' \text{ s.t. } hw(\mathbf{c}' - \mathbf{y}) \leq t$$

Another (equivalent) way of looking at it:

Syndrome Decoding:

Given **syndrome** $\mathbf{s} = \mathbf{yH}^\top$, find $\mathbf{e}$ of weight at most t such that $\mathbf{s} = \mathbf{eH}^\top$

Encoding of messages:

$$\mathbf{c} = \mathbf{mG}$$

Transmission errors introduced:

$$\mathbf{y} = \mathbf{c} + \mathbf{e}$$

Decoding: A procedure *Decode*():

$$\text{Find } \mathbf{c}' \text{ s.t. } hw(\mathbf{c}' - \mathbf{y}) \leq t$$

Another (equivalent) way of looking at it:

Syndrome Decoding:

Given **syndrome** $\mathbf{s} = \mathbf{yH}^\top$, find $\mathbf{e}$ of weight at most t such that $\mathbf{s} = \mathbf{eH}^\top$

- **Syndrome decoding, equivalently, decoding of random codes is NP-hard**

McEliece and Niederreiter cryptosystems

Linear codes for cryptography

- Idea for cryptographic use: **Scramble efficiently decodable codes to hide structure**

Linear codes for cryptography

- Idea for cryptographic use: **Scramble efficiently decodable codes to hide structure**
- Legitimate user has trapdoor – private key + efficient decoder $\Rightarrow$ can decode

Linear codes for cryptography

- Idea for cryptographic use: **Scramble efficiently decodable codes to hide structure**
- Legitimate user has trapdoor – private key + efficient decoder $\Rightarrow$ can decode
- Adversary without trapdoor is faced with a random code and generic, inefficient decoding

Linear codes for cryptography

- Idea for cryptographic use: **Scramble efficiently decodable codes to hide structure**
- Legitimate user has trapdoor – private key + efficient decoder $\Rightarrow$ can decode
- Adversary without trapdoor is faced with a random code and generic, inefficient decoding
- Instantiations:
  - McEliece 1978
    - irreducible binary Goppa codes - still used today!
    - everything else - broken!
    - $n = 1024$, $k = 524$, $t = 50$
    - public key size: 536576 bits, ciphertext size: 1024 bits
    - today, security of 60 bits

Linear codes for cryptography

- Idea for cryptographic use: **Scramble efficiently decodable codes to hide structure**
- Legitimate user has trapdoor – private key + efficient decoder $\Rightarrow$ can decode
- Adversary without trapdoor is faced with a random code and generic, inefficient decoding
- Instantiations:
  - McEliece 1978
    - irreducible binary Goppa codes - still used today!
    - everything else - broken!
    - $n = 1024,\ k = 524,\ t = 50$
    - public key size: 536576 bits, ciphertext size: 1024 bits
    - today, security of 60 bits
  - Niederreiter 1986
    - Reed-Solomon codes - broken 1992 by Sidelnikov & Chestakov

Linear codes for cryptography

- Idea for cryptographic use: **Scramble efficiently decodable codes to hide structure**
- Legitimate user has trapdoor – private key + efficient decoder $\Rightarrow$ can decode
- Adversary without trapdoor is faced with a random code and generic, inefficient decoding
- Instantiations:
  - McEliece 1978
    - irreducible binary Goppa codes - still used today!
    - everything else - broken!
    - $n = 1024, \ k = 524, \ t = 50$
    - public key size: 536576 bits, ciphertext size: 1024 bits
    - today, security of 60 bits
  - Niederreiter 1986
    - Reed-Solomon codes - broken 1992 by Sidelnikov & Chestakov
  - **McEliece and Niederreiter constructions are equivalent!**

Key generation

Private key: generator matrix $\mathbf{G}'$
invertible matrix $\mathbf{S}$
permutation matrix $\mathbf{P}$

Public key: $\mathbf{G} = \mathbf{S}\mathbf{G}'\mathbf{P}$

Key generation

Private key: generator matrix $\mathbf{G}'$
invertible matrix $\mathbf{S}$
permutation matrix $\mathbf{P}$

Public key: $\mathbf{G} = \mathbf{S}\mathbf{G}'\mathbf{P}$

Encryption of message m

Generate random error $\mathbf{e}$ of weight-t. Compute ciphertext:

$$\mathbf{y} = \mathbf{m}\mathbf{G} + \mathbf{e}$$

Key generation

Private key: generator matrix $\mathbf{G}'$
invertible matrix $\mathbf{S}$
permutation matrix $\mathbf{P}$

Public key: $\mathbf{G} = \mathbf{S}\mathbf{G}'\mathbf{P}$

Encryption of message m

Generate random error $\mathbf{e}$ of weight-t. Compute ciphertext:

$$\mathbf{y} = \mathbf{m}\mathbf{G} + \mathbf{e}$$

Decryption of ciphertext y

Compute $\mathbf{y}' = \mathbf{y}\mathbf{P}^{-1}$ and

$$\mathbf{x} = DecodeG(\mathbf{y}')$$

Compute $\mathbf{m} = \mathbf{x}\mathbf{S}^{-1}$.

Key generation

Private key: parity-check matrix $\mathbf{H}'$
invertible matrix $\mathbf{S}$
permutation matrix $\mathbf{P}$

Public key: $\mathbf{H} = \mathbf{S}\mathbf{H}'\mathbf{P}$

McEliece Cryptosystem - 1978 & Niederreiter Cryptosystem - 1986

Key generation

Private key: generator matrix $\mathbf{G}'$
invertible matrix $\mathbf{S}$
permutation matrix $\mathbf{P}$

Public key: $\mathbf{G} = \mathbf{S}\mathbf{G}'\mathbf{P}$

Encryption of message m

Generate random error $\mathbf{e}$ of weight-t. Compute ciphertext:

$$\mathbf{y} = \mathbf{m}\mathbf{G} + \mathbf{e}$$

Decryption of ciphertext y

Compute $\mathbf{y}' = \mathbf{y}\mathbf{P}^{-1}$ and

$$\mathbf{x} = DecodeG(\mathbf{y}')$$

Compute $\mathbf{m} = \mathbf{x}\mathbf{S}^{-1}$.

Key generation

Private key: parity-check matrix $\mathbf{H}'$
invertible matrix $\mathbf{S}$
permutation matrix $\mathbf{P}$

Public key: $\mathbf{H} = \mathbf{S}\mathbf{H}'\mathbf{P}$

Encryption of message m

Transform $\mathbf{m}$ into weight-t error $\mathbf{e}$. Compute ciphertext:

$$\mathbf{y} = \mathbf{e}\mathbf{H}^{\top}$$

McEliece Cryptosystem - 1978 & Niederreiter Cryptosystem - 1986

Key generation

Private key: generator matrix $\mathbf{G}'$
invertible matrix $\mathbf{S}$
permutation matrix $\mathbf{P}$

Public key: $\mathbf{G} = \mathbf{S}\mathbf{G}'\mathbf{P}$

Encryption of message m

Generate random error $\mathbf{e}$ of weight-t. Compute ciphertext:

$$\mathbf{y} = \mathbf{m}\mathbf{G} + \mathbf{e}$$

Decryption of ciphertext y

Compute $\mathbf{y}' = \mathbf{y}\mathbf{P}^{-1}$ and

$$\mathbf{x} = \mathit{DecodeG}(\mathbf{y}')$$

Compute $\mathbf{m} = \mathbf{x}\mathbf{S}^{-1}$.

Key generation

Private key: parity-check matrix $\mathbf{H}'$
invertible matrix $\mathbf{S}$
permutation matrix $\mathbf{P}$

Public key: $\mathbf{H} = \mathbf{S}\mathbf{H}'\mathbf{P}$

Encryption of message m

Transform $\mathbf{m}$ into weight-t error $\mathbf{e}$. Compute ciphertext:

$$\mathbf{y} = \mathbf{e}\mathbf{H}^\top$$

Decryption of ciphertext y

Compute $\mathbf{y}' = \mathbf{y}(\mathbf{S}^\top)^{-1}$, (syndrome) decode $\mathbf{y}'$

$$\mathbf{y}' = \mathbf{e}'\mathbf{H}'^\top$$

Compute $\mathbf{e} = \mathbf{e}'(\mathbf{P}^\top)^{-1}$.

Variety of code-based cryptosystems

- **Variety of constructions**
  - McEliece and Niederreiter encryption schemes (NIST finalist: Classic McEliece '17)
  - Alekhnovich '03 encryption [Alekhnovich '03]
  - CFS signature[Courtois, Finiasz & Sendrier '01]
  - Fiat-Shamir signatures [Stern '93; Veron '95; Cayrel, Gaborit & Girault '07]
  - Quasi - cyclic schemes (HQC '17)

Variety of code-based cryptosystems

- **Variety of constructions**
  - McEliece and Niederreiter encryption schemes (NIST finalist: Classic McEliece '17)
  - Alekhnovich '03 encryption [Alekhnovich '03]
  - CFS signature[Courtois, Finiasz & Sendrier '01]
  - Fiat-Shamir signatures [Stern '93; Veron '95; Cayrel, Gaborit & Girault '07]
  - Quasi - cyclic schemes (HQC '17)
- **Variety of metrics**
  - Hamming metric
  - Rank metric
  - Lee metric

Variety of code-based cryptosystems

- **Variety of constructions**
  - McEliece and Niederreiter encryption schemes (NIST finalist: Classic McEliece '17)
  - Alekhnovich '03 encryption [Alekhnovich '03]
  - CFS signature[Courtois, Finiasz & Sendrier '01]
  - Fiat-Shamir signatures [Stern '93; Veron '95; Cayrel, Gaborit & Girault '07]
  - Quasi - cyclic schemes (HQC '17)
- **Variety of metrics**
  - Hamming metric
  - Rank metric
  - Lee metric
- **Variety of codes**
  - Goppa codes
  - LDPC, MDPC (NIST finalist: BIKE '17) and LRPC
  - Reed-Solomon codes and Gabidulin codes

NIST code-based KEMs

NIST finalists

- July 22nd, 2020 - 3rd round NIST Finalists and Alternates announced
- 4 KEM finalists (5 alternates) - **3 code-based in Hamming metric**
- 3 signature finalists (3 alternates) - **No code-based**
- Decision based mostly on security considerations!
- NIST: *Performance wasn't the primary factor in our decisions, but we stayed aware of it*

Encryption/KEMs			
Crystals-Kyber	Lattice	MLWE	
Saber	Lattice	MLWR	
FrodoKEM	Lattice	LWE	
~~Round 5~~	~~Lattice~~	~~LWR/RLWR~~	
~~LAC~~	~~Lattice~~	~~RLWE~~	
~~NewHope~~	~~Lattice~~	~~RLWE~~	
~~Three Bears~~	~~Lattice~~	~~IMLWE~~	
NTRU	Lattice	NTRU	
NTRUprime	Lattice	NTRU	
SIKE	Isogeny	Isogeny	
Classic McEliece	Codes	Goppa	
~~NTS-KEM~~	~~Codes~~	~~Goppa~~	~~(merged)~~
BIKE	Codes	short Hamming	
HQC	Codes	short Hamming	
~~LEDAcrypt~~	~~Codes~~	~~short~~	
~~ROLLO~~	~~Codes~~	~~low rank~~	
~~RQC~~	~~Codes~~	~~low rank~~	

Signatures		
CRYSTALS-Dilithium	Lattice	Fiat-Shamir
~~qTesla~~	~~Lattice~~	~~Fiat-Shamir~~
Falcon	Lattice	Hash then sign
SPHINCS+	Symm	Hash
Picnic	Symm	ZKP
~~LUOV~~	~~MultVar~~	~~UOV~~
Rainbow	MultVar	UOV
GeMMS	MultVar	HFEv-
~~MQDSS~~	~~MultVar~~	~~Fiat-Shamir~~

NIST 4th round KEM candidates

Currently in the competition:

- **Classic McEliece**
    - Based on the Niederreiter cryptosystem with binary Goppa codes
    - Considered to be a conservative choice
    - No decoding failures

NIST 4th round KEM candidates

Currently in the competition:

- **Classic McEliece**
  - Based on the Niederreiter cryptosystem with binary Goppa codes
  - Considered to be a conservative choice
  - No decoding failures
- **BIKE**
  - Based on the Niederreiter cryptosystem with QC-MDPC (Quasi Cyclic Moderate-Density Parity-Check) codes
  - Bit-flipping decoding (now constant time)
  - Negligible decoding failure rate

NIST 4th round KEM candidates

Currently in the competition:

- **Classic McEliece**
  - Based on the Niederreiter cryptosystem with binary Goppa codes
  - Considered to be a conservative choice
  - No decoding failures
- **BIKE**
  - Based on the Niederreiter cryptosystem with QC-MDPC (Quasi Cyclic Moderate-Density Parity-Check) codes
  - Bit-flipping decoding (now constant time)
  - Negligible decoding failure rate
- **HQC**
  - Random Quasi Cyclic codes (BCH $\otimes$ repetition codes, now Read-Muller $\otimes$ Reed-Solomon)
  - BCH decoding, now RMRS
  - Negligible decoding failure rate

NIST 4th round KEM candidates comparisson

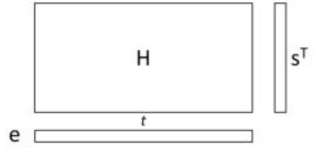
Algorithm	Security	pub.key(B)	priv.key(B)	ciphertxt	keygen/s	encaps/s	decaps/s
Classic McEliece348864	Level 1	261 120	6 492	128	7.99	69325.00	19486.00
Classic McEliece460896	Level 3	524 160	13 932	240	2.53	38832.67	7627.00
Classic McEliece6688128	Level 5	104 992	14 120	240	1.87	20083.00	6355.67
Classic McEliece6960119	Level 5	1 047 319	13 948	226	1.95	19673.67	6911.33
Classic McEliece8192128	Level 5	1 357 824	14 120	240	1.84	15075.33	6317.00
BIKE	Level 1	1 540	280	1 572	3944.00	22975.00	1154.33
BIKE	Level 3	3 082	418	3 114	1315.89	10289.33	509.83
BIKE	Level 5	5 122	580	5 154	586.33	5140.67	185.60
HQC-128	Level 1	2 249	40	4 481	24009.67	12494.67	6728.33
HQC-192	Level 3	4 522	40	9 026	10973.67	5644.67	3294.00
HQC-256	Level 5	7 245	40	14 469	5945.33	3055.33	1740.67
KYBER512	Level 1	800	1 632	768	93635.67	74457.67	107878.00
KYBER768	Level 3	1 184	2 400	1 088	60386.00	50918.67	68550.33
KYBER1024	Level 5	1 568	3 168	1 568	46629.33	38147.67	49443.33

Security of Code-based crypto – Information Set Decoding

Naive approach for syndrome decoding

Syndrome Decoding:

Given **syndrome** $\mathbf{s}$, find $\mathbf{e}$ of weight at most t such that $\mathbf{s} = \mathbf{e}\mathbf{H}^\top$

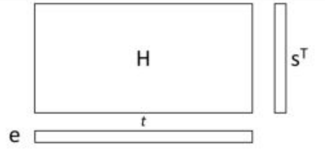


- $\mathbf{e}$ determines a linear combination of columns of $\mathbf{H}$ equal to $\mathbf{s}$!

Naive approach for syndrome decoding

Syndrome Decoding:

Given **syndrome** $\mathbf{s}$, find $\mathbf{e}$ of weight at most t such that $\mathbf{s} = \mathbf{e}\mathbf{H}^\top$

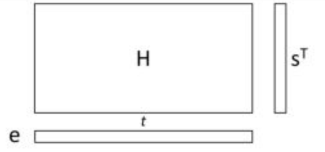


- $\mathbf{e}$ determines a linear combination of columns of $\mathbf{H}$ equal to $\mathbf{s}$!
- **Straightforward idea**: Try out all linear combinations of t columns of $\mathbf{H}$!

Naive approach for syndrome decoding

Syndrome Decoding:

Given **syndrome** $\mathbf{s}$, find $\mathbf{e}$ of weight at most t such that $\mathbf{s} = \mathbf{e}\mathbf{H}^\top$

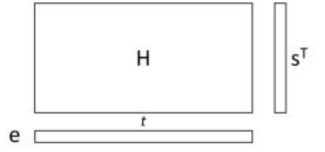


- $\mathbf{e}$ determines a linear combination of columns of $\mathbf{H}$ equal to $\mathbf{s}$!
- **Straightforward idea**: Try out all linear combinations of t columns of $\mathbf{H}$!
  - Guess and verify approach until correct linear combination is found

Naive approach for syndrome decoding

Syndrome Decoding:

Given **syndrome** $\mathbf{s}$, find $\mathbf{e}$ of weight at most t such that $\mathbf{s} = \mathbf{e}\mathbf{H}^\top$

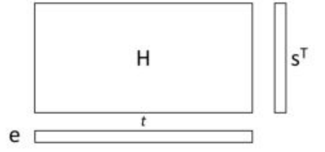


- $\mathbf{e}$ determines a linear combination of columns of $\mathbf{H}$ equal to $\mathbf{s}$!
- **Straightforward idea**: Try out all linear combinations of t columns of $\mathbf{H}$!
  - Guess and verify approach until correct linear combination is found
  - **Cost**: $\binom{n}{t}$ column operations

Naive approach for syndrome decoding

Syndrome Decoding:

Given **syndrome** $\mathbf{s}$, find $\mathbf{e}$ of weight at most t such that $\mathbf{s} = \mathbf{e}\mathbf{H}^\top$

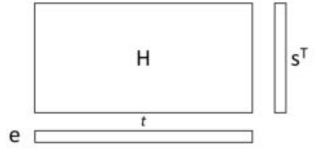


- $\mathbf{e}$ determines a linear combination of columns of $\mathbf{H}$ equal to $\mathbf{s}$!
- **Straightforward idea**: Try out all linear combinations of t columns of $\mathbf{H}$!
  - Guess and verify approach until correct linear combination is found
  - **Cost**: $\binom{n}{t}$ column operations
- We can do better using **Birthday paradox** $\approx \sqrt{\binom{n}{t}}$ column operations!

Naive approach for syndrome decoding

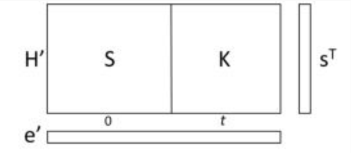
Syndrome Decoding:

Given **syndrome** $\mathbf{s}$, find $\mathbf{e}$ of weight at most t such that $\mathbf{s} = \mathbf{e}\mathbf{H}^\top$



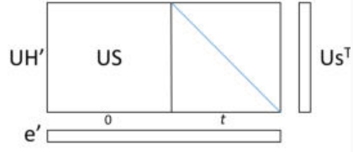
- $\mathbf{e}$ determines a linear combination of columns of $\mathbf{H}$ equal to $\mathbf{s}$!
- **Straightforward idea**: Try out all linear combinations of t columns of $\mathbf{H}$!
  - Guess and verify approach until correct linear combination is found
  - **Cost**: $\binom{n}{t}$ column operations
- We can do better using **Birthday paradox** $\approx \sqrt{\binom{n}{t}}$ column operations!
- Even better using **Information set decoding**!

Information set decoding [Prange '62]



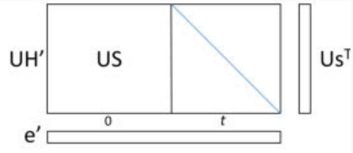
- Split **H** randomly in two parts **S** of k columns and **K** of $n - k$ columns, and hope that all positions of **S** are **error-free** (i.e. **S** is an **information set**)
  - I.e. $\mathbf{H}' = \mathbf{HP} = [\mathbf{S} \mid \mathbf{K}]$ (Set also $\mathbf{e}' = \mathbf{e}\mathbf{P}^\top$)

Information set decoding [Prange '62]



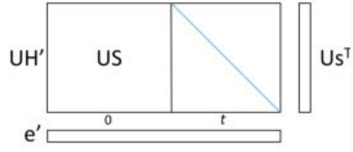
- Split $\mathbf{H}$ randomly in two parts $\mathbf{S}$ of k columns and $\mathbf{K}$ of $n-k$ columns, and hope that all positions of $\mathbf{S}$ are **error-free** (i.e. $\mathbf{S}$ is an **information set**)
  - I.e. $\mathbf{H}' = \mathbf{HP} = [\mathbf{S} \mid \mathbf{K}]$ (Set also $\mathbf{e}' = \mathbf{eP}^\top$)
  - Probability that guess is correct $\binom{k}{0}\binom{n-k}{t}/\binom{n}{t}$
- Compute $\mathbf{U}$ s.t. $\mathbf{K}$ is Gauss-reduced

Information set decoding [Prange '62]



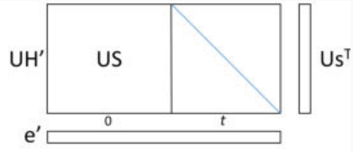
- Split $\mathbf{H}$ randomly in two parts $\mathbf{S}$ of k columns and $\mathbf{K}$ of $n-k$ columns, and hope that all positions of $\mathbf{S}$ are **error-free** (i.e. $\mathbf{S}$ is an **information set**)
  - I.e. $\mathbf{H}' = \mathbf{HP} = [\mathbf{S} \mid \mathbf{K}]$ (Set also $\mathbf{e}' = \mathbf{eP}^\top$)
  - Probability that guess is correct $\binom{k}{0}\binom{n-k}{t}/\binom{n}{t}$
- Compute $\mathbf{U}$ s.t. $\mathbf{K}$ is Gauss-reduced
- If guess is correct, $\mathbf{sU}^\top$ has weight t

Information set decoding [Prange '62]



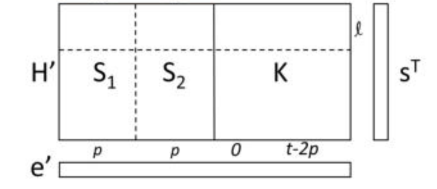
- Split $\mathbf{H}$ randomly in two parts $\mathbf{S}$ of k columns and $\mathbf{K}$ of $n-k$ columns, and hope that all positions of $\mathbf{S}$ are **error-free** (i.e. $\mathbf{S}$ is an **information set**)
  - I.e. $\mathbf{H}' = \mathbf{HP} = [\mathbf{S} \mid \mathbf{K}]$ (Set also $\mathbf{e}' = \mathbf{eP}^\top$)
  - Probability that guess is correct $\binom{k}{0}\binom{n-k}{t}/\binom{n}{t}$
- Compute $\mathbf{U}$ s.t. $\mathbf{K}$ is Gauss-reduced
- If guess is correct, $\mathbf{sU}^\top$ has weight t
  - Cost $n(n-k)$ column operations

Information set decoding [Prange '62]



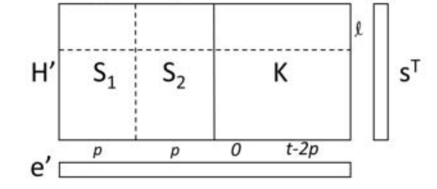
- Split $\mathbf{H}$ randomly in two parts $\mathbf{S}$ of k columns and $\mathbf{K}$ of $n-k$ columns, and hope that all positions of $\mathbf{S}$ are **error-free** (i.e. $\mathbf{S}$ is an **information set**)
  - I.e. $\mathbf{H}' = \mathbf{HP} = [\mathbf{S} \mid \mathbf{K}]$ (Set also $\mathbf{e}' = \mathbf{eP}^\top$)
  - Probability that guess is correct $\binom{k}{0}\binom{n-k}{t}/\binom{n}{t}$
- Compute $\mathbf{U}$ s.t. $\mathbf{K}$ is Gauss-reduced
- If guess is correct, $\mathbf{sU}^\top$ has weight t
  - Cost $n(n-k)$ column operations
- We can do slightly better by **relaxing "error-freeness"** of information set [Lee-Brickell '88]
  - better probability but more work

Introducing collision search in ISD [Stern ’89; Dumer ’91]



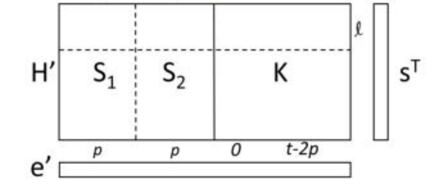
- Split **H** as before, except hope for the error distribution in the figure the positions of **S contain** $2p$ **errors, where** p **in left half** $\mathbf{S}_1$**, and** p **in right half** $\mathbf{S}_2$ **and there are no errors on chosen** ℓ **positions outside of information set**

Introducing collision search in ISD [Stern '89; Dumer '91]



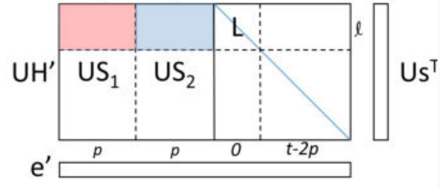
- Split **H** as before, except hope for the error distribution in the figure the positions of **S contain** $2p$ **errors, where** p **in left half** $\mathbf{S}_1$**, and** p **in right half** $\mathbf{S}_2$ **and there are no errors on chosen** ℓ **positions outside of information set**

Introducing collision search in ISD [Stern '89; Dumer '91]



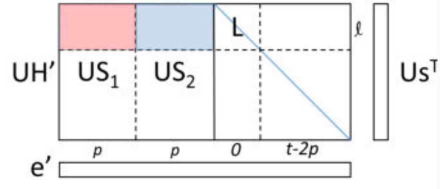
- Split **H** as before, except hope for the error distribution in the figure the positions of **S contain** $2p$ **errors, where** p **in left half** $\mathbf{S}_1$**, and** p **in right half** $\mathbf{S}_2$ **and there are no errors on chosen** ℓ **positions outside of information set**
- Gauss-reduce **K** as before

Introducing collision search in ISD [Stern '89; Dumer '91]



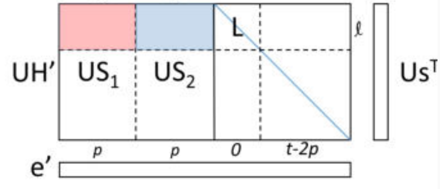
- Split $\mathbf{H}$ as before, except hope for the error distribution in the figure the positions of **S contain** $2p$ **errors, where** p **in left half** $\mathbf{S}_1$**, and** p **in right half** $\mathbf{S}_2$ **and there are no errors on chosen** ℓ **positions outside of information set**
- Gauss-reduce $\mathbf{K}$ as before
- If guess is correct, $\exists \bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2$ of weight p and $\mathbf{s}\mathbf{U}^\top + \bar{\mathbf{e}}_1 \mathbf{S}_1^\top \mathbf{U}^\top + \bar{\mathbf{e}}_2 \mathbf{S}_2^\top \mathbf{U}^\top$ has weight $t - 2p$

Introducing collision search in ISD [Stern '89; Dumer '91]



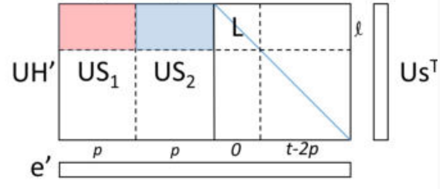
- Split **H** as before, except hope for the error distribution in the figure the positions of **S contain** $2p$ **errors, where** p **in left half** $\mathbf{S}_1$**, and** p **in right half** $\mathbf{S}_2$ **and there are no errors on chosen** ℓ **positions outside of information set**
- Gauss-reduce **K** as before
- If guess is correct, $\exists \bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2$ of weight p and $\mathbf{s}\mathbf{U}^\top + \bar{\mathbf{e}}_1\mathbf{S}_1^\top\mathbf{U}^\top + \bar{\mathbf{e}}_2\mathbf{S}_2^\top\mathbf{U}^\top$ has weight $t - 2p$
- so far, the same as before!

Introducing collision search in ISD [Stern '89; Dumer '91]



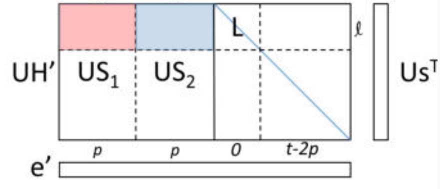
- Split $\mathbf{H}$ as before, except hope for the error distribution in the figure the positions of **S contain** $2p$ **errors, where** p **in left half** $\mathbf{S}_1$**, and** p **in right half** $\mathbf{S}_2$ **and there are no errors on chosen** ℓ **positions outside of information set**
- Gauss-reduce $\mathbf{K}$ as before
- If guess is correct, $\exists \bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2$ of weight p and $\mathbf{s}\mathbf{U}^\top + \bar{\mathbf{e}}_1 \mathbf{S}_1^\top \mathbf{U}^\top + \bar{\mathbf{e}}_2 \mathbf{S}_2^\top \mathbf{U}^\top$ has weight $t - 2p$
- so far, the same as before!
- Idea for speedup of finding $\bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2$: **collision search on smaller set of ℓ rows**

Introducing collision search in ISD [Stern '89; Dumer '91]



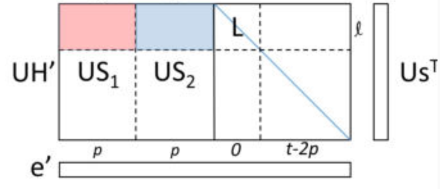
- Split $\mathbf{H}$ as before, except hope for the error distribution in the figure the positions of **S contain** $2p$ **errors, where** p **in left half** $\mathbf{S}_1$**, and** p **in right half** $\mathbf{S}_2$ **and there are no errors on chosen** ℓ **positions outside of information set**
- Gauss-reduce $\mathbf{K}$ as before
- If guess is correct, $\exists \bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2$ of weight p and $\mathbf{s}\mathbf{U}^\top + \bar{\mathbf{e}}_1\mathbf{S}_1^\top\mathbf{U}^\top + \bar{\mathbf{e}}_2\mathbf{S}_2^\top\mathbf{U}^\top$ has weight $t - 2p$
- so far, the same as before!
- Idea for speedup of finding $\bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2$: **collision search on smaller set of** ℓ **rows**
  - Collision benefits from birthday paradox

Introducing collision search in ISD [Stern '89; Dumer '91]



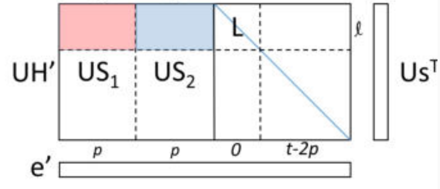
- Split **H** as before, except hope for the error distribution in the figure the positions of **S contain** $2p$ **errors, where** p **in left half** $\mathbf{S}_1$**, and** p **in right half** $\mathbf{S}_2$ **and there are no errors on chosen** ℓ **positions outside of information set**
- Gauss-reduce **K** as before
- If guess is correct, $\exists \bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2$ of weight p and $\mathbf{s}\mathbf{U}^\top + \bar{\mathbf{e}}_1\mathbf{S}_1^\top\mathbf{U}^\top + \bar{\mathbf{e}}_2\mathbf{S}_2^\top\mathbf{U}^\top$ has weight $t - 2p$
- so far, the same as before!
- Idea for speedup of finding $\bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2$: **collision search on smaller set of ℓ rows**
  - Collision benefits from birthday paradox
  - Smaller set of ℓ rows is like an early abort for non-collision

Introducing collision search in ISD [Stern '89; Dumer '91]



- Split $\mathbf{H}$ as before, except hope for the error distribution in the figure the positions of **S contain** $2p$ **errors, where** p **in left half** $\mathbf{S}_1$**, and** p **in right half** $\mathbf{S}_2$ **and there are no errors on chosen** ℓ **positions outside of information set**
- Gauss-reduce $\mathbf{K}$ as before
- If guess is correct, $\exists \bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2$ of weight p and $\mathbf{s}\mathbf{U}^\top + \bar{\mathbf{e}}_1 \mathbf{S}_1^\top \mathbf{U}^\top + \bar{\mathbf{e}}_2 \mathbf{S}_2^\top \mathbf{U}^\top$ has weight $t - 2p$
- so far, the same as before!
- Idea for speedup of finding $\bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2$: **collision search on smaller set of ℓ rows**
  - Collision benefits from birthday paradox
  - Smaller set of ℓ rows is like an early abort for non-collision
  - p and ℓ are small parameters that optimize the complexity

Introducing collision search in ISD [Stern '89; Dumer '91]



- Split $\mathbf{H}$ as before, except hope for the error distribution in the figure the positions of **S contain** $2p$ **errors, where** p **in left half** $\mathbf{S}_1$**, and** p **in right half** $\mathbf{S}_2$ **and there are no errors on chosen** ℓ **positions outside of information set**
- Gauss-reduce $\mathbf{K}$ as before
- If guess is correct, $\exists \bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2$ of weight p and $\mathbf{s}\mathbf{U}^\top + \bar{\mathbf{e}}_1\mathbf{S}_1^\top\mathbf{U}^\top + \bar{\mathbf{e}}_2\mathbf{S}_2^\top\mathbf{U}^\top$ has weight $t - 2p$
- so far, the same as before!
- Idea for speedup of finding $\bar{\mathbf{e}}_1, \bar{\mathbf{e}}_2$: **collision search on smaller set of ℓ rows**
  - Collision benefits from birthday paradox
  - Smaller set of ℓ rows is like an early abort for non-collision
  - p and ℓ are small parameters that optimize the complexity

ISD attacks timeline

- Information Set Decoding: [Prange '62] - $2^{0.1208n}$
- Allow non-perfect information set: [Lee & Brickell '88]
- Birthday improvement: [Stern, 89], [Dumer '91]
- Initial McEliece parameters broken: [Bernstein, Lange, & Peters '08]
- Ball-collision decoding [Bernstein, Lange, & Peters '11]
- Asymptotic exponent improved [May, Meurer, & Thomae '11]
- Decoding one out of many [Sendrier '11]
- Even better asymptotic exponent [Becker, Joux, May, & Meurer '12] - $2^{0.1019n}$
- "Nearest Neighbor" variant [May & Ozerov '15]
- Sublinear error weight [Canto Torres & Sendrier '16]
- ISD using Quantum walks (post-quantum) [Kachigar-Tillich '17]
- Nearest Neighbor BJMM [Both-May '17] - $2^{0.0953n}$
- Post-quantum "Nearest Neighbor" [Kirshanova '18]

Security of Code-based crypto – Other attacks

Other attacks

- **Dual attacks** (lattice style)
  - statistical decoding - reduce to LPN
  - outperform ISD for low rate codes
  - very recent, still work in progress [Carrier et al.'22, Meyer et al.'23]

Other attacks

- **Dual attacks** (lattice style)
  - statistical decoding - reduce to LPN
  - outperform ISD for low rate codes
  - very recent, still work in progress [Carrier et al.'22, Meyer et al.'23]
- **DOOM** (Decode One Out of Many)
  - Attacker is satisfied with one decoded ciphertext, when given many
  - ISD algorithms can be improved by $\mathcal{O}(\sqrt{n})$
  - influence on quasi-cyclic codes, MDPC codes

Other attacks

- **Dual attacks** (lattice style)
  - statistical decoding - reduce to LPN
  - outperform ISD for low rate codes
  - very recent, still work in progress [Carrier et al.'22, Meyer et al.'23]
- **DOOM** (Decode One Out of Many)
  - Attacker is satisfied with one decoded ciphertext, when given many
  - ISD algorithms can be improved by $\mathcal{O}(\sqrt{n})$
  - influence on quasi-cyclic codes, MDPC codes
- **Key recovery attacks**
  - LDPC codes - polynomial-time (constant density)
  - MDPC codes - generic decoding only $\mathcal{O}(2^{\sqrt{n}})$ (density $\mathcal{O}(\sqrt{n})$)
  - Algebraic attacks
    - Polynomial-time distinguisher for high-rate alternant and Goppa codes
    - No influence on Classic McEliece

Reaction attack [Verheul et al. '98]

Reaction attack [Verheul et al. ’98]

$\mathbf{m}_1, \mathbf{c}_1$

Reaction attack [Verheul et al. '98]

$\mathbf{m}_1, \mathbf{c}_1$ $\xrightarrow{\mathbf{c}_1}$

Reaction attack [Verheul et al. '98]

$\mathbf{m}_1, \mathbf{c}_1 \xrightarrow{\quad\quad \mathbf{c}_1 \quad\quad} \checkmark \leftarrow Decode(\mathbf{c}_1)$

Reaction attack [Verheul et al. '98]

$\mathbf{m}_1, \mathbf{c}_1$ $\xrightarrow{\quad\quad\mathbf{c}_1\quad\quad}$ $\checkmark \leftarrow \mathit{Decode}(\mathbf{c}_1)$

$\mathbf{m}_2, \mathbf{c}_2$

Reaction attack [Verheul et al. '98]

$$\mathbf{m}_1, \mathbf{c}_1 \xrightarrow{\qquad\qquad \mathbf{c}_1 \qquad\qquad} \checkmark \leftarrow \mathit{Decode}(\mathbf{c}_1)$$

$$\mathbf{m}_2, \mathbf{c}_2 \xrightarrow{\qquad\qquad \mathbf{c}_2 \qquad\qquad}$$

Reaction attack [Verheul et al. '98]

$$\mathbf{m}_1, \mathbf{c}_1 \xrightarrow{\quad\quad\quad \mathbf{c}_1 \quad\quad\quad} \checkmark \leftarrow \mathit{Decode}(\mathbf{c}_1)$$

$$\mathbf{m}_2, \mathbf{c}_2 \xrightarrow{\quad\quad\quad \mathbf{c}_2 \quad\quad\quad} \checkmark \leftarrow \mathit{Decode}(\mathbf{c}_2)$$

$$
\begin{array}{ccc}
\mathbf{m}_1, \mathbf{c}_1 & \xrightarrow{\quad\quad\mathbf{c}_1\quad\quad} & \checkmark \leftarrow \mathit{Decode}(\mathbf{c}_1) \\
\mathbf{m}_2, \mathbf{c}_2 & \xrightarrow{\quad\quad\mathbf{c}_2\quad\quad} & \checkmark \leftarrow \mathit{Decode}(\mathbf{c}_2) \\
 & \xrightarrow{\quad\quad\cdots\quad\quad} &
\end{array}
$$

Reaction attack [Verheul et al. '98]

$\mathbf{m}_1, \mathbf{c}_1$	$\xrightarrow{\quad \mathbf{c}_1 \quad}$	$\checkmark \leftarrow \mathit{Decode}(\mathbf{c}_1)$
$\mathbf{m}_2, \mathbf{c}_2$	$\xrightarrow{\quad \mathbf{c}_2 \quad}$	$\checkmark \leftarrow \mathit{Decode}(\mathbf{c}_2)$
	$\xrightarrow{\quad \dots \quad}$	
$\mathbf{m}_t, \mathbf{c}_t$		

Reaction attack [Verheul et al. '98]

$$\mathbf{m}_1, \mathbf{c}_1 \quad \xrightarrow{\quad \mathbf{c}_1 \quad} \quad \checkmark \leftarrow \mathit{Decode}(\mathbf{c}_1)$$

$$\mathbf{m}_2, \mathbf{c}_2 \quad \xrightarrow{\quad \mathbf{c}_2 \quad} \quad \checkmark \leftarrow \mathit{Decode}(\mathbf{c}_2)$$

$$\xrightarrow{\quad \dots \quad}$$

$$\mathbf{m}_t, \mathbf{c}_t \quad \xrightarrow{\quad \mathbf{c}_t \quad}$$

Reaction attack [Verheul et al. '98]

$$\mathbf{m}_1, \mathbf{c}_1 \quad \xrightarrow{\qquad \mathbf{c}_1 \qquad} \quad \checkmark \leftarrow \mathit{Decode}(\mathbf{c}_1)$$

$$\mathbf{m}_2, \mathbf{c}_2 \quad \xrightarrow{\qquad \mathbf{c}_2 \qquad} \quad \checkmark \leftarrow \mathit{Decode}(\mathbf{c}_2)$$

$$\xrightarrow{\qquad \dots \qquad}$$

$$\mathbf{m}_t, \mathbf{c}_t \quad \xrightarrow{\qquad \mathbf{c}_t \qquad} \quad \text{X} \leftarrow \mathit{Decode}(\mathbf{c}_t)$$

Reaction attack [Verheul et al. '98]

$\mathbf{m}_1, \mathbf{c}_1 \xrightarrow{\mathbf{c}_1} \checkmark \leftarrow \mathit{Decode}(\mathbf{c}_1)$

$\mathbf{m}_2, \mathbf{c}_2 \xrightarrow{\mathbf{c}_2} \checkmark \leftarrow \mathit{Decode}(\mathbf{c}_2)$

$\xrightarrow{\cdots}$

$\mathbf{m}_t, \mathbf{c}_t \xrightarrow{\mathbf{c}_t} \text{X} \leftarrow \mathit{Decode}(\mathbf{c}_t)$

$\xleftarrow{\text{Pls resend!}}$

Reaction attack [Verheul et al. '98]

$\mathbf{m}_1, \mathbf{c}_1 \xrightarrow{\quad\mathbf{c}_1\quad} \checkmark \leftarrow \mathit{Decode}(\mathbf{c}_1)$

$\mathbf{m}_2, \mathbf{c}_2 \xrightarrow{\quad\mathbf{c}_2\quad} \checkmark \leftarrow \mathit{Decode}(\mathbf{c}_2)$

$\xrightarrow{\quad\ldots\quad}$

$\mathbf{m}_t, \mathbf{c}_t \xrightarrow{\quad\mathbf{c}_t\quad} X \leftarrow \mathit{Decode}(\mathbf{c}_t)$

$\xleftarrow{\quad\text{Pls resend!}\quad}$

A reaction (decoding failure) attack on Niederreiter

- Idea: iteratively test the error vector positions
- For position i, $\mathbf{s}' \leftarrow \mathbf{s} \oplus \mathbf{H}_i$

$$\mathbf{H} \quad \cdot \quad \mathbf{e} = \mathbf{s}$$

$$\begin{bmatrix} 1 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

A reaction (decoding failure) attack on Niederreiter

- Idea: iteratively test the error vector positions
- For position i, $\quad \mathbf{s}' \leftarrow \mathbf{s} \oplus \mathbf{H}_i$

$$\mathbf{H} \cdot \mathbf{e} = \mathbf{s}$$

$$\begin{bmatrix} 1 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

A reaction (decoding failure) attack on Niederreiter

- Idea: iteratively test the error vector positions
- For position i, $\mathbf{s}' \leftarrow \mathbf{s} \oplus \mathbf{H}_i$

$$\mathbf{H} \cdot \mathbf{e} = \mathbf{s}$$

$$\begin{bmatrix} 1 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

A reaction (decoding failure) attack on Niederreiter

- Idea: iteratively test the error vector positions
- For position i, $\quad \mathbf{s}' \leftarrow \mathbf{s} \oplus \mathbf{H}_i$

$$\mathbf{H} \quad \cdot \quad \mathbf{e} \quad = \quad \mathbf{s}'$$

$$\begin{bmatrix} 1 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

A reaction (decoding failure) attack on Niederreiter

- Idea: iteratively test the error vector positions
- For position i, $\mathbf{s}' \leftarrow \mathbf{s} \oplus \mathbf{H}_i$
- Ask a decoding failure oracle whether t is exceeded
  - A redundant error ($\mathbf{e}_i = 1$) cancels out
  - An additional error ($\mathbf{e}_i = 0$) leads to failure

$$\mathbf{H} \cdot \mathbf{e} = \mathbf{s}$$

$$\begin{bmatrix} 1 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

A reaction (decoding failure) attack on Niederreiter

- Idea: iteratively test the error vector positions
- For position i, $\mathbf{s}' \leftarrow \mathbf{s} \oplus \mathbf{H}_i$
- Ask a decoding failure oracle whether t is exceeded
  - A redundant error ($\mathbf{e}_i = 1$) cancels out
  - An additional error ($\mathbf{e}_i = 0$) leads to failure
- Effort: k queries
  - Attacker only needs to recover an information set

$$\mathbf{H} \cdot \mathbf{e} = \mathbf{s}$$

$$\begin{bmatrix} 1 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ \vdots \\ \mathbf{1} \\ 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

A reaction (decoding failure) attack on Niederreiter

- Idea: iteratively test the error vector positions
- For position i, $\mathbf{s}' \leftarrow \mathbf{s} \oplus \mathbf{H}_i$
- Ask a decoding failure oracle whether t is exceeded
  - A redundant error ($\mathbf{e}_i = 1$) cancels out
  - An additional error ($\mathbf{e}_i = 0$) leads to failure
- Effort: k queries
  - Attacker only needs to recover an information set
- Even lees by iterative chunking
- Even less if attacker has some computational power to solve a smaller Information Set Decoding problem

$$\mathbf{H} \cdot \mathbf{e} = \mathbf{s}$$

$$\begin{bmatrix} 1 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Side-channel attacks

- Previous attack causes decoding failures regardless of the decoding failure rate of the scheme (if any at all)

Side-channel attacks

- Previous attack causes decoding failures regardless of the decoding failure rate of the scheme (if any at all)
- Side-channel decoding failure oracle can be efficiently constructed
  - easy to turn into a message recovery attack (as seen above)

Side-channel attacks

- Previous attack causes decoding failures regardless of the decoding failure rate of the scheme (if any at all)
- Side-channel decoding failure oracle can be efficiently constructed
  - easy to turn into a message recovery attack (as seen above)
- Recently first key recovery attack on McEliece [Guo,Johansson,Johansson '22]
- Very recently, high-redundancy used in key recovery attack on McEliece [Brinkmann et al.'23]

Side-channel attacks

- Previous attack causes decoding failures regardless of the decoding failure rate of the scheme (if any at all)
- Side-channel decoding failure oracle can be efficiently constructed
  - easy to turn into a message recovery attack (as seen above)
- Recently first key recovery attack on McEliece [Guo,Johansson,Johansson ’22]
- Very recently, high-redundancy used in key recovery attack on McEliece [Brinkmann et al.’23]
- Decoding algorithms are difficult to make constant time
  - Timing attacks on McEliece, HQC, BIKE
  - Rejection sampling exploited in HQC, BIKE

Side-channel attacks

- Previous attack causes decoding failures regardless of the decoding failure rate of the scheme (if any at all)
- Side-channel decoding failure oracle can be efficiently constructed
  - easy to turn into a message recovery attack (as seen above)
- Recently first key recovery attack on McEliece [Guo,Johansson,Johansson '22]
- Very recently, high-redundancy used in key recovery attack on McEliece [Brinkmann et al.'23]
- Decoding algorithms are difficult to make constant time
  - Timing attacks on McEliece, HQC, BIKE
  - Rejection sampling exploited in HQC, BIKE
- **Side-channel attacks - biggest issue currently for post-quantum schemes**
  - For code-based schemes still not many countermeasures available
  - Protection influences performance significantly
  - Cheap and effective countermeasures are necessary

Side-channel attacks

- Previous attack causes decoding failures regardless of the decoding failure rate of the scheme (if any at all)
- Side-channel decoding failure oracle can be efficiently constructed
  - easy to turn into a message recovery attack (as seen above)
- Recently first key recovery attack on McEliece [Guo,Johansson,Johansson '22]
- Very recently, high-redundancy used in key recovery attack on McEliece [Brinkmann et al.'23]
- Decoding algorithms are difficult to make constant time
  - Timing attacks on McEliece, HQC, BIKE
  - Rejection sampling exploited in HQC, BIKE
- **Side-channel attacks - biggest issue currently for post-quantum schemes**
  - For code-based schemes still not many countermeasures available
  - Protection influences performance significantly
  - Cheap and effective countermeasures are necessary

NIST's additional round on signatures

Code-based candidates

- Enhanced pqsigRM
- FuLeeca
- Wave
- CROSS
- SDitH
- LESS
- MEDS
- ALTEQ

Code-based candidates

- Enhanced pqsigRM
- FuLeeca
- Wave
- CROSS
- SDitH
- LESS
- MEDS
- ALTEQ

- **5 Fiat-Shamir signatures**
- **3 of them based on equivalence problem**

Digital signatures via the Fiat-Shamir transform

Σ-protocol

$\mathcal{P}(\mathsf{pk}, \mathsf{sk})$		$\mathcal{V}(\mathsf{pk})$
$\mathsf{com} \leftarrow \mathcal{P}_0\ (\mathsf{sk})$	$\xrightarrow{\mathsf{com}}$	
	$\xleftarrow{\mathsf{ch}}$	$\mathsf{ch} \xleftarrow{\$} \mathsf{ChS}\ (1^k)$
$\mathsf{resp} \leftarrow \mathcal{P}_1\ (\mathsf{sk}, \mathsf{com}, \mathsf{ch})$	$\xrightarrow{\mathsf{resp}}$	
		$b \leftarrow \mathsf{Vf}\ (\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$

Digital signatures via the Fiat-Shamir transform

Σ-protocol

$$
\begin{array}{lcl}
\mathcal{P}(\mathsf{pk}, \mathsf{sk}) & & \mathcal{V}(\mathsf{pk}) \\
\hline
\mathsf{com} \leftarrow {\mathcal{P}_0}^{r}(\mathsf{sk}) & & \\
 & \xrightarrow{\quad \mathsf{com} \quad} & \\
 & & \mathsf{ch} \xleftarrow{\$} \mathsf{ChS}^{r}(1^k) \\
 & \xleftarrow{\quad \mathsf{ch} \quad} & \\
\mathsf{resp} \leftarrow {\mathcal{P}_1}^{r}(\mathsf{sk}, \mathsf{com}, \mathsf{ch}) & & \\
 & \xrightarrow{\quad \mathsf{resp} \quad} & \\
 & & b \leftarrow \mathsf{Vf}^{r}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})
\end{array}
$$

Digital signatures via the Fiat-Shamir transform

Σ-protocol

$\mathcal{P}(\mathsf{pk}, \mathsf{sk})$		$\mathcal{V}(\mathsf{pk})$
$\mathsf{com} \leftarrow \mathcal{P}_0{}^r(\mathsf{sk})$	$\xrightarrow{\mathsf{com}}$	
$\mathsf{ch} \xleftarrow{\$} \mathsf{ChS}^r(1^k)$	$\xleftarrow{\mathsf{ch}}$	
$\mathsf{resp} \leftarrow \mathcal{P}_1{}^r(\mathsf{sk}, \mathsf{com}, \mathsf{ch})$	$\xrightarrow{\mathsf{resp}}$	
		$b \leftarrow \mathsf{Vf}^r(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$

$\downarrow \quad \downarrow \quad \downarrow$

FS signature

Signer
$\mathsf{com} \leftarrow \mathcal{P}_0{}^r(\mathsf{sk})$
$\mathsf{ch} \leftarrow H(m, \mathsf{com})$
$\mathsf{resp} \leftarrow \mathcal{P}_1{}^r(\mathsf{sk}, \mathsf{com}, \mathsf{ch})$
output : $\sigma = (\mathsf{com}, \mathsf{resp})$

Verifier
$\mathsf{ch} \leftarrow H(m, \mathsf{com})$
$b \leftarrow \mathsf{Vf}^r(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$
output : b

Digital signatured based on hard equivalence problems

Code equivalence problem $\mathsf{CE}(\mathcal{C}_0, \mathcal{C}_1)$:
Given $\mathcal{C}_0$ and $\mathcal{C}_1$, find (if any) an isometry (preserves metric) ϕ s.t. $\mathcal{C}_1 = \phi(\mathcal{C}_0)$

Digital signatured based on hard equivalence problems

Code equivalence problem CE$(\mathcal{C}_0, \mathcal{C}_1)$:
Given $\mathcal{C}_0$ and $\mathcal{C}_1$, find (if any) an isometry (preserves metric) ϕ s.t. $\mathcal{C}_1 = \phi(\mathcal{C}_0)$

Hard code-based equivalence problems in NIST's 4th round?

- **Code equivalence** - **LESS** - Baldi et. al (Biasse et al.'20, LESS-FM - Barenghi et al.'21)
  - Hamming metric, linear codes
  - isometry defined by permutation matrix

Digital signatured based on hard equivalence problems

Code equivalence problem $\mathrm{CE}(\mathcal{C}_0, \mathcal{C}_1)$:
Given $\mathcal{C}_0$ and $\mathcal{C}_1$, find (if any) an isometry (preserves metric) ϕ s.t. $\mathcal{C}_1 = \phi(\mathcal{C}_0)$

Hard code-based equivalence problems in NIST's 4th round?

- **Code equivalence** - **LESS** - Baldi et. al (Biasse et al.'20, LESS-FM - Barenghi et al.'21)
  - Hamming metric, linear codes
  - isometry defined by permutation matrix
- **Matrix code equivalence** - **MEDS** - with T.Chou, R.Niederhagen, E.Persichetti, T.H.Randrianarisoa, L.Ran, K.Reijnders, M.Trimoska, '22

Digital signatured based on hard equivalence problems

Code equivalence problem CE($\mathcal{C}_0, \mathcal{C}_1$):
Given $\mathcal{C}_0$ and $\mathcal{C}_1$, find (if any) an isometry (preserves metric) ϕ s.t. $\mathcal{C}_1 = \phi(\mathcal{C}_0)$

Hard code-based equivalence problems in NIST's 4th round?

- **Code equivalence** - **LESS** - Baldi et. al (Biasse et al.'20, LESS-FM - Barenghi et al.'21)
  - Hamming metric, linear codes
  - isometry defined by permutation matrix
- **Matrix code equivalence** - **MEDS** - with T.Chou, R.Niederhagen, E.Persichetti, T.H.Randrianarisoa, L.Ran, K.Reijnders, M.Trimoska, '22
  - Rank metric, matrix codes
  - isometry defined by non-singular matrices $\mathbf{A}, \mathbf{B}$

Digital signatured based on hard equivalence problems

Code equivalence problem CE$(\mathcal{C}_0, \mathcal{C}_1)$:
Given $\mathcal{C}_0$ and $\mathcal{C}_1$, find (if any) an isometry (preserves metric) ϕ s.t. $\mathcal{C}_1 = \phi(\mathcal{C}_0)$

Hard code-based equivalence problems in NIST's 4th round?

- **Code equivalence** - **LESS** - Baldi et. al (Biasse et al.'20, LESS-FM - Barenghi et al.'21)
  - Hamming metric, linear codes
  - isometry defined by permutation matrix
- **Matrix code equivalence** - **MEDS** - with T.Chou, R.Niederhagen, E.Persichetti, T.H.Randrianarisoa, L.Ran, K.Reijnders, M.Trimoska, '22
  - Rank metric, matrix codes
  - isometry defined by non-singular matrices $\mathbf{A}, \mathbf{B}$
- **Alternate trilinear form equivalence** - **ALTEQ** - Blase et al. (Tang et al.'22)
  - Rank metric, skew-symmetric matrix codes
  - isometry defined by non-singular matrix $\mathbf{A}$

Σ protocol from Code Equivalence Problems

Let ϕ be an isometry s.t. $\mathcal{C}_1 = \phi(\mathcal{C}_0)$.
Given $\mathcal{C}_0, \mathcal{C}_1$, the prover $\mathcal{P}$ wants to prove to the verifier $\mathcal{V}$ knowledge of ϕ without revealing any information about it

$$\begin{tikzcd}
\mathcal{C}_0 \arrow[d, dashed, "\phi"'] \\
\mathcal{C}_1
\end{tikzcd}$$

$\mathcal{P}(\mathcal{C}_0, \mathcal{C}_1, \phi)$	$\mathcal{V}(\mathcal{C}_0, \mathcal{C}_1)$

Σ protocol from Code Equivalence Problems

Let ϕ be an isometry s.t. $\mathcal{C}_1 = \phi(\mathcal{C}_0)$.

Given $\mathcal{C}_0, \mathcal{C}_1$, the prover $\mathcal{P}$ wants to prove to the verifier $\mathcal{V}$ knowledge of ϕ without revealing any information about it

$$\begin{tikzcd}
\mathcal{C}_0 \arrow[r, "\phi_0"] \arrow[d, dashed, "\phi"'] & \mathcal{C}' \\
\mathcal{C}_1 &
\end{tikzcd}$$

$\mathcal{P}(\mathcal{C}_0, \mathcal{C}_1, \phi)$	$\mathcal{V}(\mathcal{C}_0, \mathcal{C}_1)$

Let ϕ be an isometry s.t. $\mathcal{C}_1 = \phi(\mathcal{C}_0)$.
Given $\mathcal{C}_0, \mathcal{C}_1$, the prover $\mathcal{P}$ wants to prove to the verifier $\mathcal{V}$ knowledge of ϕ without revealing any information about it

$$\begin{tikzcd}
\mathcal{C}_0 \arrow[r, "\phi_0"] \arrow[d, dashed, "\phi"'] & \mathcal{C}' \\
\mathcal{C}_1 \arrow[ur, "\phi_1"'] &
\end{tikzcd}$$

$\mathcal{P}(\mathcal{C}_0, \mathcal{C}_1, \phi)$	$\mathcal{V}(\mathcal{C}_0, \mathcal{C}_1)$

Σ protocol from Code Equivalence Problems

Let ϕ be an isometry s.t. $\mathcal{C}_1 = \phi(\mathcal{C}_0)$.

Given $\mathcal{C}_0, \mathcal{C}_1$, the prover $\mathcal{P}$ wants to prove to the verifier $\mathcal{V}$ knowledge of ϕ without revealing any information about it

$$\begin{tikzcd}
\mathcal{C}_0 \arrow[d, dashed, "\phi"'] & \mathcal{C}' \\
\mathcal{C}_1 &
\end{tikzcd}$$

$\mathcal{P}(\mathcal{C}_0, \mathcal{C}_1, \phi)$	$\mathcal{V}(\mathcal{C}_0, \mathcal{C}_1)$
$\mathsf{com} \leftarrow \mathcal{C}'$	

Σ protocol from Code Equivalence Problems

Let ϕ be an isometry s.t. $\mathcal{C}_1 = \phi(\mathcal{C}_0)$.

Given $\mathcal{C}_0, \mathcal{C}_1$, the prover $\mathcal{P}$ wants to prove to the verifier $\mathcal{V}$ knowledge of ϕ without revealing any information about it

$$\begin{tikzcd} \mathcal{C}_0 \arrow[d, dashed, "\phi"'] & \mathcal{C}' \\ \mathcal{C}_1 & \end{tikzcd}$$

$\mathcal{P}(\mathcal{C}_0, \mathcal{C}_1, \phi)$		$\mathcal{V}(\mathcal{C}_0, \mathcal{C}_1)$
com $\leftarrow \mathcal{C}'$	$\xrightarrow{\text{com}}$	

Σ protocol from Code Equivalence Problems

Let ϕ be an isometry s.t. $\mathcal{C}_1 = \phi(\mathcal{C}_0)$.
Given $\mathcal{C}_0, \mathcal{C}_1$, the prover $\mathcal{P}$ wants to prove to the verifier $\mathcal{V}$ knowledge of ϕ without revealing any information about it

$$\begin{tikzcd} \mathcal{C}_0 \arrow[d, dashed, "\phi"'] & & \mathcal{C}' \\ \mathcal{C}_1 & & \end{tikzcd}$$

$\mathcal{P}(\mathcal{C}_0, \mathcal{C}_1, \phi)$		$\mathcal{V}(\mathcal{C}_0, \mathcal{C}_1)$
$\mathsf{com} \leftarrow \mathcal{C}'$	$\xrightarrow{\quad \mathsf{com} \quad}$	
		$\mathsf{ch} \xleftarrow{\$} \{0, 1\}$

Σ protocol from Code Equivalence Problems

Let ϕ be an isometry s.t. $\mathcal{C}_1 = \phi(\mathcal{C}_0)$.

Given $\mathcal{C}_0, \mathcal{C}_1$, the prover $\mathcal{P}$ wants to prove to the verifier $\mathcal{V}$ knowledge of ϕ without revealing any information about it

$$
\begin{tikzcd}
\mathcal{C}_0 \arrow[d, dashed, "\phi"'] & \mathcal{C}' \\
\mathcal{C}_1 &
\end{tikzcd}
$$

$\mathcal{P}(\mathcal{C}_0, \mathcal{C}_1, \phi)$		$\mathcal{V}(\mathcal{C}_0, \mathcal{C}_1)$
$\text{com} \leftarrow \mathcal{C}'$	$\xrightarrow{\text{com}}$	
	$\xleftarrow{\text{ch}}$	$\text{ch} \xleftarrow{\$} \{0, 1\}$

Let ϕ be an isometry s.t. $\mathcal{C}_1 = \phi(\mathcal{C}_0)$.

Given $\mathcal{C}_0, \mathcal{C}_1$, the prover $\mathcal{P}$ wants to prove to the verifier $\mathcal{V}$ knowledge of ϕ without revealing any information about it

$$\begin{tikzcd}
\mathcal{C}_0 \arrow[d, dashed, "\phi"'] & & \mathcal{C}' \\
\mathcal{C}_1 & &
\end{tikzcd}$$

$\mathcal{P}(\mathcal{C}_0, \mathcal{C}_1, \phi)$		$\mathcal{V}(\mathcal{C}_0, \mathcal{C}_1)$
com $\leftarrow \mathcal{C}'$	$\xrightarrow{\text{com}}$	
	$\xleftarrow{\text{ch}}$	ch $\xleftarrow{\$} \{0, 1\}$
resp $\leftarrow \phi_{\text{ch}}$		

Let ϕ be an isometry s.t. $\mathcal{C}_1 = \phi(\mathcal{C}_0)$.

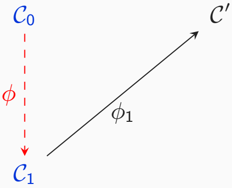
Given $\mathcal{C}_0, \mathcal{C}_1$, the prover $\mathcal{P}$ wants to prove to the verifier $\mathcal{V}$ knowledge of ϕ without revealing any information about it

$$\begin{tikzcd} \mathcal{C}_0 \arrow[r, "\phi_0"] \arrow[d, dashed, "\phi"'] & \mathcal{C}' \\ \mathcal{C}_1 & \end{tikzcd}$$

$\mathcal{P}(\mathcal{C}_0, \mathcal{C}_1, \phi)$		$\mathcal{V}(\mathcal{C}_0, \mathcal{C}_1)$
$\text{com} \leftarrow \mathcal{C}'$	$\xrightarrow{\text{com}}$	
	$\xleftarrow{\text{ch}}$	$\text{ch} \xleftarrow{\$} \{0, 1\}$
$\text{resp} \leftarrow \phi_{\text{ch}}$	$\xrightarrow{\text{resp}}$	

Σ protocol from Code Equivalence Problems

Let ϕ be an isometry s.t. $\mathcal{C}_1 = \phi(\mathcal{C}_0)$.
Given $\mathcal{C}_0, \mathcal{C}_1$, the prover $\mathcal{P}$ wants to prove to the verifier $\mathcal{V}$ knowledge of ϕ without revealing any information about it



$\mathcal{P}(\mathcal{C}_0, \mathcal{C}_1, \phi)$		$\mathcal{V}(\mathcal{C}_0, \mathcal{C}_1)$
com $\leftarrow \mathcal{C}'$	$\xrightarrow{\text{com}}$	
	$\xleftarrow{\text{ch}}$	ch $\xleftarrow{\$} \{0, 1\}$
resp $\leftarrow \phi_{\text{ch}}$	$\xrightarrow{\text{resp}}$	

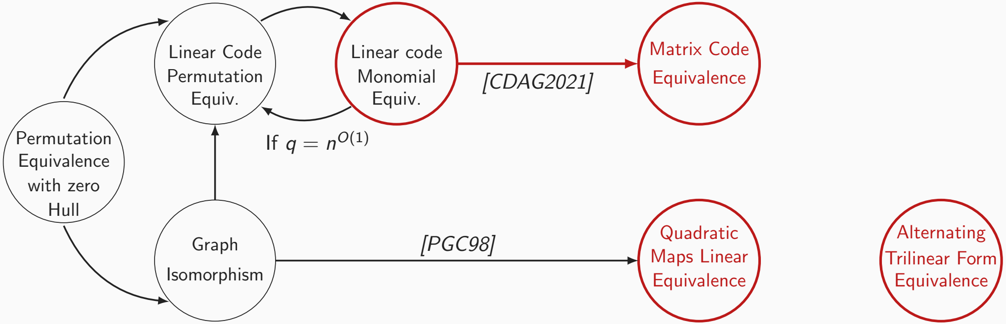
Σ protocol from Code Equivalence Problems

Let ϕ be an isometry s.t. $\mathcal{C}_1 = \phi(\mathcal{C}_0)$.
Given $\mathcal{C}_0, \mathcal{C}_1$, the prover $\mathcal{P}$ wants to prove to the verifier $\mathcal{V}$ knowledge of ϕ without revealing any information about it

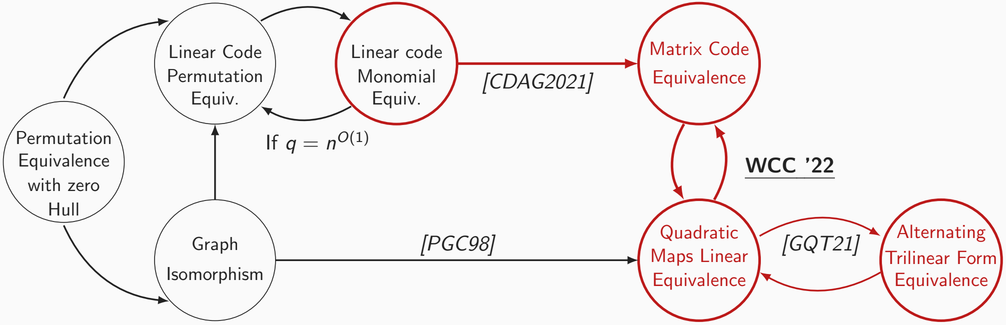
$$\begin{tikzcd}
\mathcal{C}_0 \arrow[r, "\phi_0"] \arrow[d, dashed, "\phi"'] & \mathcal{C}' \\
\mathcal{C}_1 \arrow[ur, "\phi_1"'] &
\end{tikzcd}$$

$\mathcal{P}(\mathcal{C}_0, \mathcal{C}_1, \phi)$		$\mathcal{V}(\mathcal{C}_0, \mathcal{C}_1)$
com $\leftarrow \mathcal{C}'$	$\xrightarrow{\text{com}}$	
	$\xleftarrow{\text{ch}}$	ch $\xleftarrow{\$} \{0, 1\}$
resp $\leftarrow \phi_{\text{ch}}$	$\xrightarrow{\text{resp}}$	
		$\mathcal{C}' \stackrel{?}{=} \phi_{\text{ch}}(\mathcal{C}_{\text{ch}})$

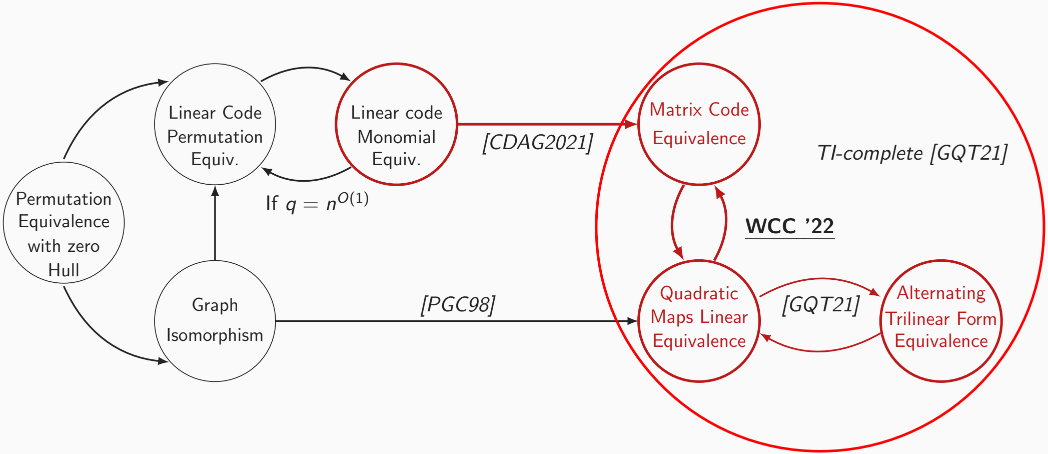
Relation between problems



Relation between problems



- Homogenous Quadratic Maps Linear Equivalence (hQMLE) problem is well known equivalence problem from multivariate crypto (instance of Isomorphism of Polynomials)



- Homogenous Quadratic Maps Linear Equivalence (hQMLE) problem is well known equivalence problem from multivariate crypto (instance of Isomorphism of Polynomials)

Parameters and performance of LESS, MEDS, ALTEQ

Level	param. set	public key size (KB)	signature size (KB)
I	LESS-1b	13.7	8.4
I	MEDS-9923	9.9	9.9
I	ALTEQ Balanced	8	16
III	LESS-3b	34.5	18.4
III	MEDS-41711	41.7	41
III	ALTEQ Balanced	32	48

- **Standard optimizations:** Multiple Public Keys + Fixed-Weight Challenge Strings + Seed tree

Parameters and performance of LESS, MEDS, ALTEQ

Level	param. set	public key size (KB)	signature size (KB)
I	LESS-1b	13.7	8.4
I	MEDS-9923	9.9	9.9
I	ALTEQ Balanced	8	16
III	LESS-3b	34.5	18.4
III	MEDS-41711	41.7	41
III	ALTEQ Balanced	32	48

- **Standard optimizations:** Multiple Public Keys + Fixed-Weight Challenge Strings + Seed tree
- **New in MEDS:** Public Key Compression
    - generate public key partially from seed $\Rightarrow$ signature size reduction
    - **Work in progress:** use similar idea during signing

Parameters and performance of LESS, MEDS, ALTEQ

Level	param. set	public key size (KB)	signature size (KB)
I	LESS-1b	13.7	8.4
I	MEDS-9923	9.9	9.9
I	ALTEQ Balanced	8	16
III	LESS-3b	34.5	18.4
III	MEDS-41711	41.7	41
III	ALTEQ Balanced	32	48

- **Standard optimizations:** Multiple Public Keys + Fixed-Weight Challenge Strings + Seed tree
- **New in MEDS:** Public Key Compression
  - generate public key partially from seed $\Rightarrow$ signature size reduction
  - **Work in progress:** use similar idea during signing
- **Brand new in LESS:** Information Set formulation, Canonical forms
  - significant signature reduction

Thank you for listening!

Post-Quantum Cryptography Conference

PKI Consortium