QcBits:

constant-time small-key code-based cryptography

Tung Chou

Technische Universiteit Eindhoven, The Netherlands

Linear codes

Linear codes

ullet a linear subspace in \mathbb{F}_2^N

Linear codes

- a linear subspace in \mathbb{F}_2^N
- can be defined by a parity-check matrix H, e.g.,

$$C = \{c \mid Hc = 0\}$$

Linear codes

- a linear subspace in \mathbb{F}_2^N
- can be defined by a parity-check matrix H, e.g.,

$$C = \{c \mid Hc = 0\}$$

Decoding

Linear codes

- a linear subspace in \mathbb{F}_2^N
- can be defined by a parity-check matrix H, e.g.,

$$C = \{c \mid Hc = 0\}$$

Decoding

• compute e (or c) given c + e, where e is of weight $\leq t$

Linear codes

- a linear subspace in \mathbb{F}_2^N
- can be defined by a parity-check matrix H, e.g.,

$$C = \{c \mid Hc = 0\}$$

Decoding

- compute e (or c) given c + e, where e is of weight $\leq t$
- compute e given the syndrome He = H(c + e)

Code-based encryption

McEliece versus Niederreiter

	plaintext	ciphertext
McEliece	С	c + e
Niederreiter	е	H^*e

Code-based encryption

McEliece versus Niederreiter

	plaintext	ciphertext
McEliece	С	c + e
Niederreiter	е	H [∗] e

• General shape

 ${\sf McEliece/Niederreiter} + {\bf some} \ {\bf code}$

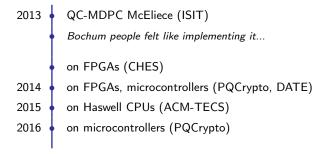
	Binary Goppa codes	QC-MDPC codes	
Confidence	unbroken since 1978	unbroken since 2013	

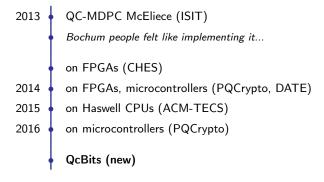
	Binary Goppa codes	QC-MDPC codes	
Confidence	unbroken since 1978	unbroken since 2013	
Efficiency	fast (McBits, CHES 2013)	not so fast	

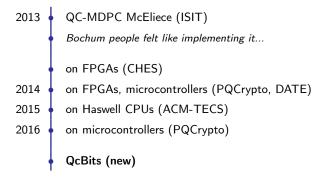
	Binary Goppa codes	QC-MDPC codes	
Confidence	unbroken since 1978	unbroken since 2013	
Efficiency	fast (McBits, CHES 2013)	not so fast	
Key size	≈ 100 kilobytes	pprox 1 kilobyte	

2013 QC-MDPC McEliece (ISIT)

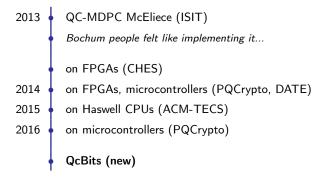




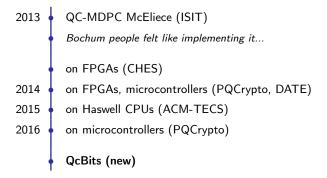




The problem is timing attacks.

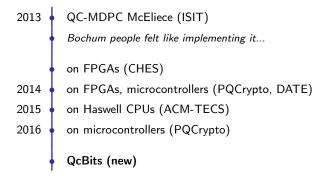


The problem is timing attacks.



The problem is timing attacks.

PQCrypto 2014: constant-time operations assuming no caches



The problem is timing attacks.

- PQCrypto 2014: constant-time operations assuming no caches
- QcBits: constant-time for a wide-variety of 32/64-bit platforms

Performance results

platform	key-pair	encrypt	decrypt	reference	scheme
Haswell	784 192	82 732	1 560 072	(new) QcBits	KEM/DEM
	14 234 347	34 123	3 104 624	ACMTECS 2015	McEliece
Cortex-M4	140 372 822	2 244 489	14 679 937	(new) QcBits	KEM/DEM
	63 185 108	2 623 432	18 416 012	PQCrypto 2016	KEM/DEM
	148 576 008	7 018 493	42 129 589	PQCrypto 2014	McEliece

Cycle counts for key-pair generation, encryption, and decryption for 80-bit pre-quantum security. Numbers in RED are non-constant-time. Numbers in BLUE are constant-time.

• MDPC: moderate-density-parity-check

- MDPC: moderate-density-parity-check
- QC: quasi-cyclic (for saving bandwidth and memory)

- MDPC: moderate-density-parity-check
- QC: quasi-cyclic (for saving bandwidth and memory)

$$(H^{(0)} \quad H^{(1)}) = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \in \mathbb{F}_2^{n \times 2n}$$

- MDPC: moderate-density-parity-check
- QC: quasi-cyclic (for saving bandwidth and memory)

$$(H^{(0)} \quad H^{(1)}) = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \in \mathbb{F}_2^{n \times 2n}$$

QcBits:

• [n = 4801, w = 90, t = 84] for 80-bit security

- MDPC: moderate-density-parity-check
- QC: quasi-cyclic (for saving bandwidth and memory)

$$(H^{(0)} \quad H^{(1)}) = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \in \mathbb{F}_2^{n \times 2n}$$

QcBits:

- [n = 4801, w = 90, t = 84] for 80-bit security
- further requires $H^{(i)}$ to have row weight w/2 (same for the Bochum papers)

Start with finding v = c + e such that $H^*v = H^*e$. Compute Hv.

Start with finding v = c + e such that $H^*v = H^*e$. Compute Hv.

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} v = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Start with finding v = c + e such that $H^*v = H^*e$. Compute Hv.

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} v = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Start with finding v = c + e such that $H^*v = H^*e$. Compute Hv.

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} v = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$u = (2 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 2) \in \mathbb{Z}^{2n}$$

Start with finding v = c + e such that $H^*v = H^*e$. Compute Hv.

$$\begin{pmatrix}
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0
\end{pmatrix} v = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$v = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$v = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$v = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Flip v_i if u_i is large. Repeat until Hv = 0.

Start with finding v = c + e such that $H^*v = H^*e$. Compute Hv.

$$\begin{pmatrix}
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0
\end{pmatrix} v = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$u = \begin{pmatrix} 2 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 2 \end{pmatrix} \in \mathbb{Z}^{2n}$$

Flip v_i if u_i is large. Repeat until Hv = 0.

Rationale

Start with finding v = c + e such that $H^*v = H^*e$. Compute Hv.

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} v = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$u = \begin{pmatrix} 2 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 2 \end{pmatrix} \in \mathbb{Z}^{2n}$$

Flip v_i if u_i is large. Repeat until Hv = 0.

Rationale

• parity= 0: perhaps no errors. no information.

Start with finding v = c + e such that $H^*v = H^*e$. Compute Hv.

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} v = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$+)$$

$$u = \begin{pmatrix} 2 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 2 \end{pmatrix} \in \mathbb{Z}^{2n}$$

Flip v_i if u_i is large. Repeat until Hv = 0.

Rationale

- parity= 0: perhaps no errors. no information.
- parity= 1: one score for each possible position.

8

 ${\sf Natural\ questions}$

Natural questions

• what do you mean by higher probability? (don't know)

Natural questions

- what do you mean by higher probability? (don't know)
- repeat how many times? (don't know)

Natural questions

- what do you mean by higher probability? (don't know)
- repeat how many times? (don't know)
- always work? (probably not)

Natural questions

- what do you mean by higher probability? (don't know)
- repeat how many times? (don't know)
- always work? (probably not)
- constant-time iterations?

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} v = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} v = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$+)$$

$$u = \begin{pmatrix} 2 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 2 \end{pmatrix} \in \mathbb{Z}^{2n}$$

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} v = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$u = \begin{pmatrix} 2 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 2 \end{pmatrix} \in \mathbb{Z}^{2n}$$

Step 1 computing the syndrome: $O(n^2)$

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} v = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$v = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$u = \begin{pmatrix} 2 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 2 \end{pmatrix} \in \mathbb{Z}^{2n}$$

Step 1 computing the syndrome: $O(n^2)$

Step 2 computing the unsatisfied parity checks: $O(n^2)$

$$\begin{pmatrix}
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0
\end{pmatrix} v = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$u = \begin{pmatrix} 2 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 2 \end{pmatrix} \in \mathbb{Z}^{2n}$$

Step 1 computing the syndrome: $O(n^2)$

Step 2 computing the unsatisfied parity checks: $O(n^2)$

• Bochum strategy: compute u_0 , flip v_0 , compute u_1 , flip u_1 , etc.

$$f,g\in \mathbb{F}_2[x]/\big(x^n-1\big)$$

$$f,g \in \mathbb{F}_{2}[x]/(x^{n}-1)$$

$$\downarrow$$

$$\begin{pmatrix} f_{0} & f_{n-1} & \dots & f_{1} & g_{0} & g_{n-1} & \dots & g_{1} \\ f_{1} & f_{0} & \dots & f_{2} & g_{1} & g_{0} & \dots & g_{2} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ f_{n-1} & f_{n-2} & \dots & f_{0} & g_{n-1} & g_{n-2} & \dots & g_{0} \end{pmatrix} \begin{pmatrix} v_{0} \\ v_{1} \\ \vdots \\ v_{2n-1} \end{pmatrix} = s$$

$$f,g \in \mathbb{F}_{2}[x]/(x^{n}-1)$$

$$\downarrow$$

$$\begin{pmatrix} f_{0} & f_{n-1} & \dots & f_{1} & g_{0} & g_{n-1} & \dots & g_{1} \\ f_{1} & f_{0} & \dots & f_{2} & g_{1} & g_{0} & \dots & g_{2} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ f_{n-1} & f_{n-2} & \dots & f_{0} & g_{n-1} & g_{n-2} & \dots & g_{0} \end{pmatrix} \begin{pmatrix} v_{0} \\ v_{1} \\ \vdots \\ v_{2n-1} \end{pmatrix} = s$$

$$\downarrow$$

$$\begin{pmatrix} f & xf & \dots & x^{n-1}f & g & xg & \dots & x^{n-1}g \end{pmatrix} \begin{pmatrix} v_{0} \\ v_{1} \\ \vdots \\ v_{2n-1} \end{pmatrix} = s$$

$$f,g \in \mathbb{F}_{2}[x]/(x^{n}-1)$$

$$\downarrow$$

$$\begin{pmatrix} f_{0} & f_{n-1} & \dots & f_{1} & g_{0} & g_{n-1} & \dots & g_{1} \\ f_{1} & f_{0} & \dots & f_{2} & g_{1} & g_{0} & \dots & g_{2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ f_{n-1} & f_{n-2} & \dots & f_{0} & g_{n-1} & g_{n-2} & \dots & g_{0} \end{pmatrix} \begin{pmatrix} v_{0} \\ v_{1} \\ \vdots \\ v_{2n-1} \end{pmatrix} = s$$

$$\downarrow$$

$$f = xf \quad \dots \quad x^{n-1}f \quad g \quad xg \quad \dots \quad x^{n-1}g \end{pmatrix} \begin{pmatrix} v_{0} \\ v_{1} \\ \vdots \\ v_{2n-1} \end{pmatrix} = s$$

$$\downarrow$$

$$s = v^{(0)}f + v^{(1)}g \in \mathbb{F}_{2}[x]/(x^{n}-1)$$

Sparse-times-dense polynomial in
$$\mathbb{F}_2[x]/(x^n-1)$$

QcBits computes vf as

$$x^{i_1}v + x^{i_2}v + \cdots$$

• Each $x^i v$ is simply a rotation of v.

Sparse-times-dense polynomial in
$$\mathbb{F}_2[x]/(x^n-1)$$

QcBits computes vf as

$$x^{i_1}v + x^{i_2}v + \cdots$$

• Each $x^i v$ is simply a rotation of v.

Sparse-times-dense polynomial in
$$\mathbb{F}_2[x]/(x^n-1)$$

QcBits computes vf as

$$x^{i_1}v + x^{i_2}v + \cdots$$

- Each $x^i v$ is simply a rotation of v.
- Addition can be carried out using XOR instrctions.

Sparse-times-dense polynomial in $\mathbb{F}_2[x]/(x^n-1)$

QcBits computes vf as

$$x^{i_1}v + x^{i_2}v + \cdots$$

- Each $x^i v$ is simply a rotation of v.
- Addition can be carried out using XOR instrctions.
- Constant-time rotations?

Rotating by $i = (i_k i_{k-1} \dots i_0)_2$ bits:

ullet conditionally rotate by 2^k bits.

- conditionally rotate by 2^k bits.
- conditionally rotate by 2^{k-1} bits, and so on.

- conditionally rotate by 2^k bits.
- conditionally rotate by 2^{k-1} bits, and so on.
- Example for $i = 010011_2$ and polynomial

$$\left(x^{8} + x^{10} + x^{12} + x^{14}\right) + \left(x^{16} + x^{17} + x^{20} + x^{21}\right) + \left(x^{24} + x^{25} + x^{26} + x^{27}\right) + \left(x^{36} + x^{37} + x^{38} + x^{39}\right)$$

Rotating by $i = (i_k i_{k-1} \dots i_0)_2$ bits:

- conditionally rotate by 2^k bits.
- conditionally rotate by 2^{k-1} bits, and so on.
- Example for $i = 010011_2$ and polynomial

$$(x^8 + x^{10} + x^{12} + x^{14}) + (x^{16} + x^{17} + x^{20} + x^{21}) + (x^{24} + x^{25} + x^{26} + x^{27}) + (x^{36} + x^{37} + x^{38} + x^{39})$$

$$00000000_2 \quad 01010101_2 \quad 00110011_2 \quad 00001111_2 \quad 11110000_2$$

111100002

- conditionally rotate by 2^k bits.
- conditionally rotate by 2^{k-1} bits, and so on.
- Example for $i = 010011_2$ and polynomial

$$\left(x^{8} + x^{10} + x^{12} + x^{14}\right) + \left(x^{16} + x^{17} + x^{20} + x^{21}\right) + \left(x^{24} + x^{25} + x^{26} + x^{27}\right) + \left(x^{36} + x^{37} + x^{38} + x^{39}\right)$$

	000000002	010101012	001100112	000011112	111100002
010011 ₂	010101012	001100112	000011112	111100002	000000002

- conditionally rotate by 2^k bits.
- conditionally rotate by 2^{k-1} bits, and so on.
- Example for $i = 010011_2$ and polynomial

$$\left(x^{8} + x^{10} + x^{12} + x^{14}\right) + \left(x^{16} + x^{17} + x^{20} + x^{21}\right) + \left(x^{24} + x^{25} + x^{26} + x^{27}\right) + \left(x^{36} + x^{37} + x^{38} + x^{39}\right)$$

	00000000_2	01010101 ₂	00110011_2	00001111_2	111100002
010011 ₂	010101012	001100112	000011112	111100002	000000002
010011 ₂	00001111_2	11110000_2	00000000_2	01010101_2	00110011_2

- conditionally rotate by 2^k bits.
- conditionally rotate by 2^{k-1} bits, and so on.
- Example for $i = 010011_2$ and polynomial

$$\left(x^{8} + x^{10} + x^{12} + x^{14}\right) + \left(x^{16} + x^{17} + x^{20} + x^{21}\right) + \left(x^{24} + x^{25} + x^{26} + x^{27}\right) + \left(x^{36} + x^{37} + x^{38} + x^{39}\right)$$

	000000002	010101012	001100112	000011112	111100002
010011 ₂	010101012	001100112	000011112	111100002	000000002
010011 ₂	00001111_2	11110000_2	00000000_2	01010101_2	00110011_2
0100112	00110011_2	000011112	111100002	00000000_2	010101012

- conditionally rotate by 2^k bits.
- conditionally rotate by 2^{k-1} bits, and so on.
- Example for $i = 010011_2$ and polynomial

$$\left(x^{8} + x^{10} + x^{12} + x^{14}\right) + \left(x^{16} + x^{17} + x^{20} + x^{21}\right) + \left(x^{24} + x^{25} + x^{26} + x^{27}\right) + \left(x^{36} + x^{37} + x^{38} + x^{39}\right)$$

	000000002	01010101 ₂	00110011_2	00001111_2	11110000 ₂
0100112	010101012	001100112	000011112	111100002	000000002
010011 ₂	00001111_2	11110000_2	00000000_2	01010101_2	00110011_2
0100112	00110011_2	00001111_2	11110000_2	00000000_2	01010101_2
0100112	01100001_2	11111110_2	00000000_2	00001010_2	101001102

$$f,g \in \mathbb{Z}[x]/(x^n-1)$$

$$f,g \in \mathbb{Z}[x]/(x^{n}-1)$$

$$\downarrow$$

$$\begin{pmatrix} f_{0} & f_{1} & \dots & f_{n-1} & g_{0} & g_{1} & \dots & g_{n-1} \\ f_{n-1} & f_{0} & \dots & f_{n-2} & g_{n-1} & g_{0} & \dots & g_{n-2} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ f_{1} & f_{2} & \dots & f_{0} & g_{1} & g_{2} & \dots & g_{0} \end{pmatrix} v = \begin{pmatrix} s_{0} \\ s_{1} \\ \vdots \\ s_{n-1} \end{pmatrix}$$

$$f,g \in \mathbb{Z}[x]/(x^{n}-1)$$

$$\downarrow$$

$$\begin{pmatrix} f_{0} & f_{1} & \dots & f_{n-1} & g_{0} & g_{1} & \dots & g_{n-1} \\ f_{n-1} & f_{0} & \dots & f_{n-2} & g_{n-1} & g_{0} & \dots & g_{n-2} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ f_{1} & f_{2} & \dots & f_{0} & g_{1} & g_{2} & \dots & g_{0} \end{pmatrix} v = \begin{pmatrix} s_{0} \\ s_{1} \\ \vdots \\ s_{n-1} \end{pmatrix}$$

$$\downarrow$$

$$\begin{pmatrix} f & g \\ xf & xg \\ \vdots \\ x^{n-1}f & x^{n-1}g \end{pmatrix} v = \begin{pmatrix} s_{0} \\ s_{1} \\ \vdots \\ s_{n-1} \end{pmatrix}$$

$$f,g \in \mathbb{Z}[x]/(x^{n}-1)$$

$$\downarrow$$

$$\begin{pmatrix} f_{0} & f_{1} & \dots & f_{n-1} & g_{0} & g_{1} & \dots & g_{n-1} \\ f_{n-1} & f_{0} & \dots & f_{n-2} & g_{n-1} & g_{0} & \dots & g_{n-2} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ f_{1} & f_{2} & \dots & f_{0} & g_{1} & g_{2} & \dots & g_{0} \end{pmatrix} v = \begin{pmatrix} s_{0} \\ s_{1} \\ \vdots \\ s_{n-1} \end{pmatrix}$$

$$\downarrow$$

$$\begin{pmatrix} f & g \\ xf & xg \\ \vdots \\ x^{n-1}f & x^{n-1}g \end{pmatrix} v = \begin{pmatrix} s_{0} \\ s_{1} \\ \vdots \\ s_{n-1} \end{pmatrix}$$

$$\downarrow$$

$$u = (sf, sg) \in \mathbb{Z}[x]/(x^{n}-1)$$

How to deal with decoding failures?

How to deal with decoding failures?

• QcBits for higher security levels?

How to deal with decoding failures?

- QcBits for higher security levels?
- better decoder?

How to deal with decoding failures?

- QcBits for higher security levels?
- better decoder?
- better parameters?

How to deal with decoding failures?

- · QcBits for higher security levels?
- better decoder?
- better parameters?

Is equal weight distribution ok?

How to deal with decoding failures?

- · QcBits for higher security levels?
- better decoder?
- better parameters?

Is equal weight distribution ok?

• at least it's close enough to the original proposal

How to deal with decoding failures?

- · QcBits for higher security levels?
- better decoder?
- better parameters?

Is equal weight distribution ok?

· at least it's close enough to the original proposal

More research is required to build up confidence.

www.win.tue.nl/~tchou/qcbits/