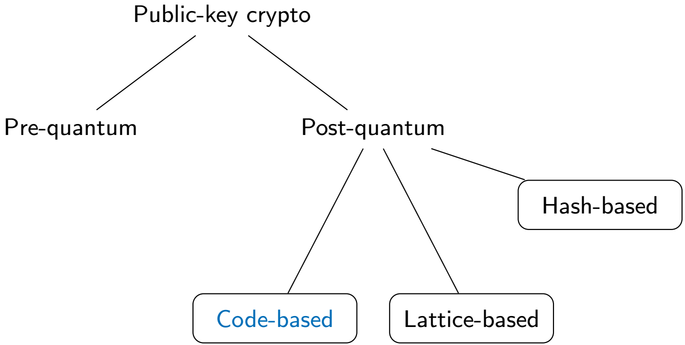
QcBits:
constant-time small-key code-based cryptography

Tung Chou

Technische Universiteit Eindhoven, The Netherlands

Code-based crypto



Coding theory

Code

- 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	c	$c + e$
Niederreiter	e	He

- General shape

McEliece/Niederreiter + **some code**

Binary Goppa versus QC-MDPC codes

1976	Public-key crypto
1977	RSA
1978	McEliece + Binary Goppa Codes
1996	NTRU encrypt
2013	QC-MDPC McEliece

Binary Goppa versus QC-MDPC codes

1976	Public-key crypto
1977	RSA
1978	McEliece + Binary Goppa Codes
	($\approx$ 100-KB Niederrieter keys)
1996	NTRU encrypt
2013	QC-MDPC McEliece

($\approx$ 1-KB Niederrieter keys)

QcBits

2013 — QC-MDPC McEliece

QcBits

2013	QC-MDPC McEliece
	on FPGAs (CHES)
2014	on FPGAs, microcontrollers (PQCrypto, DATE)
2015	on Haswell CPUs (ACM-TECS)
2016	on microcontrollers (PQCrypto)

QcBits

2013	•	QC-MDPC McEliece
	•	on FPGAs (CHES)
2014	•	on FPGAs, microcontrollers (PQCrypto, DATE)
2015	•	on Haswell CPUs (ACM-TECS)
2016	•	on microcontrollers (PQCrypto)
	•	**QcBits (new)**

QcBits

2013	QC-MDPC McEliece
	on FPGAs (CHES)
2014	on FPGAs, microcontrollers (PQCrypto, DATE)
2015	on Haswell CPUs (ACM-TECS)
2016	on microcontrollers (PQCrypto)
	QcBits (new)

The problem is timing attacks.

QcBits

2013		QC-MDPC McEliece
		on FPGAs (CHES)
2014		on FPGAs, microcontrollers (PQCrypto, DATE)
2015		on Haswell CPUs (ACM-TECS)
2016		on microcontrollers (PQCrypto)
		QcBits (new)

The problem is timing attacks.

QcBits

2013	QC-MDPC McEliece
	on FPGAs (CHES)
2014	on FPGAs, microcontrollers (PQCrypto, DATE)
2015	on Haswell CPUs (ACM-TECS)
2016	on microcontrollers (PQCrypto)
	QcBits (new)

The problem is timing attacks.

- PQCrypto 2014: constant-time operations assuming no caches

QcBits

2013	QC-MDPC McEliece
	on FPGAs (CHES)
2014	on FPGAs, microcontrollers (PQCrypto, DATE)
2015	on Haswell CPUs (ACM-TECS)
2016	on microcontrollers (PQCrypto)
	QcBits (new)

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.

QC-MDPC codes

QC-MDPC codes

- MDPC: moderate-density-parity-check

QC-MDPC codes

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

QC-MDPC codes

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

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

QC-MDPC codes

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

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

Statistical decoding (bit flipping)

Start with $v = c + e$.

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

Statistical decoding (bit flipping)

Start with $v = c + e$.

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

Statistical decoding (bit flipping)

Start with $v = c + e$.

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

$$+)\quad \overline{}$$

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

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

Statistical decoding (bit flipping)

Start with $v = c + e$.

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

$$
+)\;\overline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}
$$

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

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

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

Syndrome computation: polynomial view

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

Syndrome computation: polynomial view

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

$$\downarrow$$

$$\begin{pmatrix} f_0 & f_{n-1} & \cdots & f_1 & g_0 & g_{n-1} & \cdots & g_1 \\ f_1 & f_0 & \cdots & f_2 & g_1 & g_0 & \cdots & g_2 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ f_{n-1} & f_{n-2} & \cdots & f_0 & g_{n-1} & g_{n-2} & \cdots & g_0 \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{2n-1} \end{pmatrix} = s$$

Syndrome computation: polynomial view

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

$$\downarrow$$

$$\begin{pmatrix} f_0 & f_{n-1} & \cdots & f_1 & g_0 & g_{n-1} & \cdots & g_1 \\ f_1 & f_0 & \cdots & f_2 & g_1 & g_0 & \cdots & g_2 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ f_{n-1} & f_{n-2} & \cdots & f_0 & g_{n-1} & g_{n-2} & \cdots & g_0 \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{2n-1} \end{pmatrix} = s$$

$$\downarrow$$

$$\begin{pmatrix} f & xf & \cdots & x^{n-1}f & g & xg & \cdots & x^{n-1}g \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{2n-1} \end{pmatrix} = s$$

Syndrome computation: polynomial view

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

$$\downarrow$$

$$\begin{pmatrix} f_0 & f_{n-1} & \cdots & f_1 & g_0 & g_{n-1} & \cdots & g_1 \\ f_1 & f_0 & \cdots & f_2 & g_1 & g_0 & \cdots & g_2 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ f_{n-1} & f_{n-2} & \cdots & f_0 & g_{n-1} & g_{n-2} & \cdots & g_0 \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{2n-1} \end{pmatrix} = s$$

$$\downarrow$$

$$\begin{pmatrix} f & xf & \cdots & x^{n-1}f & g & xg & \cdots & x^{n-1}g \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{2n-1} \end{pmatrix} = s$$

$$\downarrow$$

$$s = v^{(0)} f + v^{(1)} g \in \mathbb{F}_2[x]/(x^n - 1)$$

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

Compute $vf \in \mathbb{F}_2[x]/(x^n - 1)$

- v dense: array of 32/64-bit words
- f sparse: $i_1, i_2, \ldots$, where $f_i = 1$

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

Compute $vf \in \mathbb{F}_2[x]/(x^n - 1)$

- v dense: array of 32/64-bit words
- f sparse: $i_1, i_2, \ldots$, where $f_i = 1$

QcBits computes vf as

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

- Each $x^i v$ is simply a rotation of v.
- Summation can be carried out by XOR instructions.
- Constant-time rotations?

Constant-time rotations (barrel shifter)

Task: compute $v \ggg s$, where $s = (s_1 s_2 s_3 \ldots s_\ell)_2$.

v

Constant-time rotations (barrel shifter)

Task: compute $v \ggg s$, where $s = (s_1 s_2 s_3 \ldots s_\ell)_2$.

$$
\begin{tikzcd}
v \arrow[r] \arrow[d, "s_1 = 0"'] & v \ggg 2^{\ell-1} \arrow[dl, "s_1 = 1"] \\
v' &
\end{tikzcd}
$$

Constant-time rotations (barrel shifter)

Task: compute $v \gg s$, where $s = (s_1 s_2 s_3 \ldots s_\ell)_2$.

$$
\begin{tikzcd}
v \arrow[r] \arrow[d, "s_1 = 0"'] & v \gg 2^{\ell-1} \arrow[dl, "s_1 = 1"'] \\
v' \arrow[r] \arrow[d, "s_2 = 0"'] & v' \gg 2^{\ell-2} \arrow[dl, "s_2 = 1"'] \\
v'' &
\end{tikzcd}
$$

Constant-time rotations (barrel shifter)

Task: compute $v \ggg s$, where $s = (s_1 s_2 s_3 \dots s_\ell)_2$.

$$
\begin{tikzcd}
v \arrow[r] \arrow[d, "s_1=0"'] & v \ggg 2^{\ell-1} \arrow[dl, "s_1=1"'] \\
v' \arrow[r] \arrow[d, "s_2=0"'] & v' \ggg 2^{\ell-2} \arrow[dl, "s_2=1"'] \\
v'' \arrow[r] \arrow[d, "s_3=0"'] & v'' \ggg 2^{\ell-3} \arrow[dl, "s_3=1"'] \\
\vdots &
\end{tikzcd}
$$

Computing u: polynomial view

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

Computing u: polynomial view

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

$$\downarrow$$

$$\begin{pmatrix} f_0 & f_1 & \dots & f_{n-1} & g_0 & g_1 & \cdots & g_{n-1} \\ f_{n-1} & f_0 & \dots & f_{n-2} & g_{n-1} & g_0 & \cdots & g_{n-2} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ f_1 & f_2 & \dots & f_0 & g_1 & g_2 & \cdots & g_0 \end{pmatrix} v = \begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_{n-1} \end{pmatrix}$$

Computing u: polynomial view

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

$$\downarrow$$

$$\begin{pmatrix} f_0 & f_1 & \cdots & f_{n-1} & g_0 & g_1 & \cdots & g_{n-1} \\ f_{n-1} & f_0 & \cdots & f_{n-2} & g_{n-1} & g_0 & \cdots & g_{n-2} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ f_1 & f_2 & \cdots & f_0 & g_1 & g_2 & \cdots & g_0 \end{pmatrix} v = \begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_{n-1} \end{pmatrix}$$

$$\downarrow$$

$$\begin{pmatrix} f & g \\ xf & xg \\ \vdots & \\ x^{n-1} f & x^{n-1} g \end{pmatrix} v = \begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_{n-1} \end{pmatrix}$$

Computing u: polynomial view

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

$$\downarrow$$

$$\begin{pmatrix} f_0 & f_1 & \cdots & f_{n-1} & g_0 & g_1 & \cdots & g_{n-1} \\ f_{n-1} & f_0 & \cdots & f_{n-2} & g_{n-1} & g_0 & \cdots & g_{n-2} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ f_1 & f_2 & \cdots & f_0 & g_1 & g_2 & \cdots & g_0 \end{pmatrix} v = \begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_{n-1} \end{pmatrix}$$

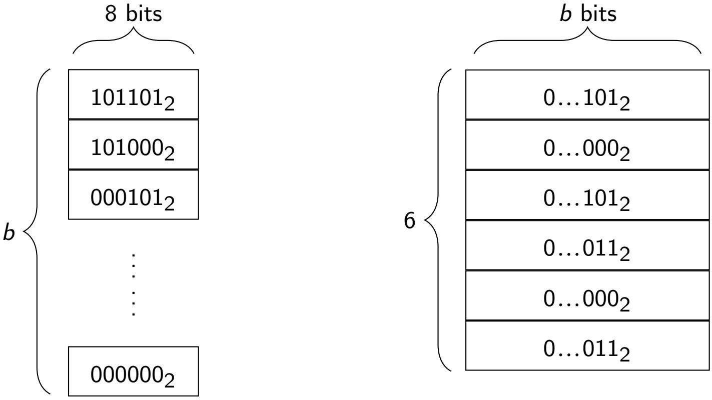
$$\downarrow$$

$$\begin{pmatrix} f & g \\ xf & xg \\ \vdots & \\ x^{n-1}f & x^{n-1}g \end{pmatrix} v = \begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_{n-1} \end{pmatrix}$$

$$\downarrow$$

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

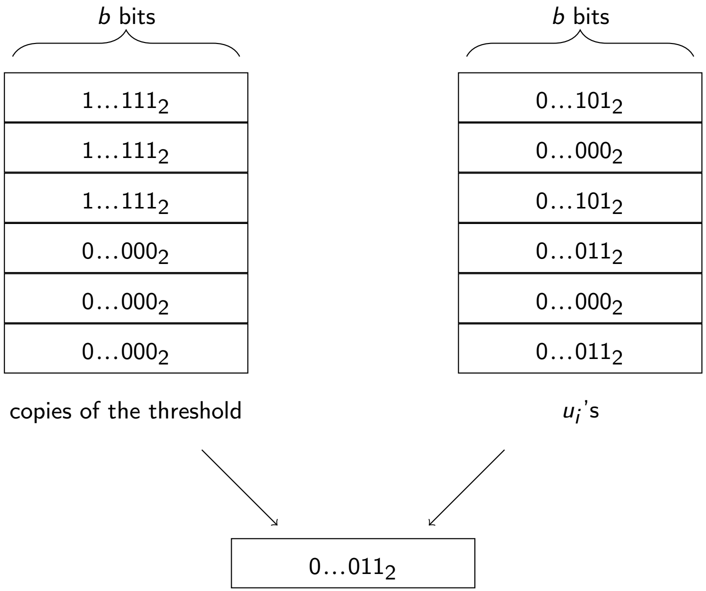
Accumulating $x^i s$



Non-bitsliced

Bitsliced

Flipping bits



Failure rate

Problem

- Can the adversary exploit decoding failures?
- Reviewer comment: "...the number of iterations in the decoding step is fixed to a value which is heuristically judged high enough..."
- QcBits: 10^{-8} for 2^{80} security; worse for higher security levels

Solution?

- Julia Chaulet, Nicolas Sendrier. "Worst case QC-MDPC decoder for Mceliece cryptosystem". ISIT 2016.

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