HQC: **H**amming **Q**uasi-**C**yclic

An IND-CCA2 Code-based Public Key Encryption Scheme

June the 8th, 2021

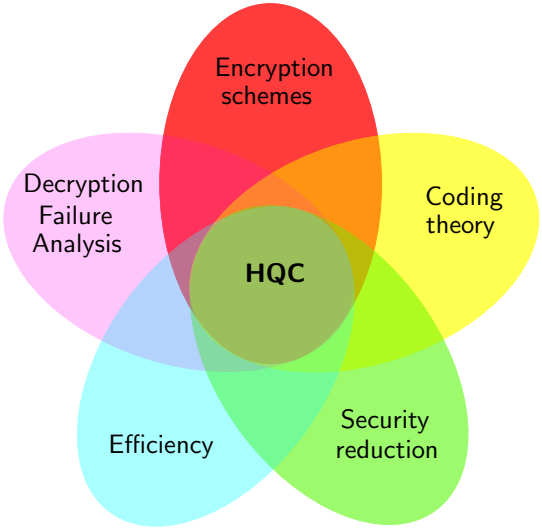
NIST 3RD PQC Standardization Conference

https://pqc-hqc.org

C. Aguilar Melchor	ISAE-Supaéro, Univ. Toulouse	A. Dion	ISAE-Supaéro, Univ. Toulouse
N. Aragon	University of Limoges	**P. Gaborit**	University of Limoges
S. Bettaieb	Worldline	J. Lacan	ISAE-Supaéro, Univ. Toulouse
L. Bidoux	Worldline	E. Persichetti	Florida Atlantic University
O. Blazy	University of Limoges	J.-M. Robert	University of Toulon
J. Bos	Wordline	P. Véron	University of Toulon
J.-C. Deneuville	ENAC, University of Toulouse	G. Zémor	IMB, University of Bordeaux

Outline

HQC Classification / Design Rationale



Important features:

- IND-CPA code-based PKE
- Reduction to a well-known and difficult problem:

  Decoding random quasi-cyclic codes
- No hidden trap in the code
- Efficient decoding
- Precise DFR analysis

HQC Encryption Scheme

Encryption scheme in **H**amming metric, using **Q**uasi-**C**yclic Codes

- Notation: Secret data - Public data - One-time Randomness
- $\mathbf{G}$ is the generator matrix of some public code $\mathcal{C}$
- $\mathcal{S}_w^n(\mathbb{F}_2) = \{\mathbf{x} \in \mathbb{F}_2^n \text{ such that } \omega(\mathbf{x}) = w\}$

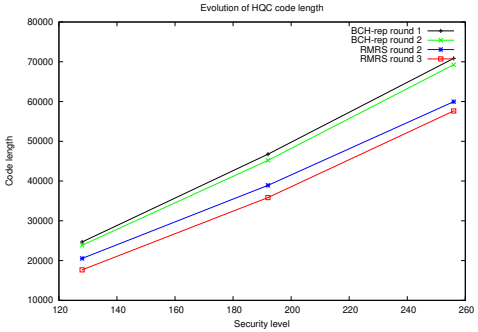
Alice		Bob
$\mathbf{seed_h} \xleftarrow{\$} \{0,1\}^\lambda$, $\mathbf{h} \xleftarrow{\mathbf{seed_h}} \mathbb{F}_2^n$ $\mathbf{x}, \mathbf{y} \xleftarrow{\$} \mathcal{S}_w^n(\mathbb{F}_2)$, $\mathbf{s} \leftarrow \mathbf{x} + \mathbf{hy}$	$\xrightarrow{\mathbf{seed_h}, \mathbf{s}}$	$\mathbf{r}_1, \mathbf{r}_2 \xleftarrow{\$} \mathcal{S}_w^n(\mathbb{F}_2)$, $\mathbf{e} \xleftarrow{\$} \mathcal{S}_w^n(\mathbb{F}_2)$ $\mathbf{u} \leftarrow \mathbf{r}_1 + \mathbf{hr}_2, \quad \mathbf{v} \leftarrow \mathbf{mG} + \mathbf{sr}_2 + \mathbf{e}$
$\mathbf{m} \leftarrow \mathcal{C}.\text{Decode}(\mathbf{v} - \mathbf{uy})$	$\xleftarrow{\mathbf{u}, \mathbf{v}}$	

3rd round tweaks

◇ We provided a **better decryption failure analysis** that allows to decrease the size of our public keys.

◇ We switched from the BCH-repetition decoder to a **concatenated Reed-Muller and Reed-Solomon (RMRS) decoder**.

◇ The **size of the decoded messages are set to the security level** (i.e dimension 128 instead of 256 for level 1), thus improving the decoding capability of the code.



3rd round parameters and timings

Sizes in **bytes**

	pk size	ct size	Improvement wrt. 2nd round
hqc-128	2,249	4,481	**28%**
hqc-192	4,522	9,026	**23%**
hqc-256	7,245	14,469	**18%**

Timings in **kilocycles**

	AVX2 Implementation			Improvement wrt. 2nd round		
	Keygen	Encaps	Decaps	Keygen	Encaps	Decaps
hqc-128	83	197	349	**59%**	**48%**	**30%**
hqc-192	200	456	740	**50%**	**40%**	**24%**
hqc-256	400	887	1478	**38%**	**29%**	**8%**

Hardware implementation

◇ We now only use **KECCAK-based random oracles** in order to limit software footprint.

◇ **HLS implementation** of HQC: C translated into VHDL by Xilinx tools.
- Easy to modify, good for quick tests.
- Compatible with the software KATs.
- Improvable VHDL by tweaking/replacing modules → there is room for improvement.

Hardware performances

Function	Frequency (MHz)	Slices	LUT	FF	BRAM	Cycles	Time (ms)
Keygen	150	3.9k	12k	9k	3	40k	0.27
Encaps	151	5.5k	16k	13k	5	89k	0.59
Decaps	152	6.2k	19k	15k	9	190k	1.2

◇ **Compact version**: 2 times smaller and 10 times slower.

Questions ?

HQC official website and updates:
https://pqc-hqc.org/