A timing attack against HQC

Thales Bandiera Paiva   Routo Terada

Institute of Mathematics and Statistics
University of Sao Paulo, Brazil

tpaiva@ime.usp.br
rt@ime.usp.br

2019-08-15

Motivation

HQC encryption scheme

- Code-based scheme relying on the quasi-cyclic decoding problem
- Candidate for NIST's Post-Quantum Crypto standardization process
- Offers advantages when compared to other code-based candidates:
  - Reasonably small keys (much smaller than the original McEliece)
  - It does not use a secret sparse structure (unlike MDPC or LDPC)
  - The submitters provide a detailed analysis of the failure probability

Timing attack

- We attack the reference implementation submitted to NIST
- The attack recovers the secret key with 400M decryption timings

The HQC encryption scheme [HQC]

Setup

- Fix an $[n, k]$-linear code $\mathcal{C}$ capable of correcting a large number of errors with overwhelming probability
- The authors propose $\mathcal{C}$ as the tensor product of a BCH code and a repetition code
- For 128 bits parameters $n = 22,229$ and $k = 256$

Code $\mathcal{C}$ is a public parameter and has the following operations

- $\mathbf{c} \leftarrow \mathrm{Encode}_{\mathcal{C}}(\mathbf{m}) = \mathrm{Encode}_{\mathrm{Rep}}(\mathrm{Encode}_{\mathrm{BCH}}(\mathbf{m}))$ adds redundancy to a message $\mathbf{m}$
- $\mathrm{Decode}_{\mathcal{C}}(\mathbf{c} + \mathbf{e}) = \mathrm{Decode}_{\mathrm{BCH}}(\mathrm{Decode}_{\mathrm{Rep}}(\mathbf{c} + \mathbf{e}))$ recovers $\mathbf{m}$ from the corrupted codeword, where $\mathbf{e}$ is a sparse vector

[HQC] Melchor, Carlos Aguilar, et al. "Hamming Quasi-Cyclic (HQC)."

The HQC encryption scheme

1. **Key Generation**
   - $\mathbf{h} \xleftarrow{\$} \mathbb{F}_2^n$
   - $\mathbf{x}, \mathbf{y} \xleftarrow{\$}$ sparse vectors from $\mathbb{F}_2^n$
   - $\mathbf{s} \leftarrow \mathbf{x} + \mathbf{y} \cdot \mathbf{h} = \mathbf{x} + \mathbf{y}\,\mathrm{rot}(\mathbf{h})$
   - $K_{\mathsf{Pub}} = (\mathbf{s}, \mathbf{h})$ and $K_{\mathsf{Sec}} = (\mathbf{x}, \mathbf{y})$
2. **Encrypting a message $\mathbf{m} \in \mathbb{F}_2^k$**
   - $\mathbf{r}_1, \mathbf{r}_2, \mathbf{e} \xleftarrow{\$}$ sparse vectors from $\mathbb{F}_2^n$
   - $\mathbf{u} \leftarrow \mathbf{r}_1 + \mathbf{r}_2 \cdot \mathbf{h}$ (advice)
   - $\mathbf{v} \leftarrow \mathrm{Encode}_{\mathcal{C}}(\mathbf{m}) + \mathbf{s} \cdot \mathbf{r}_2 + \mathbf{e}$ (very corrupted codeword)
   - Return $\mathbf{c} \leftarrow (\mathbf{u}, \mathbf{v})$
3. **Decrypting a ciphertext $\mathbf{c} = (\mathbf{u}, \mathbf{v})$**
   - $\mathbf{c}' \leftarrow \mathbf{v} + \mathbf{u} \cdot \mathbf{y} = \mathrm{Encode}_{\mathcal{C}}(\mathbf{m}) + \mathbf{x} \cdot \mathbf{r}_2 + \mathbf{r}_1 \cdot \mathbf{y} + \mathbf{e}$
   - $\hat{\mathbf{m}} \leftarrow \mathrm{Decode}_{\mathcal{C}}(\mathbf{c}') = \mathrm{Decode}_{\mathcal{C}}(\mathrm{Encode}_{\mathcal{C}}(\mathbf{m}) + \mathbf{e}')$

Notation

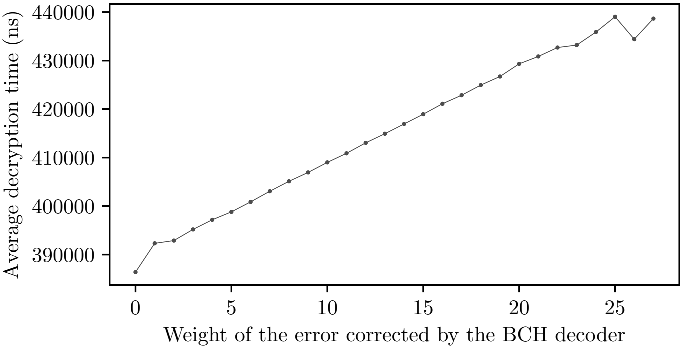
$$\mathrm{rot}(\mathbf{h}) = \begin{bmatrix} h_0 & h_1 & \dots & h_{n-1} \\ h_{n-1} & h_0 & \dots & h_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ h_1 & h_2 & \dots & h_0 \end{bmatrix}$$

Information leakage

- Let $\mathbf{e}' = \mathbf{x} \cdot \mathbf{r}_2 + \mathbf{y} \cdot \mathbf{r}_1 + \mathbf{e}$
- Consider the decoding procedure used for decryption

$$\mathsf{Decode}_{\mathcal{C}}(\mathbf{c}') = \mathsf{Decode}_{\mathsf{BCH}}(\mathsf{Decode}_{\mathsf{Rep}}(\mathsf{Encode}_{\mathcal{C}}(\mathbf{m}) + \mathbf{e}'))$$

- $\mathsf{Decode}_{\mathsf{BCH}}$ is not constant time $\Rightarrow$ The weight of the error vector left by $\mathsf{Decode}_{\mathsf{Rep}}(\mathsf{Encode}_{\mathcal{C}}(\mathbf{m}) + \mathbf{e}')$ is **leaked**



Repetition decoding errors

- Consider a repetition block size of $n_2 = 5$

$$\begin{aligned} \mathbf{m} &= [0 \quad 1 \quad 0] \\ \mathbf{c} \leftarrow \text{Encode}_{\text{Rep}}(\mathbf{m}) &= [00000 \quad 11111 \quad 00000] \\ \mathbf{e}' &= [10010 \quad 00001 \quad 00111] \\ \mathbf{c} + \mathbf{e}' &= [10010 \quad 11110 \quad 00111] \\ \text{Decode}_{\text{Rep}}(\mathbf{c} + \mathbf{e}') &= [0 \quad 1 \quad 1] \end{aligned}$$

There are less decoding errors when

- $\mathrm{w}(\mathbf{e}')$ is lower
- Few pairs of 1's separated by less than n_2 positions

Repetition decoding errors and **spectrums**

Definition (Spectrum)

The spectrum of $\mathbf{v}$, denoted by $\sigma(\mathbf{v})$ is the set of cyclic distances between its non-null entries, together with their multiplicities.

Example

- $\mathbf{v} = [1\,0\,1\,0\,0\,0\,0\,1\,0] \Rightarrow \sigma(\mathbf{v}) = \{\mathbf{2} : 2,\ \mathbf{4} : 1\}$

Lower probability of repetition decoding errors when

- $\sigma(\mathbf{e}')$ does not have too many entries
- Small cyclic distances in $\sigma(\mathbf{e}')$ appear with lower multiplicity

It is possible to reconstruct a sparse vector from its spectrum

Connecting the dots

Recall $\mathbf{e}' = \mathbf{x} \cdot \mathbf{r}_2 + \mathbf{y} \cdot \mathbf{r}_1 + \mathbf{e}$

What we want

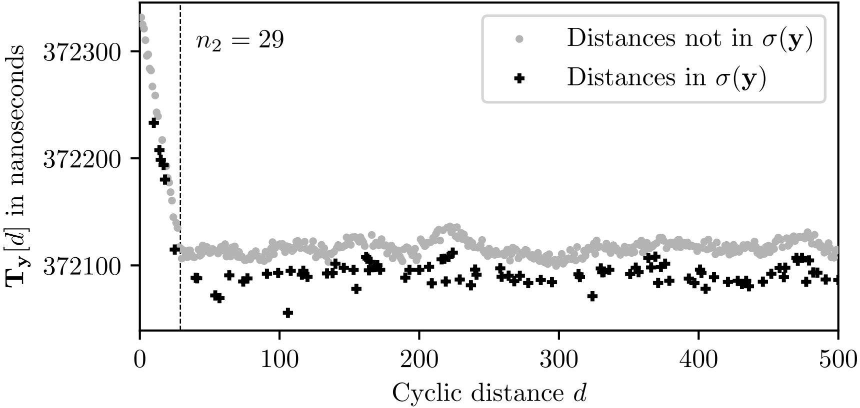
Decryption time $\rightarrow$ Weight of error $\mathbf{e}'$ $\rightarrow$ Spectrum of $\mathbf{y}$ $\rightarrow$ Build $\mathbf{y}$ from $\sigma(\mathbf{y})$ $\rightarrow$ Compute $\mathbf{x} = \mathbf{s} + \mathbf{y} \cdot \mathbf{h}$

What is missing

- Show how $\sigma(\mathbf{y} \cdot \mathbf{r}_1)$ relates to $\sigma(\mathbf{y})$ and $\sigma(\mathbf{r}_1)$

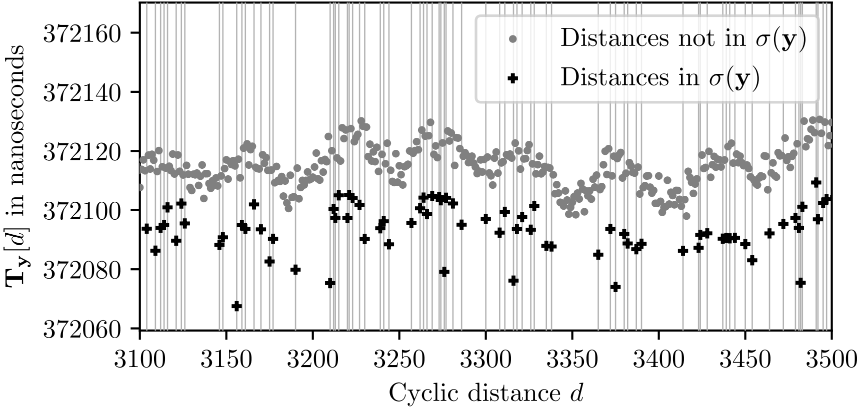
The timing attack - **128 bits security parameters**

- Consider 1 billion decoding challenges generated at random
- For each challenge record $\sigma(\mathbf{r}_1)$ and the decryption time
- $\mathbf{T_y}[d] \leftarrow$ average decryption time for the challenges in which $d \in \sigma(\mathbf{r}_1)$



Decryption time and the spectrums of $\mathbf{r}_1$ and $\mathbf{y}$ (zoom)

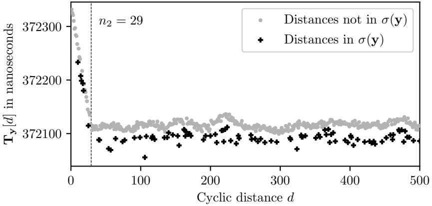
- Consider 1 billion decoding challenges generated at random
- For each challenge record $\sigma(\mathbf{r}_1)$ and the decryption time
- $\mathbf{T_y}[d] \leftarrow$ average decryption time for the challenges in which $d \in \sigma(\mathbf{r}_1)$

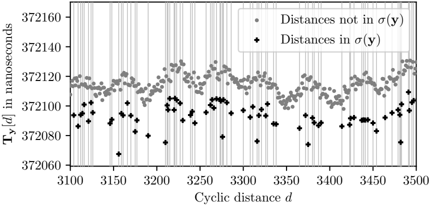


Three empirical observations

Let d be a distance in $\sigma(\mathbf{r}_1)$

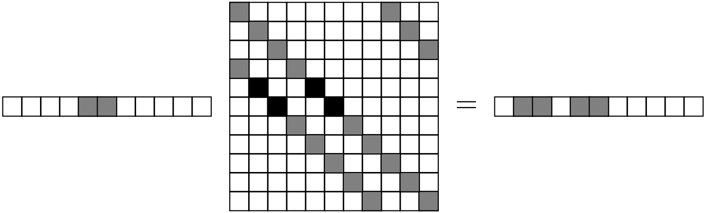
1. If d is lower than n_2, it causes slower decryption
2. If d is also in $\sigma(\mathbf{y})$, it causes faster decryption **than its neighbors which are not in $\sigma(\mathbf{y})$**
3. If d has a large number of neighbors in $\sigma(\mathbf{y})$, it causes slower decryption





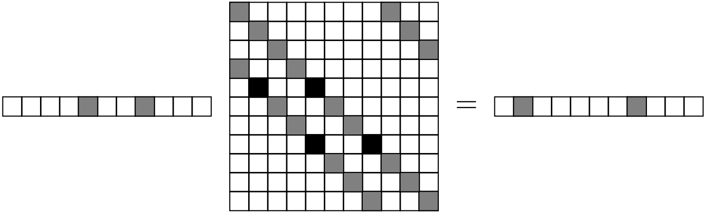
Intuition

1. d lower than n_2 causes slower decryption
   - Analyzing the product $\mathbf{r}_1 \cdot \mathbf{y}$ we get



Intuition

2. d also in $\sigma(\mathbf{y})$ causes faster decryption
   - Analyzing the product $\mathbf{r}_1 \cdot \mathbf{y}$ we get

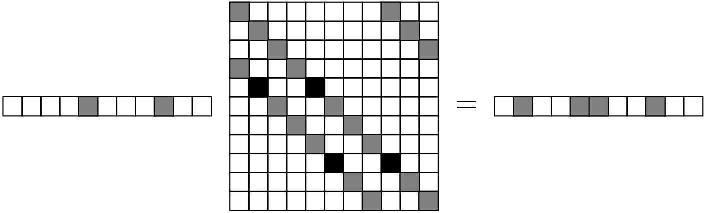


This observation was used for the reaction attack on QC-MDPC [GJS16]

[GJS16] Guo, Qian, Thomas Johansson, and Paul Stankovski. "A key recovery attack on MDPC with CCA security using decoding errors." Asiacrypt 2016

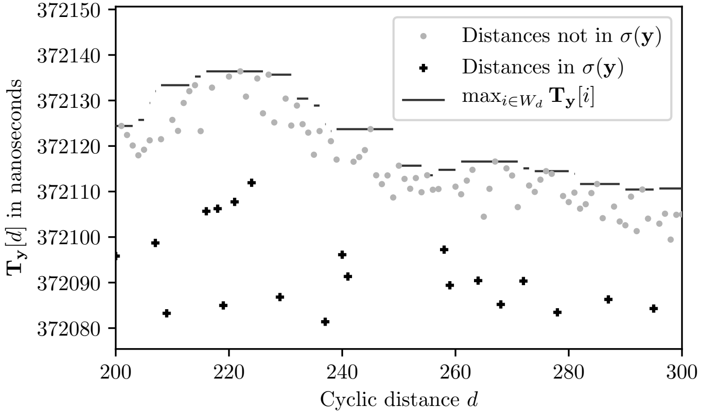
Intuition

3. d with neighbors in $\sigma(\mathbf{y})$ causes slower decryption
   - Analyzing the product $\mathbf{r}_1 \cdot \mathbf{y}$ we get



Clustering procedure

- **Input:** $\mathbf{T_y}$ and N
- **Output:** D = a set of N distances that it thinks that are outside $\sigma(\mathbf{y})$



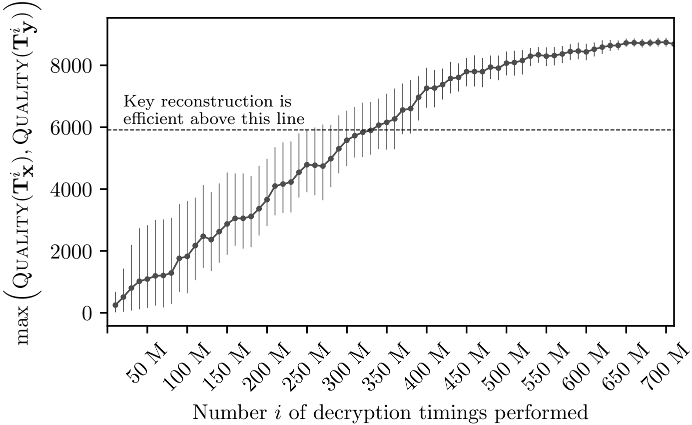
Performance of the key reconstruction algorithm

- 128 bits security parameters
- Varying sizes of D, the set of distances outside the spectrum
- Tested on an Intel i7 8700 CPU with 12 hyperthreads

$\lvert D \rvert$	Fraction of known distances outside the spectrum	**Randomized GJS** Median of the CPU time (s)	**Original GJS** Median of the CPU time (s)
9104	100%	0.51	0.98
8648	95%	0.51	10.78
8192	90%	0.50	772.64
7736	85%	0.75	6801.10
7280	80%	1.96	-
6824	75%	10.02	-
6368	70%	75.63	-
5912	65%	2767.90	-
5456	60%	-	-

Number of decryption challenges

- We used a simple clustering algorithm to get sets of (only) distances outside the spectrum of $\mathbf{y}$
- Quality($\mathbf{T}^i_{\mathbf{y}}$) denotes the number of distances outside the spectrum with can be successfully distinguished using $\mathbf{T}^i_{\mathbf{y}}$
- The key can be efficiently recovered when the Quality is above 5912 (65%)



Countermeasures

Patch the scheme

- Add some errors back after $\mathsf{Decode}_{\mathsf{Rep}}$
- Needs a careful statistical analysis
- Can make BCH decoding time independent of the secret key

Use other code $\mathcal{C}$ which admits constant-time decoding

- May not be easy to guarantee negligible error probabilities
- This is of independent interest since may lead to smaller keys

Use constant-time BCH decoders [WR19]

- The first constant-time BCH decoder appeared only months ago
- Can be up to 3 times slower
- Security against power side-channels was not yet considered

[WR19] `https://eprint.iacr.org/2019/155`

Conclusion

- We presented the first timing attack on HQC
- The attack is validated against 128 bits CCA secure version of HQC
- This is ~~almost~~ not the first time BCH decoding was targeted [DTV+19]
- Constant-time BCH decoders are the main countermeasure
  - But they are very recent and come with a performance cost

[DTV+19] https://eprint.iacr.org/2019/292.pdf