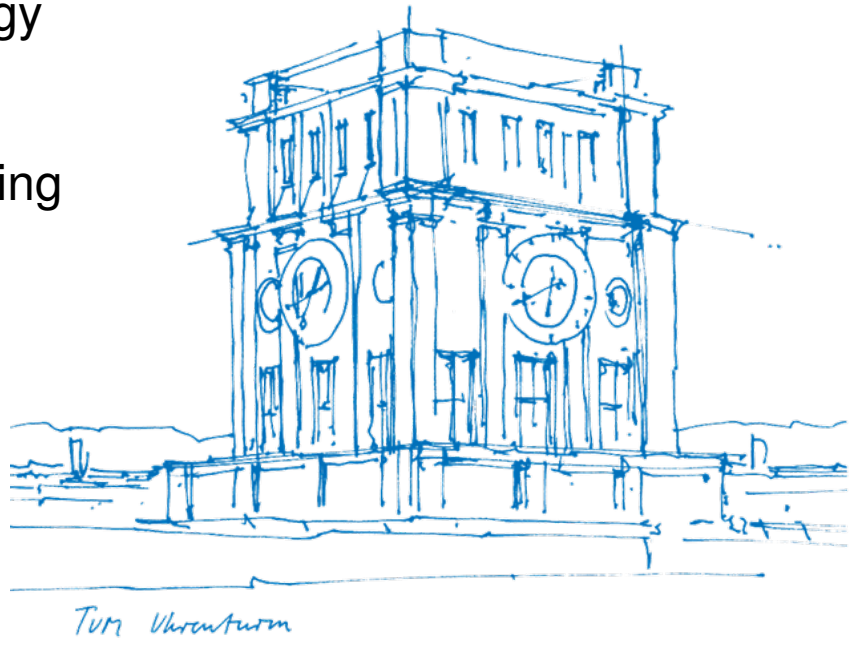
A Power Side-Channel Attack on the CCA2-Secure HQC KEM

Thomas Schamberger[1] Julian Renner[2] Georg Sigl[1] Antonia Wachter-Zeh[2]

[1]Technical University of Munich
Faculty of Electrical and Computer Engineering
Institute for Security in Information Technology

[2]Technical University of Munich
Faculty of Electrical and Computer Engineering
Institute for Communications Engineering

19.11.2020



CARDIS 2020

Outline

Motivation

- Large-scale quantum computers have strong impact on public key cryptography

  $\Rightarrow$ NIST Post-Quantum Cryptography Standardization Process

- "NIST hopes [...] this review period will include more work on **side-channel resistant implementations** [...]." [1]

[1] NIST: *Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process*

Hamming Quasi Cyclic (HQC)

- CCA2-secure KEM in the third round of the NIST contest
- Code-based cryptosystem
- Advantages of the scheme:
  - Security does not rely on hiding the structure of the used error-correcting code (in contract to e.g. "Classic McEliece").
    $\Rightarrow$ Quasi-Cyclic Syndrome Decoding problem
- Due to the cyclic structure elements are represented in the ring $\mathcal{R} := \mathbb{F}_2[X]/(X^n - 1)$

  $\Rightarrow$ Representation as binary values of n-bit size

$$x^7 + x^5 + x^4 + x^1 + 1 \; (n = 8)$$

$$1\;0\;1\;1\;0\;0\;1\;1$$

HQC - PKE

Encryption/Decryption

Algorithm 1: Encryption

Input: pk $= (\boldsymbol{h}, \boldsymbol{s})$, pt $= (\boldsymbol{m})$ and randomness θ
Output: ct $= (\boldsymbol{u}, \boldsymbol{v})$

1 $\boldsymbol{e}' \xleftarrow{\$} \mathcal{R}$ such that $\mathrm{HW}(\boldsymbol{e}') = \omega_e$ using θ
2 $(\boldsymbol{r}_1, \boldsymbol{r}_2) \xleftarrow{\$} \mathcal{R}^2$ such that $\mathrm{HW}(\boldsymbol{r}_1) = \mathrm{HW}(\boldsymbol{r}_2) = \omega_r$ using θ
3 $\boldsymbol{u} \leftarrow \boldsymbol{r}_1 + \boldsymbol{h}\boldsymbol{r}_2$
4 $\boldsymbol{v} \leftarrow \mathbf{Encode}(\boldsymbol{m}) + \boldsymbol{s}\boldsymbol{r}_2 + \boldsymbol{e}'$
5 **return** ct $= (\boldsymbol{u}, \boldsymbol{v})$

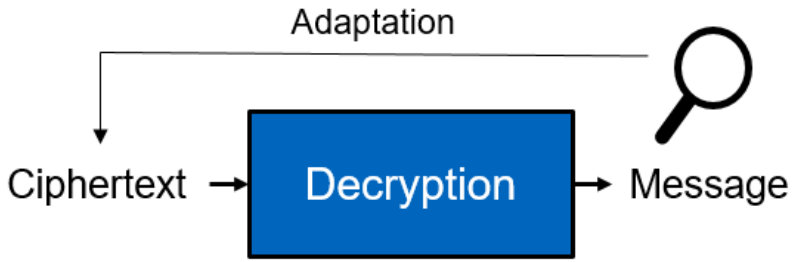
Algorithm 2: Decryption

Input: sk $= (\boldsymbol{x}, \boldsymbol{y})$, ct $= (\boldsymbol{u}, \boldsymbol{v})$
Output: $\boldsymbol{m}$

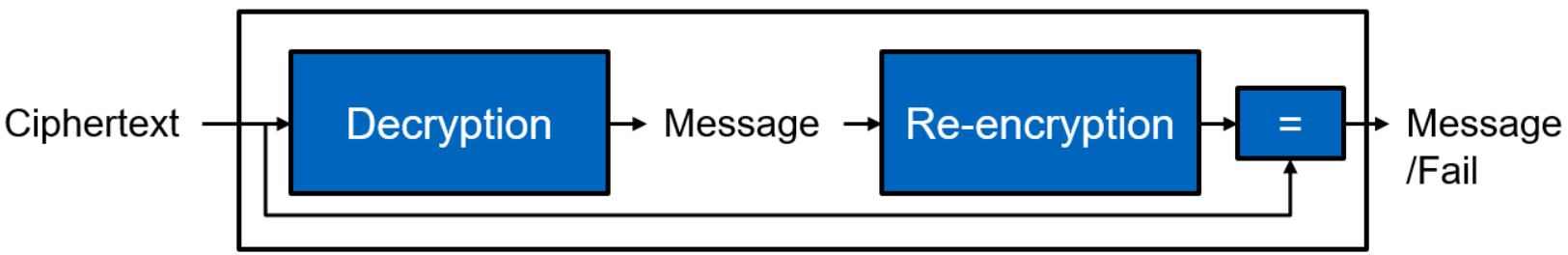
1 $\boldsymbol{v}' \leftarrow \boldsymbol{v} - \boldsymbol{u}\boldsymbol{y}$
2 $\boldsymbol{m} \leftarrow \mathbf{Decode}(\boldsymbol{v}')$
3 **return** $\boldsymbol{m}$

HQC - KEM

- The PKE version is vulnerable against chosen-ciphertext attacks [2]



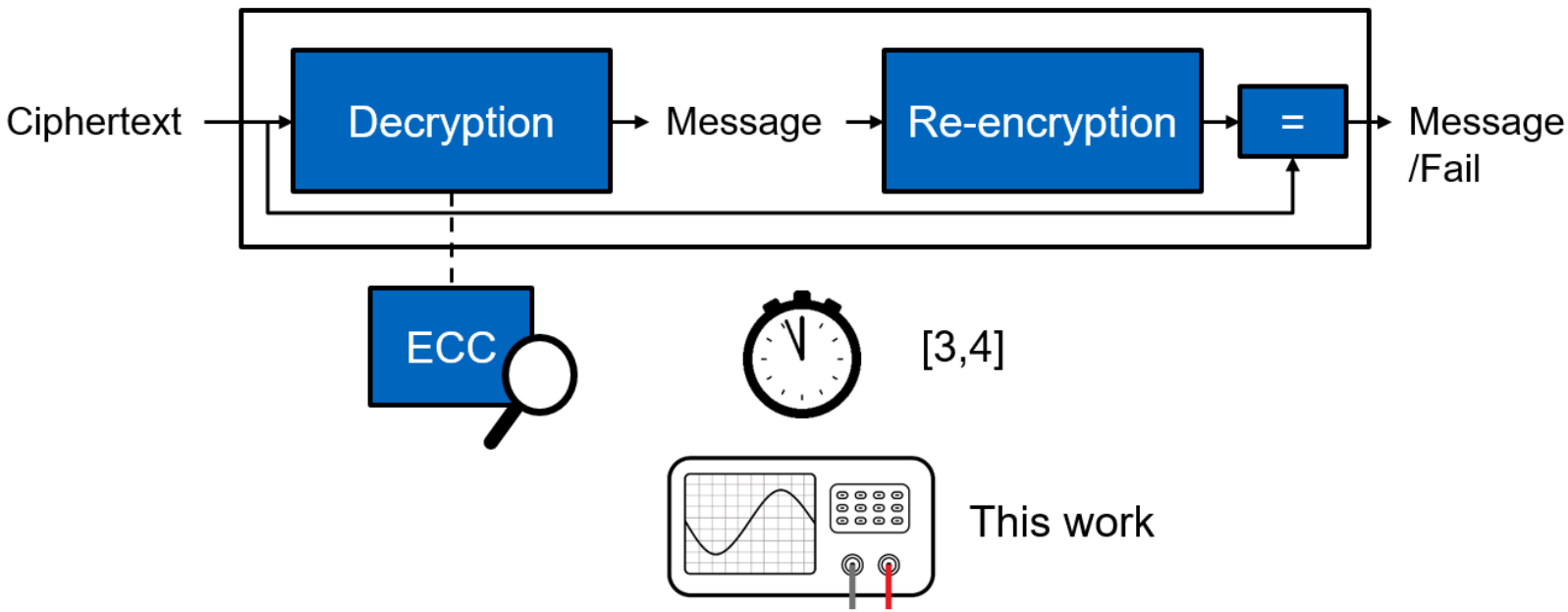
- HQC uses a variant of the Fujisaki-Okamoto transformation to achieve a CCA2-secure KEM



[2] Huguenin-Dumittan et al.: *Classical Misuse Attacks on NIST Round2 PQC: The Power of Rank-Based Schemes*

Attacks against HQC

- Published attacks use a timing side-channel in the used error correction of HQC [3,4]



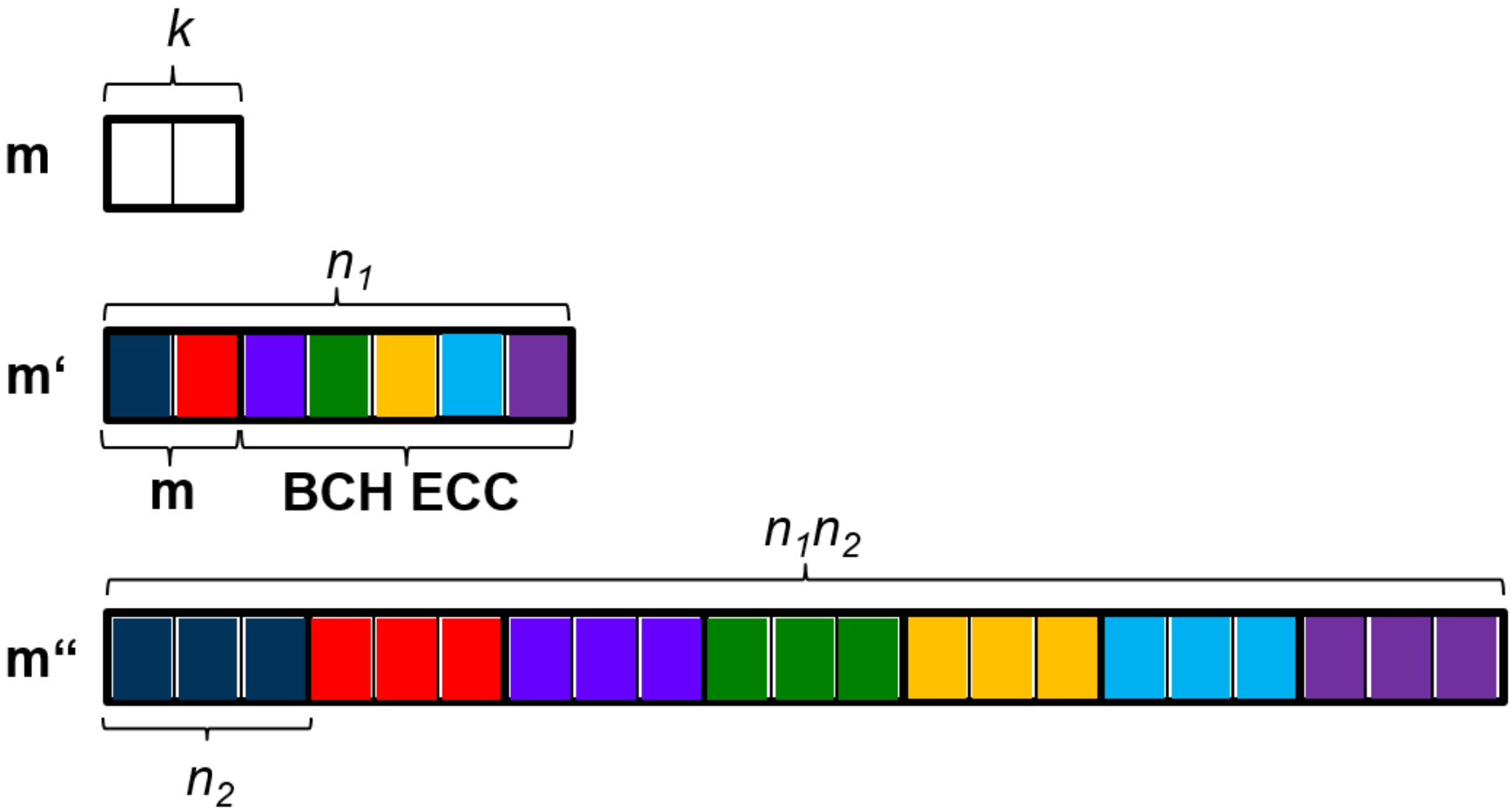
[3] Paiva et al.: *A Timing Attack on the HQC Encryption Scheme*
[4] Wafo-Tapa et al.: *A Practicable Timing Attack Against HQC and its Countermeasure*

Error Correction in HQC

- HQC uses a product code of a $[n_1, k]$ shortened BCH code $\mathcal{C}_1$ with a generator matrix $\boldsymbol{G}_1 \in \mathbb{F}_2^{k \times n_1}$ and a $[n_2, 1]$ repetition code $\mathcal{C}_2$.
- **Encode**: $\mathbb{F}_2^k \to \mathbb{F}_2^{n_1 n_2}$
  - Encode with the BCH code: $\mathbb{F}_2^k \to \mathbb{F}_2^{n_1}$
    - $\boldsymbol{m}' = (m'_0, \ldots, m'_{n_1-1}) = \boldsymbol{m}\boldsymbol{G}_1$
      $\boldsymbol{G}_1$ is a generator matrix of the BCH code $\mathcal{C}_1$.
  - Encode $\boldsymbol{m}'$ with the repetition code: $\mathbb{F}_2^{n_1} \to \mathbb{F}_2^{n_1 n_2}$
    - $\boldsymbol{m}'' = (\underbrace{m'_0, \ldots, m'_0}_{n_2 \text{ times}}, \underbrace{m'_1, \ldots, m'_1}_{n_2 \text{ times}}, m'_2, \ldots, m'_{n_1-1})$
- **Decode**: $\mathbb{F}_2^{n_1 n_2} \to \mathbb{F}_2^k$
  - Decode the repetition code: $\mathbb{F}_2^{n_1 n_2} \to \mathbb{F}_2^{n_1}$
    - Majority decoding with border $\lceil \frac{n_2}{2} \rceil$
  - Decode the BCH code: $\mathbb{F}_2^{n_1} \to \mathbb{F}_2^k$

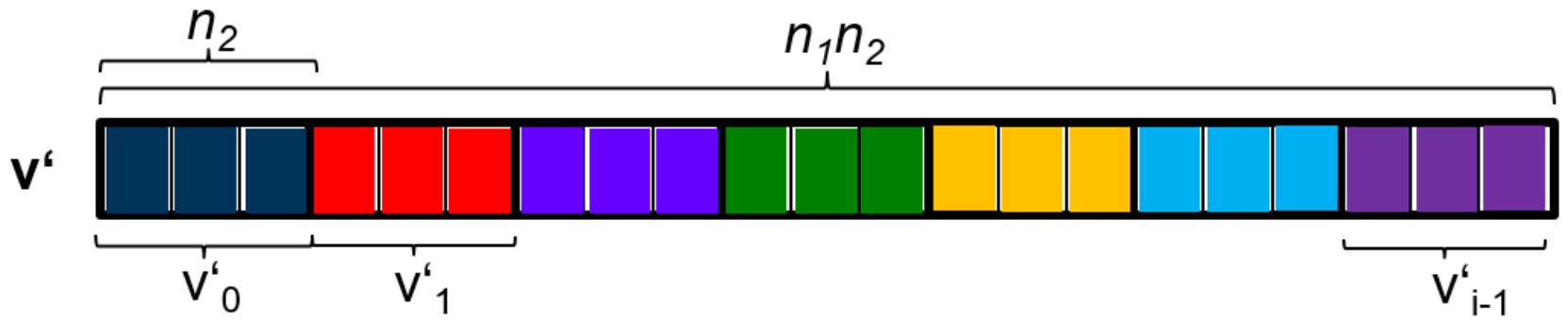
Example ECC ($k = 2$, $n_1 = 7$, $n_2 = 3$)

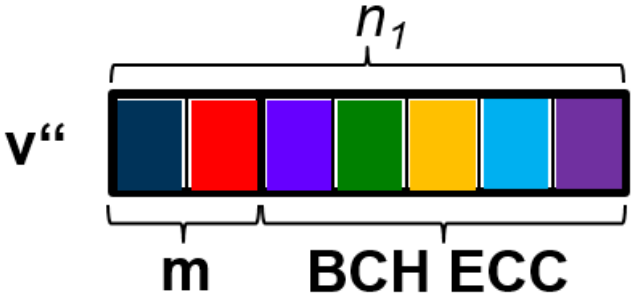
- **Encode**



Example ECC ($k = 2$, $n_1 = 7$, $n_2 = 3$)

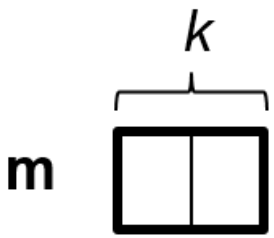
- **Decode**





$$v''_i = 1 \text{ if } \sum_{j=0}^{n_2-1} v'_{ij} \geq \left\lceil \frac{n_2}{2} \right\rceil$$

$$v''_i = 0 \text{ otherwise}$$



HQC - Attack Target

Algorithm 2: Decryption

Input: sk $= (\boldsymbol{x}, \boldsymbol{y})$, ct $= (\boldsymbol{u}, \boldsymbol{v})$
Output: $\boldsymbol{m}$

1 $\boldsymbol{v}' \leftarrow \boldsymbol{v} - \boldsymbol{u}\boldsymbol{y}$
2 $\boldsymbol{m} \leftarrow$ **Decode**$(\boldsymbol{v}')$
3 **return** $\boldsymbol{m}$

- $\boldsymbol{y} \in \mathbb{F}_2^n$ with $\mathrm{HW}(\boldsymbol{y}) = \omega$ and ω is small
- HQC-128: $n = 23869$ and $\omega = 67$
- Set $\boldsymbol{u} = (1, 0, \ldots, 0) \in \mathbb{F}_2^n$

  $\Rightarrow \boldsymbol{v}' = \boldsymbol{v} - \boldsymbol{y}$
- Task: Find $\boldsymbol{v}$ such that power side-channel gives information about $\boldsymbol{y}$
- Solution: Distinguish between correct and faulty BCH input through the power side-channel

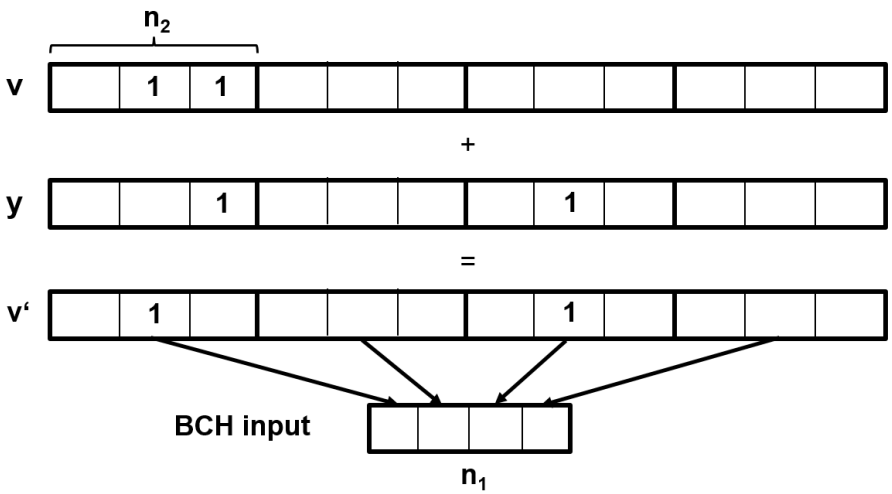
  $\Rightarrow$ Decoding oracle

Attack Approach

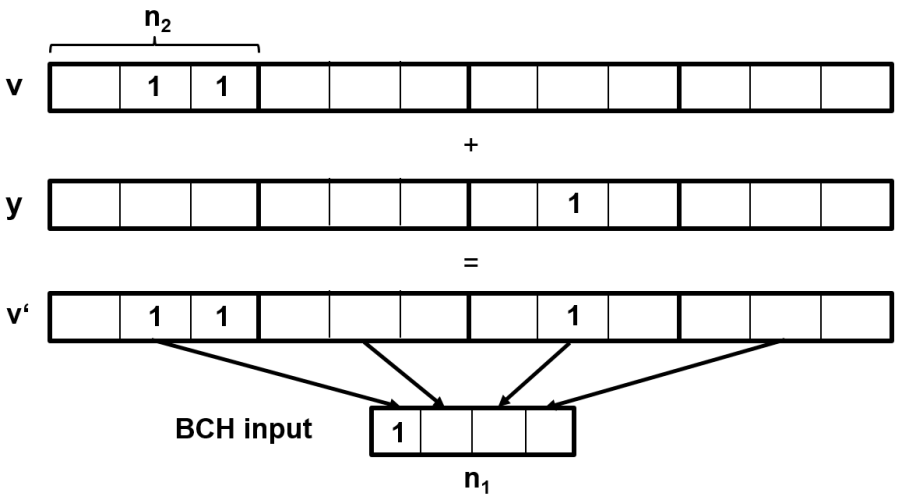
- We want to find the positions of "ones" in $\boldsymbol{y}$ (support of $\boldsymbol{y}$)
- Solution: Craft $\boldsymbol{v}$ in a way that decoding oracle allows to gain information on $\boldsymbol{y}$
- Attack with at two-step approach:
  - Find the approximate support of $\boldsymbol{y}$ (super support)
  - Find the exact support of $\boldsymbol{y}$
- BCH decoding has the following properties:
  - $\mathrm{HW}(\boldsymbol{v}'') = 0$: No Error has to be corrected
  - $\mathrm{HW}(\boldsymbol{v}'') = 1$: Error has to be corrected

Finding the super support of $\boldsymbol{y}$

- The repetition decoder decodes each chunk of n_2 bits of $\boldsymbol{v}'$ separately
  $\Rightarrow$ Attack each chunk (there are n_1 of them) separately
- Set only $\lceil \frac{n_2}{2} \rceil$ entries of $\boldsymbol{v}_i$ to 1 and v_j to zero, with $j \in [0, n_1 - 1] \setminus \{i\}$
- Example for the first chunk $\boldsymbol{y}_0$ ($n_1 = 4$, $n_2 = 3$):



No error



Error

Finding the super support of $\boldsymbol{y}$

Dependency on the $\mathrm{HW}(y_i)$

$\max\{\mathrm{HW}(\boldsymbol{y}_0), \ldots, \mathrm{HW}(\boldsymbol{y}_{n_1-1})\}$	HQC-128	HQC-192	HQC-256
1	5.59%	0.11%	$\approx$ 0%
2	93.20%	77.98%	58.99%
3	99.86%	99.25%	97.99%

Finding the super support of $\boldsymbol{y}$

Dependency on the $\mathrm{HW}(y_i)$

$\max\{\mathrm{HW}(\boldsymbol{y}_0), \ldots, \mathrm{HW}(\boldsymbol{y}_{n_1-1})\}$	HQC-128	HQC-192	HQC-256
1	5.59%	0.11%	$\approx$ 0%
2	93.20%	77.98%	58.99%
3	99.86%	99.25%	97.99%

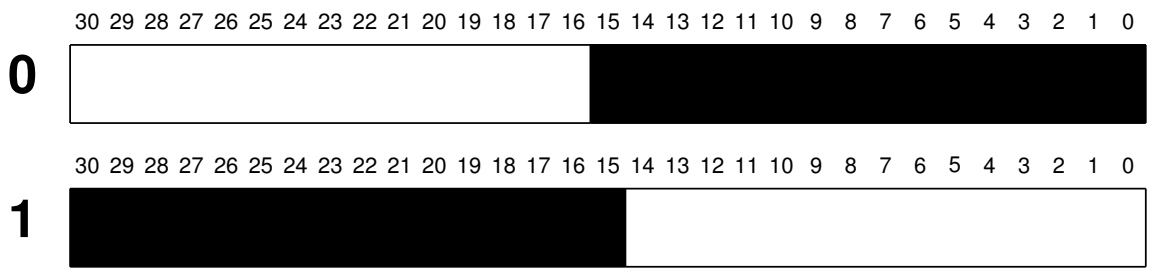


Figure: Pattern for $\mathrm{HW}(\boldsymbol{y}_i) \leq 1$ and $n_2 = 31$ (HQC-128)

Finding the super support of $\boldsymbol{y}$

Dependency on the $\mathrm{HW}(y_i)$

$\max\{\mathrm{HW}(\boldsymbol{y}_0), \ldots, \mathrm{HW}(\boldsymbol{y}_{n_1-1})\}$	HQC-128	HQC-192	HQC-256
1	5.59%	0.11%	$\approx 0\%$
2	93.20%	77.98%	58.99%
3	99.86%	99.25%	97.99%

Figure: Pattern for $\mathrm{HW}(\boldsymbol{y}_i) \leq 2$ and $n_2 = 31$ (HQC-128)

Finding the support of $\boldsymbol{y}$

- With the knowledge of the super support (approximate positions) we can now attack the exact support of $\boldsymbol{y}$
- We can check the individual bits of the corresponding $\boldsymbol{y}_i$ separately

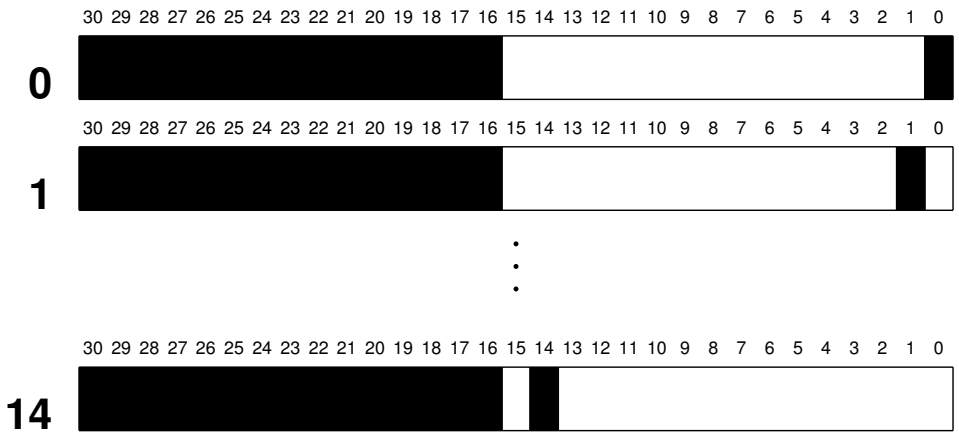


Figure: Patterns to determine $\mathsf{supp}(\boldsymbol{y}_i)$ from $\mathsf{ssupp}(\boldsymbol{y}_i)$ for $n_2 = 31$ and $\mathsf{ssupp}(\boldsymbol{y}_i) = \{0, \ldots, 14\}$.

Decoding Oracle

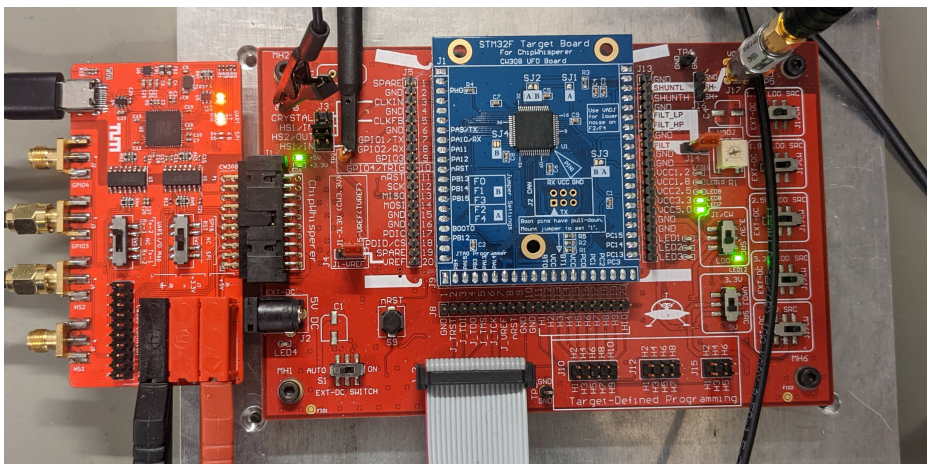
- Distinguish through power side-channel measurements if an error is corrected by the BCH decoder
- Template matching approach shown in CHES 2020 [6]:
  1. Find points of interest (POI) through a t-test
  2. Generate templates for both classes (error or no error) based on POI
  3. Template matching for a given attack trace

$\mathcal{O}_{01}^{Dec}$	$\boldsymbol{u} \in \mathbb{F}_2^n$	$\boldsymbol{v} = (\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{n1-1}) \in \mathbb{F}_2^{n_1 n_2}$ with $\boldsymbol{v}_i \in \mathbb{F}_2^{n_2}$
0 (no error)	0	$(0, \ldots, 0)$
1 (error)	0	$(\mathrm{HW}(\boldsymbol{v}_0) = \lceil \frac{n_2}{2} \rceil, 0, \ldots, 0)$

[6] Ravi et al.: *Generic Side-channel attacks on CCA-secure lattice-based PKE and KEM scheme*

Experimental Setup

- Power measurements with CW308 UFO board
- STM32F415RG microcontroller (ARM Cortex-M4)
- $f_{clk} = 10$ MHz
- $f_{sample} = 156.25$ MHz
- Reference implementation of HQC-128 (Round 2 Submission)



Decoding Oracle

1.) Finding POI through t-test

- Can we distinguish between an error and no error during the BCH decoding?

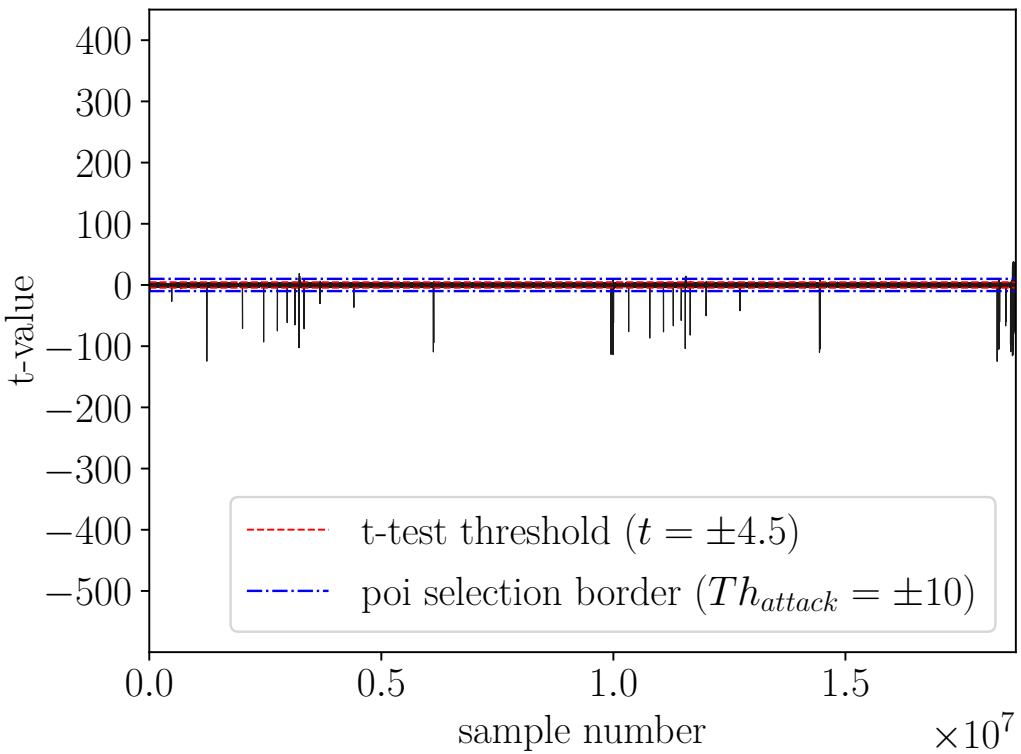
  $\Rightarrow$ T-test
- BCH Decoding as in Reference implementation:
  1. Syndrome computation
  2. Error locator polynomial
  3. Compute the roots of error locator polynomial

Decoding Oracle

1.) Finding POI through t-test

- Can we distinguish between an error and no error during the BCH decoding?

  $\Rightarrow$ T-test
- BCH Decoding as in Reference implementation:
  1. Syndrome computation
  2. Error locator polynomial
  3. Compute the roots of error locator polynomial



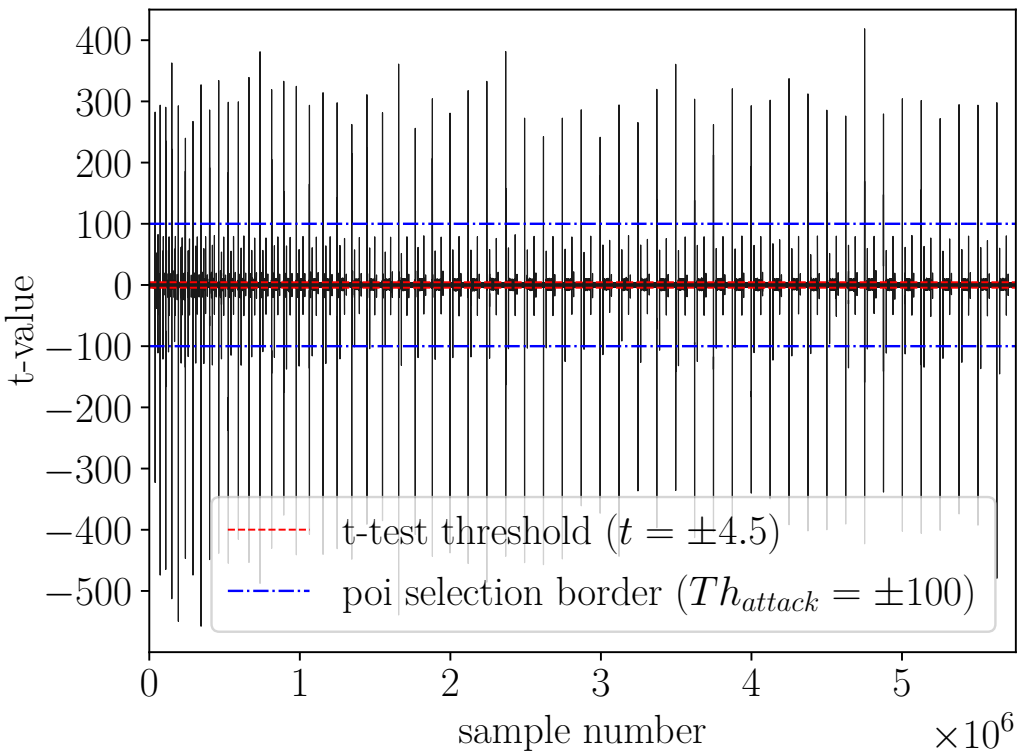
Syndrome computation

Decoding Oracle

1.) Finding POI through t-test

- Can we distinguish between an error and no error during the BCH decoding?

  $\Rightarrow$ T-test
- BCH Decoding as in Reference implementation:
  1. Syndrome computation
  2. Error locator polynomial
  3. Compute the roots of error locator polynomial

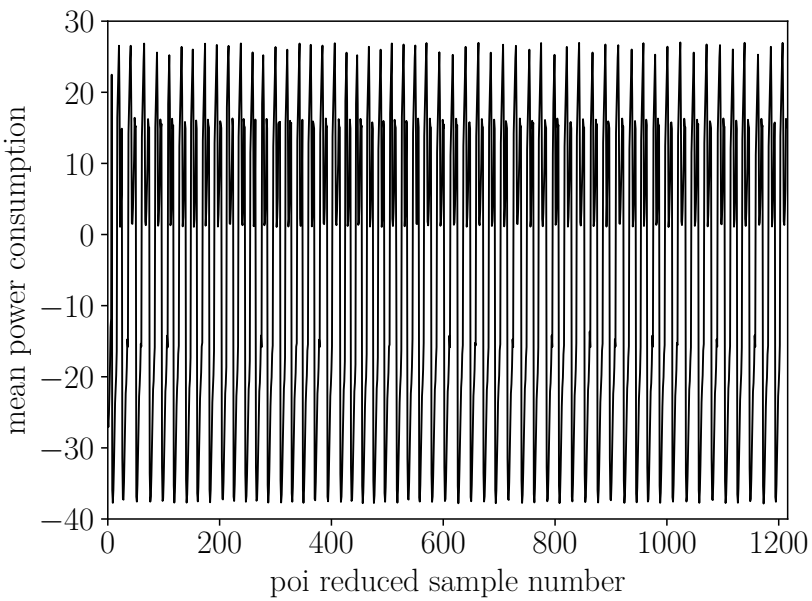


Error locator polynomial

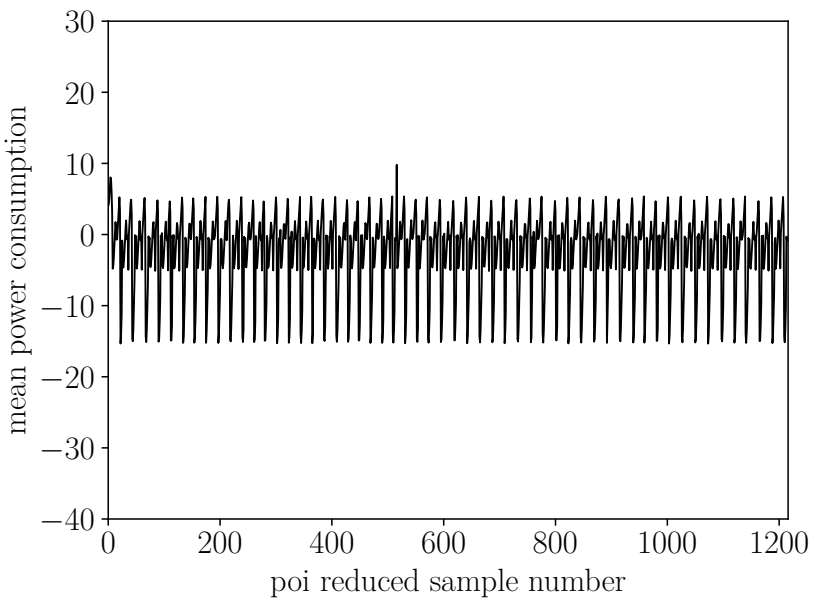
Decoding Oracle

2.) Template generation

- Generate a template for both classes:
  - No Error (t_m^0)
  - Error (t_m^1)
- Mean of all traces for a given class at POI
- Template generation on the target device (no dependency on $\boldsymbol{y}$)



t_m^0



t_m^1

Decoding Oracle

3.) Template matching

- Template matching using "sum of squared differences"
- Attack trace t_{attack} is classified to class with the lowest *SSD* result:
  - $SSD(t_{attack}, \boldsymbol{t_m^0}) < SSD(t_{attack}, \boldsymbol{t_m^1}) \rightarrow$ No Error
  - $SSD(t_{attack}, \boldsymbol{t_m^0}) > SSD(t_{attack}, \boldsymbol{t_m^1}) \rightarrow$ Error

Attack results

- Decoding oracle successfully classified on $20k$ traces
- Successful attack on the HQC-128 reference implementation
- Required template traces per class:
  - Results shown with 500 traces
  - A minimum of two traces
- Required attack traces:

	$\max\{\mathrm{HW}(\boldsymbol{y}_i)\} = 1$	$\max\{\mathrm{HW}(\boldsymbol{y}_i)\} = 2$
ssupp($\boldsymbol{y}$)	1532	4596
supp($\boldsymbol{y}$)	1005	3976

Conclusion

- Power side-channel attack on the HQC KEM
- 93.2% of possible keys of HQC-128 can be attacked
  - Can be increased with additional attack traces
- Proposed solutions for "special keys":
  - Linear algebra solution for $\mathrm{HW}(\boldsymbol{y}[n_1 n_2, n - n_1 n_2]) > 0$
  - Information set decoding
- Practical attack results on ARM Cortex-M4 with less than 10000 traces
- **Future work**:
  - Countermeasures
  - Transfer to other HQC instances (Reed-Muller + Reed-Solomon code)

Thank You!

Thomas Schamberger

t.schamberger@tum.de
https://www.sec.ei.tum.de/

Deutsche Forschungsgemeinschaft
German Research Foundation

This work was supported by the German Research Foundation (DFG) under grant number SE2989/1-1 and by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 801434).