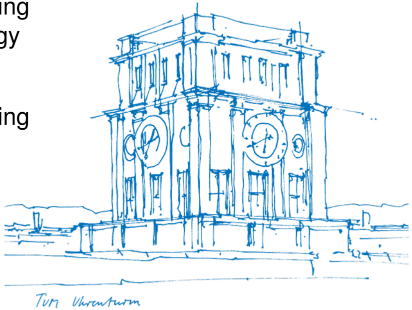
A Power Side-Channel Attack on the Reed-Muller Reed-Solomon Version of the HQC Cryptosystem

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

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

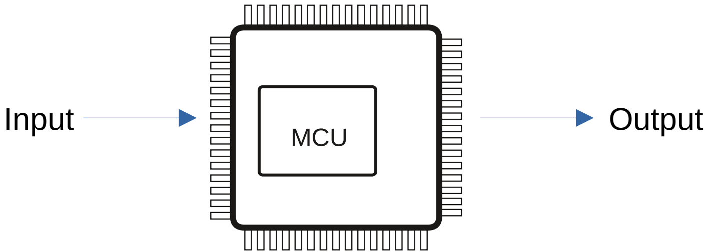
[2]Technical University of Munich
Faculty of Electrical and Computer Engineering
Institute for Communications Engineering

28.09.2022

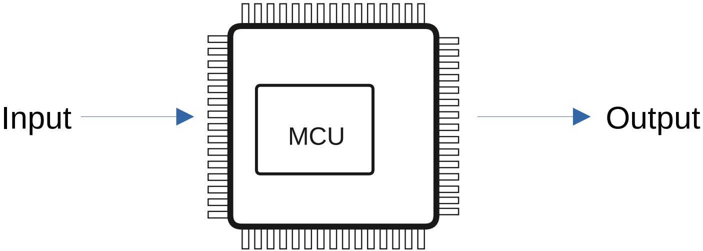


PQCrypto 2022

Side-Channel Attacks

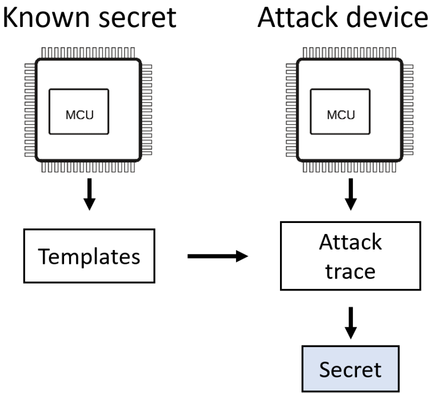


Side-Channel Attacks



Profiled Side-Channel Attacks

- Build "Templates" of the real power consumption
- Done for all different intermediate values or classes



Hamming Quasi Cyclic (HQC)

- Code-based key encapsulation mechanism (KEM)
- Fourth round *alternative* candidate

Algorithm 1:
Encrypt

Input: $\boldsymbol{m}, \mathrm{pk} = (\boldsymbol{h}, \boldsymbol{s}), \theta$
Output: c

$\boldsymbol{e}' \xleftarrow{\$(w_{\mathrm{e}}, \theta)} \mathcal{R}$
$(\boldsymbol{r}_1, \boldsymbol{r}_2) \xleftarrow{\$(w_{\mathrm{r}}, \theta)} \mathcal{R}^2$
$\boldsymbol{u} \leftarrow \boldsymbol{r}_1 + \boldsymbol{h}\boldsymbol{r}_2$
$\boldsymbol{v} \leftarrow \textbf{Encode}(\boldsymbol{m}) + \boldsymbol{s}\boldsymbol{r}_2 + \boldsymbol{e}'$
return $c = (\boldsymbol{u}, \boldsymbol{v})$

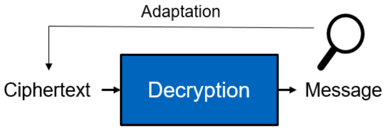
Algorithm 2:
Decrypt

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

$\boldsymbol{v}' \leftarrow \boldsymbol{v} - \boldsymbol{u}\boldsymbol{y}$
$\boldsymbol{m} \leftarrow \textbf{Decode}(\boldsymbol{v}')$
return $\boldsymbol{m}$

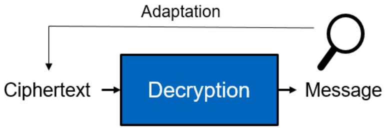
HQC - KEM

- The PKE version is vulnerable against chosen-ciphertext attacks



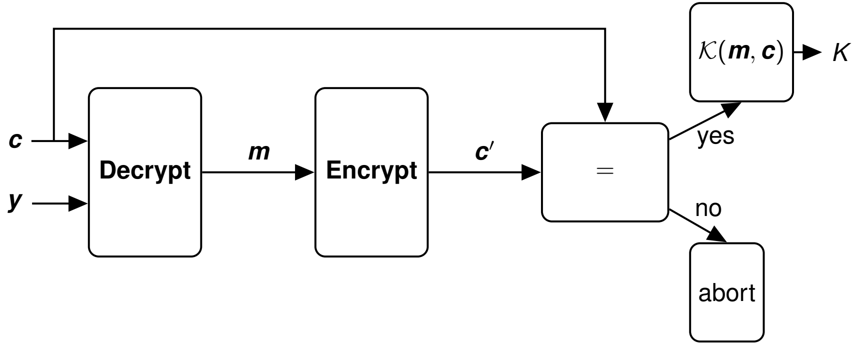
HQC - KEM

- The PKE version is vulnerable against chosen-ciphertext attacks



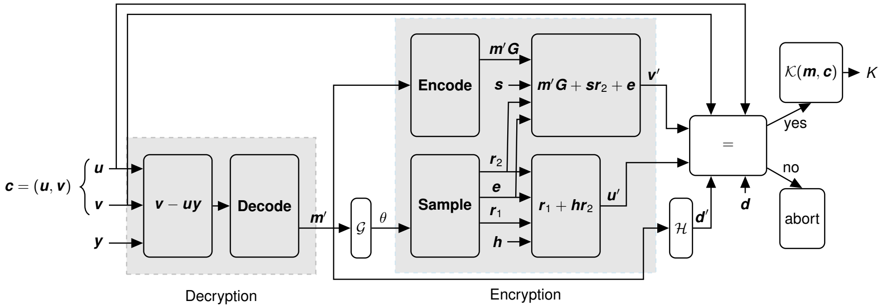
- HQC uses a variant of the Fujisaki-Okamoto transformation to achieve a CCA2-secure KEM

HQC - KEM

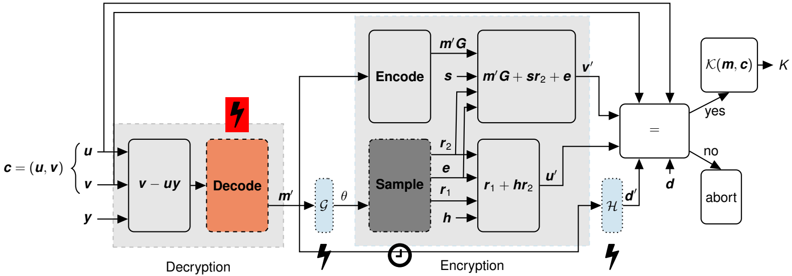


$\boldsymbol{c}$
$\boldsymbol{y}$
Decrypt
$\boldsymbol{m}$
Encrypt
$\boldsymbol{c}'$
$=$
yes
no
$\mathcal{K}(\boldsymbol{m}, \boldsymbol{c})$
K
abort

HQC - KEM


$c = (u, v)$
u
v
y
$v - uy$
Decode
m'
$\mathcal{G}$
θ
Encode
$m'G$
s
$m'G + sr_2 + e$
v'
Sample
r_2
e
r_1
h
$r_1 + hr_2$
u'
$\mathcal{H}$
d'
d
=
$\mathcal{K}(m, c)$
K
yes
no
abort
Decryption
Encryption

HQC - Side-channel attacks



This work.

[1] Uneo et al.: *Curse of re-encryption: A generic Power/EM analysis on post-quantum KEMs*, CHES 2022

[2] Guo et al.: *Don't reject this: Key-recovery timing attacks due to rejection-sampling in HQC and BIKE*, CHES 2022

Attack components

- Chosen-ciphertext attacks need two attack components:

Attack strategy

- How to find inputs with an observable oracle result based on secret

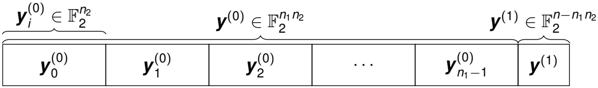
Side-channel oracle

- Classification result provides needed information for attack strategy $\rightarrow$ Oracle

$\Rightarrow$ Both steps components influence each other

Recap HQC - Parameters

	shortened RS code $[n_1, k, d_{RS}]$	duplicated RM code $[n_2, k_{RM}, d_{RM}, s]$	n	w
HQC-128	[46, 16, 31]	[384, 8, 192, 3]	17,669	66
HQC-192	[56, 24, 33]	[640, 8, 320, 5]	35,851	100
HQC-256	[90, 32, 49]	[640, 8, 320, 5]	57,637	131



Algorithm 2: Decrypt

Input: $\mathrm{sk} = (\mathbf{x}, \mathbf{y})$
$c = (\mathbf{u}, \mathbf{v})$

Output: $\mathbf{m}$

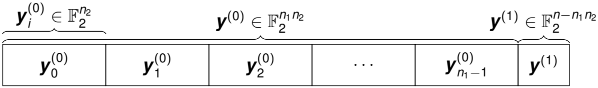
$\mathbf{v}' \leftarrow \mathbf{v} - \mathbf{u}\mathbf{y}$

$\mathbf{m} \leftarrow \mathbf{Decode}(\mathbf{v}')$

return $\mathbf{m}$

Recap HQC - Parameters

	shortened RS code $[n_1, k, d_{RS}]$	duplicated RM code $[n_2, k_{RM}, d_{RM}, s]$	n	w
HQC-128	[46, 16, 31]	[384, 8, 192, 3]	17,669	66
HQC-192	[56, 24, 33]	[640, 8, 320, 5]	35,851	100
HQC-256	[90, 32, 49]	[640, 8, 320, 5]	57,637	131



- Note: Previous versions used **repetition code** and **shortened BCH code**

Algorithm 2: Decrypt

Input: $\mathrm{sk} = (\boldsymbol{x}, \boldsymbol{y})$

$c = (\boldsymbol{u}, \boldsymbol{v})$

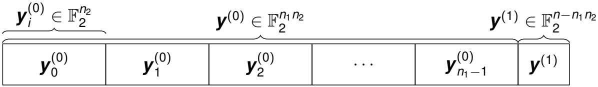
Output: $\boldsymbol{m}$

$\boldsymbol{v}' \leftarrow \boldsymbol{v} - \boldsymbol{u}\boldsymbol{y}$

$\boldsymbol{m} \leftarrow \mathbf{Decode}(\boldsymbol{v}')$

return $\boldsymbol{m}$

General Attack Idea



Algorithm 2: Decrypt

Input: $\mathrm{sk} = (\boldsymbol{x}, \boldsymbol{y})$

$c = (\boldsymbol{u}, \boldsymbol{v})$

Output: $\boldsymbol{m}$

$\boldsymbol{v}' \leftarrow \boldsymbol{v} - \boldsymbol{u}\boldsymbol{y}$

$\boldsymbol{m} \leftarrow$ **Decode**$(\boldsymbol{v}')$

return $\boldsymbol{m}$

General Attack Idea

$\boldsymbol{y}_i^{(0)} \in \mathbb{F}_2^{n_2}$ $\quad$ $\boldsymbol{y}^{(0)} \in \mathbb{F}_2^{n_1 n_2}$ $\quad$ $\boldsymbol{y}^{(1)} \in \mathbb{F}_2^{n-n_1 n_2}$

$\boldsymbol{y}_0^{(0)}$	$\boldsymbol{y}_1^{(0)}$	$\boldsymbol{y}_2^{(0)}$	$\cdots$	$\boldsymbol{y}_{n_1-1}^{(0)}$	$\boldsymbol{y}^{(1)}$

- Goal: Find support of $\boldsymbol{y}$
- Attack each $\boldsymbol{y}_i^{(0)}$ separately
- Decoding result gives information $\rightarrow$ **SCA oracle**

Algorithm 2: Decrypt

Input: $\mathrm{sk} = (\boldsymbol{x}, \boldsymbol{y})$
$c = (\boldsymbol{u}, \boldsymbol{v})$

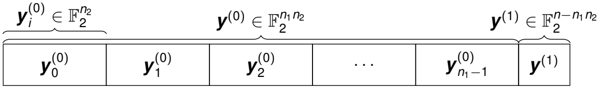
Output: $\boldsymbol{m}$

$\boldsymbol{v}' \leftarrow \boldsymbol{v} - \boldsymbol{u}\boldsymbol{y}$

$\boldsymbol{m} \leftarrow$ **Decode**$(\boldsymbol{v}')$

return $\boldsymbol{m}$

General Attack Idea



- Goal: Find support of $\boldsymbol{y}$
- Attack each $\boldsymbol{y}_i^{(0)}$ separately
- Decoding result gives information $\rightarrow$ **SCA oracle**
- Steps:
  1. Find input $c = (\boldsymbol{u} = (1, 0 \ldots, 0) \in \mathbb{F}_2^n, \boldsymbol{v})$ at decoder boundary
  2. Test each individual bit of $\boldsymbol{y}_i^{(0)}$

Algorithm 2: Decrypt

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

Output: $\boldsymbol{m}$

$\boldsymbol{v}' \leftarrow \boldsymbol{v} - \boldsymbol{u}\boldsymbol{y}$

$\boldsymbol{m} \leftarrow$ **Decode**$(\boldsymbol{v}')$

return $\boldsymbol{m}$

Power SCA from Ueno et al. [1] not working

- Transfer of plaintext-checking attack from second-round version [3]
- Problem: Only valid for bounded distance decoder
- **Reed-Muller codes** of third round version are decoded using an ML decoder
  - Decoding result depends on *number of errors* and *support*

[1] Uneo et al.: *Curse of re-encryption: A generic Power/EM analysis on post-quantum KEMs*, CHES 2022
[3] Bâetu et al.: *Misuse attacks on post-quantum cryptosystem*, 2019

Power SCA from Ueno et al. [1] not working

- Transfer of plaintext-checking attack from second-round version [3]
- Problem: Only valid for bounded distance decoder
- **Reed-Muller codes** of third round version are decoded using an ML decoder
  - Decoding result depends on *number of errors* and *support*

Power SCA from Ueno et al. [1] not working

- Transfer of plaintext-checking attack from second-round version [3]
- Problem: Only valid for bounded distance decoder
- **Reed-Muller codes** of third round version are decoded using an ML decoder
  - Decoding result depends on *number of errors* and *support*
- Strategy does not work anymore:
  - Counterexample
  - Simulations using ideal decoder results

[1] Uneo et al.: *Curse of re-encryption: A generic Power/EM analysis on post-quantum KEMs*, CHES 2022
[3] Bâetu et al.: *Misuse attacks on post-quantum cryptosystem*, 2019

Valid Attack Strategy for ML Decoder

- Based on a Close-to-0-Oracle:

$$\mathfrak{D}_0^{\boldsymbol{e}}(\boldsymbol{r}) = \begin{cases} \text{True}, & \text{if } \mathfrak{D}_{\mathcal{RM}}(\boldsymbol{r}+\boldsymbol{e}) = \boldsymbol{0}, \\ \text{False}, & \text{else} \end{cases}$$

Valid Attack Strategy for ML Decoder

- Based on a Close-to-0-Oracle:

$$\mathfrak{D}_0^{\boldsymbol{e}}(\boldsymbol{r}) = \begin{cases} \text{True}, & \text{if } \mathfrak{D}_{\mathcal{RM}}(\boldsymbol{r}+\boldsymbol{e}) = \boldsymbol{0}, \\ \text{False}, & \text{else} \end{cases}$$

- Choose input as RM codewords in a way that:
  - Oracle result is always valid depending on $\boldsymbol{e}$
  - No ties possible
- This allows to follow the general attack strategy for ML decoder

Valid Attack Strategy for ML Decoder

- Based on a Close-to-0-Oracle:

$$\mathfrak{D}_0^{\boldsymbol{e}}(\boldsymbol{r}) = \begin{cases} \text{True}, & \text{if } \mathfrak{D}_{\mathcal{RM}}(\boldsymbol{r}+\boldsymbol{e}) = \boldsymbol{0}, \\ \text{False}, & \text{else} \end{cases}$$

- Choose input as RM codewords in a way that:
  - Oracle result is always valid depending on $\boldsymbol{e}$
  - No ties possible
- This allows to follow the general attack strategy for ML decoder
- Proof in paper for $\mathrm{HW}(\boldsymbol{y}_i^{(0)}) < \frac{d_{\mathcal{RM}}}{4}$
- Valid with very high probability for HQC parameter sets

Valid Attack Strategy for ML Decoder

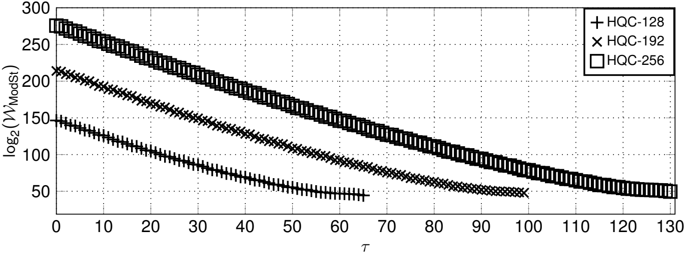
- Based on a Close-to-0-Oracle:

$$\mathfrak{D}_0^{\boldsymbol{e}}(\boldsymbol{r}) = \begin{cases} \text{True}, & \text{if } \mathfrak{D}_{\mathcal{RM}}(\boldsymbol{r}+\boldsymbol{e}) = \mathbf{0}, \\ \text{False}, & \text{else} \end{cases}$$

- Choose input as RM codewords in a way that:
  - Oracle result is always valid depending on $\boldsymbol{e}$
  - No ties possible
- This allows to follow the general attack strategy for ML decoder
- Proof in paper for $\mathrm{HW}(\boldsymbol{y}_i^{(0)}) < \frac{d_{\mathcal{RM}}}{4}$
- Valid with very high probability for HQC parameter sets
- Verified with perfect oracle calls from decoder of the reference implementation

Partial Information with Information-Set Decoding

- Partial information of $\boldsymbol{y}$ due to:
  - Amount of oracle calls is limited
  - Side-channel does not allow perfect oracle answers
- Modified variant of Stern's algorithm [4]



[4] Stern: *A method for finding codewords of small weight, Coding Theory and Applications, 1989*

Comparison with Related Work

- Our strategy requires a *maximum* amount of oracle calls:

$$n_{calls} = n_1 \cdot 4 \cdot \left(\frac{2 \cdot n_2}{4} + \frac{n_2}{4}\right)$$

Comparison with Related Work

- Our strategy requires a *maximum* amount of oracle calls:

$$n_{calls} = n_1 \cdot 4 \cdot (\frac{2 \cdot n_2}{4} + \frac{n_2}{4})$$

- Guo et al. [2] show non-deterministic timing attack:
  - Randomly increase hamming weight of RM input until decoder boundary
  - Has to be repeated several times until each position in $\boldsymbol{y}_i^{(0)}$ is evaluated
  - Uncertainty of timing oracle requires majority threshold

[2] Guo et al.: *Don't reject this: Key-recovery timing attacks due to rejection-sampling in HQC and BIKE*, CHES 2022

Comparison with Related Work

- Our strategy requires a *maximum* amount of oracle calls:

$$n_{calls} = n_1 \cdot 4 \cdot (\frac{2 \cdot n_2}{4} + \frac{n_2}{4})$$

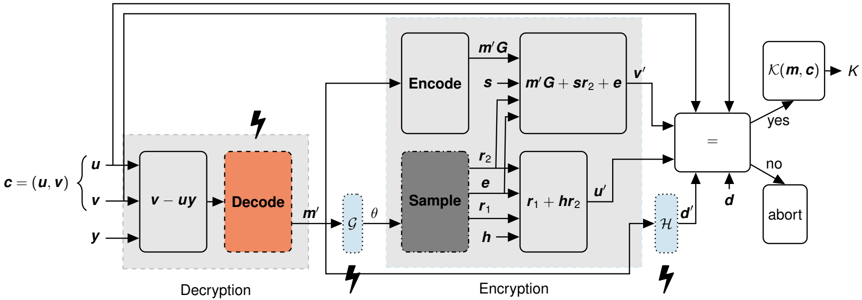
- Guo et al. [2] show non-deterministic timing attack:
  - Randomly increase hamming weight of RM input until decoder boundary
  - Has to be repeated several times until each position in $\boldsymbol{y}_i^{(0)}$ is evaluated
  - Uncertainty of timing oracle requires majority threshold

	This work	Timing Attack [2]	Strategy of [2] using $\mathfrak{D}_0^{\boldsymbol{e}}$
HQC-128	1152*46	18829*46	13174*46
HQC-192	1920*56	-	23170*56
HQC-256	1920*90	-	23170*90

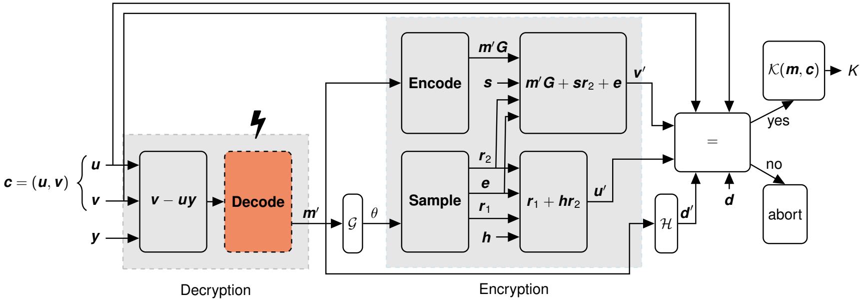
[2] Guo et al.: *Don't reject this: Key-recovery timing attacks due to rejection-sampling in HQC and BIKE*, CHES 2022

SCA Targets Overview

- Side-channel targets to build $\mathfrak{D}_0^{\boldsymbol{e}}(\boldsymbol{r})$:



Power Side-Channel of the RS Decoder



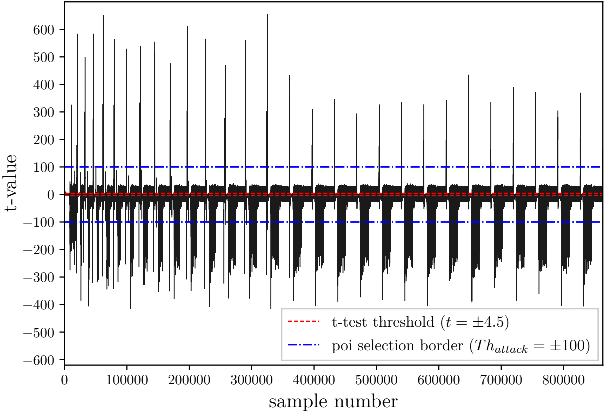
Power Side-Channel of the RS Decoder

- Adaptation of the attack on the BCH decoder from [5]
- Directly applicable as BCH codes are subcodes from RS codes
- RS decoder has to correct an error if $\boldsymbol{y}_i^{(0)}$ is not the all-zero codeword
- Attack results:
  - Latest HQC-128 reference implementation
  - STM32F415 Cortex-M4 microcontroller
  - 1000 template traces
  - 100,000 correctly classified attack traces

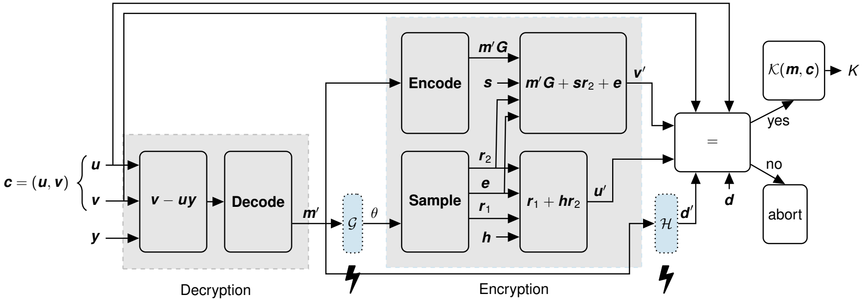
[5] Schamberger et al.: *A power side-channel attack on the cca2-secure HQC KEM*, CARDIS 2020

Power Side-Channel of the RS Decoder

- Attack target: Error-locator polynomial computation



Power Side-Channel: Hash Functions $\mathcal{G}, \mathcal{H}$



Power Side-Channel: Hash Functions $\mathcal{G}, \mathcal{H}$

- Plaintext checking oracle through SHAKE256-512 [1]
- Decoding result $\boldsymbol{m}'$ directly influences computation
- Oracle evaluated by the authors:
  - Same hardware attack target
  - SHAKE software implementation of `pqm4`
  - Machine learning classifier (CNN)
  - Accuracy of 0.998 for 10,000 attack traces
- Needs adaptation such that $(\boldsymbol{d}_{\mathcal{RS}} - 1)/2$ blocks of $\boldsymbol{y}^{(0)}$ contain an error

[1] Uneo et al.: *Curse of re-encryption: A generic Power/EM analysis on post-quantum KEMs*, CHES 2021

Conclusion

- Updated version of HQC requires new attack strategies
- Used Reed-Muller codes are decoded through ML decoder

  $\rightarrow$ breaks attack assumption from Uneo et al. [1]
- New proven attack strategy through close-to-0-Oracle
- Possible side-channel targets to build oracle:
  - Decoder during decryption
  - SHAKE256 to generate randomness for sampler
  - Timing side-channel of rejection sampling **not usable**
- Information set decoding results for partial retrieved keys
- Practical power side-channel results of the implement Reed-Solomon decoder of the HQC-128 reference implementation

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 (Deutsche Forschungsgemeinschaft, DFG) under Grant No. SE2989/1-1 as well as WA3907/4-1 and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 801434).