Don't Reject This: Key-Recovery Timing Attacks Due to Rejection-Sampling in HQC and BIKE

Qian Guo[3], Clemens Hlauschek[1,5], Thomas Johansson[3], Norman Lahr[2], Alexander Nilsson[3,4], and **Robin Leander Schröder**[1]

September 19, 2022

[1]Technische Universität Wien, Austria
[2]Fraunhofer SIT, Darmstadt, Germany
[3]Lund University, Lund, Sweden
[4]Advenica AB, Malmö, Sweden
[5]RISE GmbH, Wien, Austria

Elevator Pitch

Rejection sampling with a seed derived from the message leaks the secret key.

Fundamentals

Attack

Countermeasures

Hamming-Quasi-Cyclic (HQC)

HQC

Code-based round 3 contender

Based on hard problems related to quasi-cyclic codes.

KEM

Key Encapsulation Mechanism (KEM) $\mathcal{E}_{kem}$ with security parameter λ

Tuple of algorithms: $(\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KeyGen}(1^\lambda)$

$(k_0, c) \leftarrow\$ \mathsf{Encaps}(\mathsf{pk}, 1^\lambda)$

$k_1 \leftarrow \mathsf{Decaps}(\mathsf{sk}, c)$

Correctness: $k_0 = k_1$ with overwhelming probability

Game Based Security:

IND-CPA: given encaps oracle, can't distinguish real key from random key

IND-CCA: given additional decaps oracle

Hamming Quasi-Cyclic (HQC) PKE: Key Generation [Agu+16; Car+20]

$\mathcal{R} := \mathbb{F}_2[X]/\langle X^n - 1\rangle$

KeyGen(param)

$\mathbf{h}, \mathbf{x}, \mathbf{y} \leftarrow\!\!\$\ \mathcal{R}$ with $\omega(\mathbf{x}) = \omega(\mathbf{y}) = \omega$

$\mathtt{sk} = (\mathbf{x}, \mathbf{y})$

$\mathtt{pk} = (\mathbf{h}, \mathbf{s} = \mathbf{x} + \mathbf{h} \cdot \mathbf{y})$

return $(\mathtt{pk}, \mathtt{sk})$

HQC PKE: Encryption and Decryption

Encrypt($\texttt{pk}$, $\mathbf{m}$)

$\mathbf{e}, \mathbf{r_1}, \mathbf{r_2} \xleftarrow{\$} \mathcal{R}$ with $\omega(\mathbf{e}) = \omega_e$ and $\omega(\mathbf{r_1}) = \omega(\mathbf{r_2}) = \omega_r$

$\mathbf{u} = \mathbf{r_1} + \mathbf{h} \cdot \mathbf{r_2}$

$\mathbf{v} = \mathbf{mG} + \mathbf{s} \cdot \mathbf{r_2} + \mathbf{e}$

return $c = (\mathbf{u}, \mathbf{v})$

Decrypt($\texttt{sk} = (\mathbf{x}, \mathbf{y})$, $c = (\mathbf{u}, \mathbf{v})$)

return $\mathcal{C}$.Decode($\mathbf{v} - \mathbf{u} \cdot \mathbf{y}$)

Decryption Failures

Decoding is successful when $\mathbf{v} - \mathbf{u} \cdot \mathbf{y}$ has $\leq \delta$ errors:

$$
\begin{aligned}
&\mathbf{v} - \mathbf{u} \cdot \mathbf{y} \\
=&\mathbf{mG} + \mathbf{s} \cdot \mathbf{r_2} + \mathbf{e} - (\mathbf{r_1} + \mathbf{h} \cdot \mathbf{r_2}) \cdot \mathbf{y} \\
=&\mathbf{mG} + (\mathbf{x} + \mathbf{h} \cdot \mathbf{y}) \cdot \mathbf{r_2} + \mathbf{e} - (\mathbf{r_1} + \mathbf{h} \cdot \mathbf{r_2}) \cdot \mathbf{y} \\
=&\mathbf{mG} + \underbrace{\mathbf{x} \cdot \mathbf{r_2} + \mathbf{e} - \mathbf{r_1} \cdot \mathbf{y}}_{\text{sparse}}
\end{aligned}
$$

HQC KEM: Key Generation and Encapsulation

KeyGen(param)

return $\mathit{PKE.KeyGen}(\mathit{param})$

Encaps(pk)

$\mathbf{m} \xleftarrow{\$} \mathbb{F}_2^k$

$\theta = \mathcal{G}(\mathbf{m})$

$c = \text{PKE.Encrypt}(\text{pk}, \mathbf{m}; \theta)$

$K = \mathcal{K}(\mathbf{m}, c)$

$d = \mathcal{H}(\mathbf{m})$

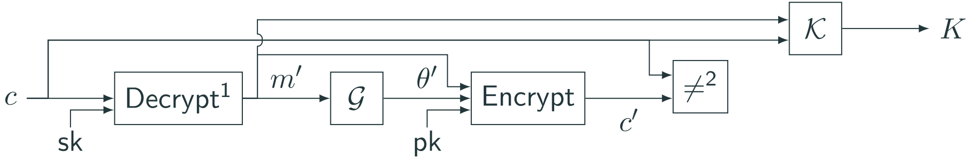
return $(K, (c, d))$

HQC KEM: Decaps

d

[1]WT+19; PT19.
[2]GJN20.

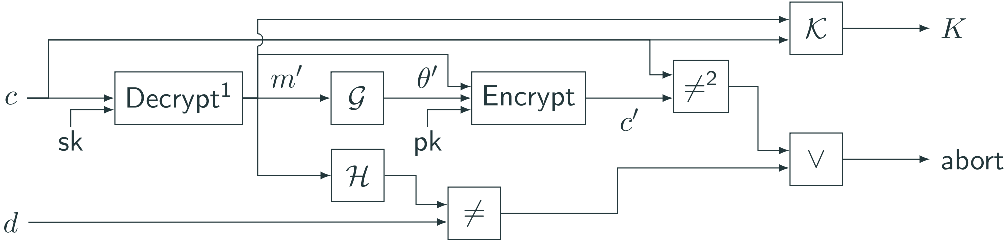
HQC KEM: Decaps



d

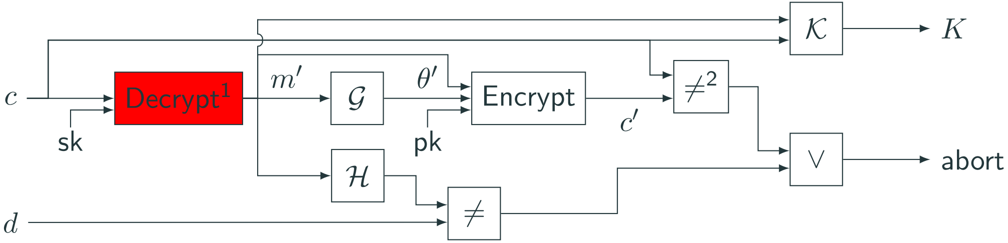
[1]WT+19; PT19.
[2]GJN20.

HQC KEM: Decaps



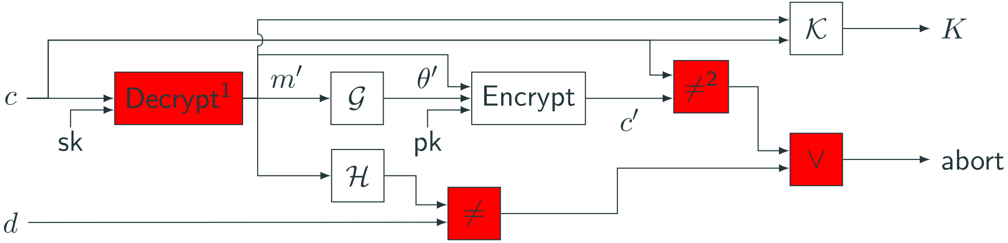
[1] WT+19; PT19.
[2] GJN20.

HQC KEM: Decaps



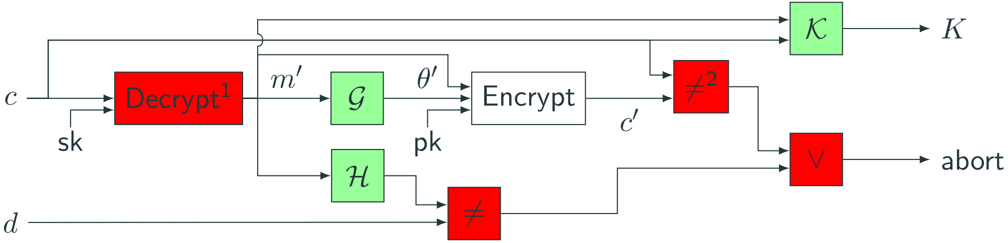
[1] WT+19; PT19.
[2] GJN20.

HQC KEM: Decaps



[1] WT+19; PT19.
[2] GJN20.

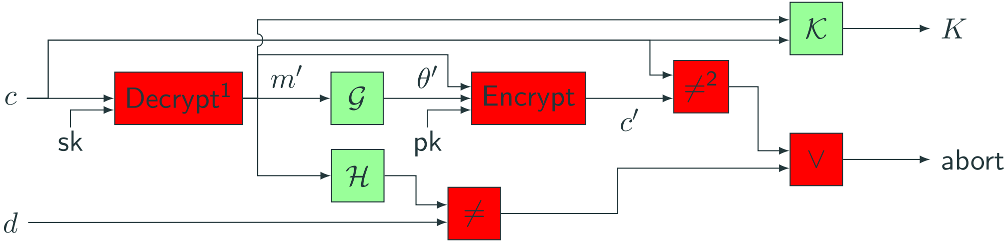
HQC KEM: Decaps



[1] WT+19; PT19.

[2] GJN20.

HQC KEM: Decaps



[1] WT+19; PT19.

[2] GJN20.

Discovery of a Timing-Variation

Discovery

Algorithm 6: Collecting timing measurements

$(\texttt{pk}, \texttt{sk}) \leftarrow \textsf{KeyGen}(1^n)$

for $\texttt{i} \in \{1, \ldots, \texttt{num\_ciphertexts}\}$ **do**

  $(\texttt{c}, \texttt{k}) \leftarrow \textsf{Encaps}(\texttt{pk})$

  for $\texttt{j} \in \{1, \ldots, \texttt{num\_measurements}\}$ **do**

    $\texttt{measure\_cycles}(\textsf{Decaps}(\texttt{sk}, \texttt{c}))$

  end

end

Detecting timing differences

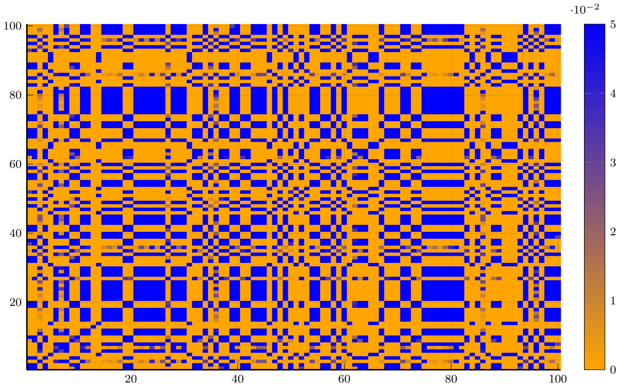
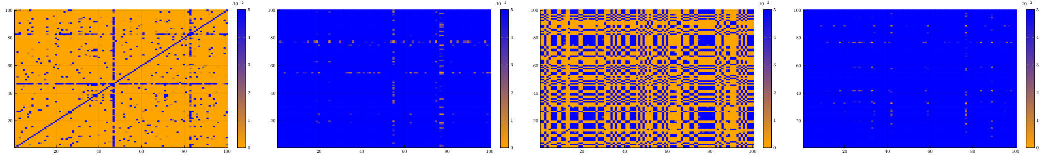


Figure 1: P-values of Welch's t-test:

Statistically significant difference & No statistically significant difference.

Detected differences: 8260 cycles ($\approx 4.13\mu s$ @ 2 GHz).

Recursing into Decaps



(a) Key loading
205 cycles

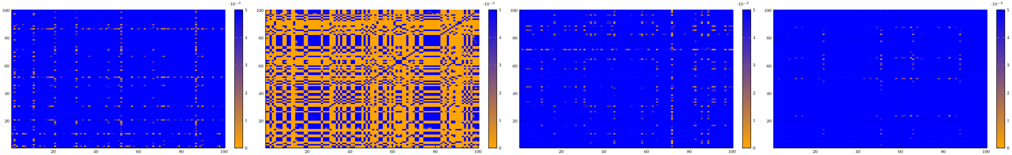
(b) Decryption
462 cycles

(c) Re-encryption
7736 cycles

(d) Shared secret
92 cycles

Figure 2: P-values of Welch's t-test

Recursing into Re-encryption



(a) Init
125 cycles

(b) Sample vector
7520 cycles

(c) Encode
59 cycles

(d) Add error
50 cycles

Figure 3: P-values of Welch's t-test

Vector sampling

How to sample $\mathbf{e}, \mathbf{r_1}, \mathbf{r_2} \leftarrow\!\$\ \mathcal{R}$ in Encrypt:

Rejection sampling of a vector of length n with Hamming weight $= w$.

Algorithm 7: vect_set_random_fixed_weight

Input: weight w, length $n \leq 2^{24}$

Result: vector v of length n with weight $\|v\| = w$

$v = \mathbf{0}^n$

$\omega = 0$

repeat

    repeat

        $i \xleftarrow{\$} [0, 2^{24})$

    until $i < \left\lfloor \frac{2^{24}}{n} \right\rfloor n$

    $i = i \mod n$

    if $v_i \neq 1$ **then**

        $v_i = 1$

        $\omega = \omega + 1$

    end

until $\omega = w$

return v

```
seedexpander(ctx, rand_bytes, random_bytes_size);
for (uint32_t i = 0 ; i < weight ; ++i) {
  do {
    if (j == random_bytes_size) {
      seedexpander(ctx, rand_bytes, random_bytes_size);
      j = 0;
    }
    random_data  = ((uint32_t) rand_bytes[j++]) << 16;
    random_data |= ((uint32_t) rand_bytes[j++]) << 8;
    random_data |= rand_bytes[j++];
  } while (random_data >= UTILS_REJECTION_THRESHOLD);
  random_data = random_data % PARAM_N;
  // [...]
```

Only performed when randomess is exhausted

Data Flow and Relevance

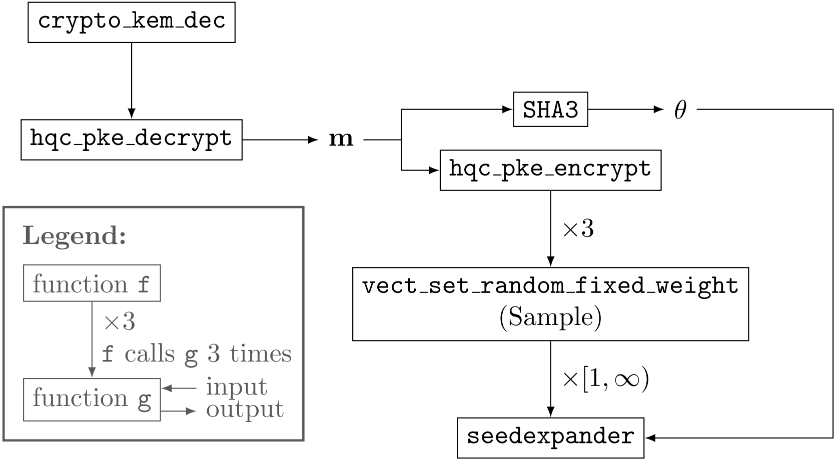
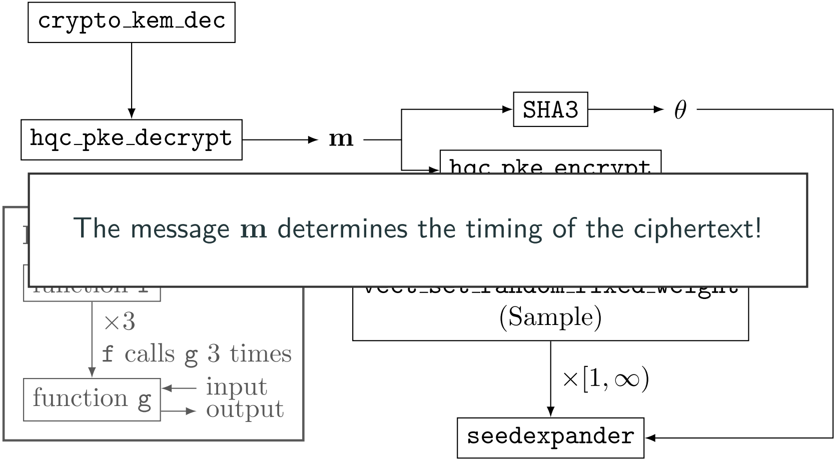


Figure 4: Data flow in HQC.

Data Flow and Relevance



The message $\mathbf{m}$ determines the timing of the ciphertext!

Figure 4: Data flow in HQC.

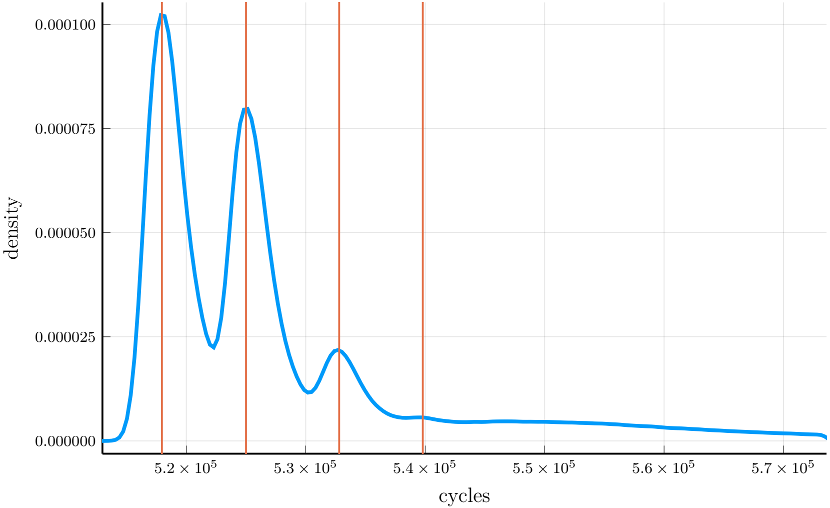


Figure 5: Timing distribution of decapsulation

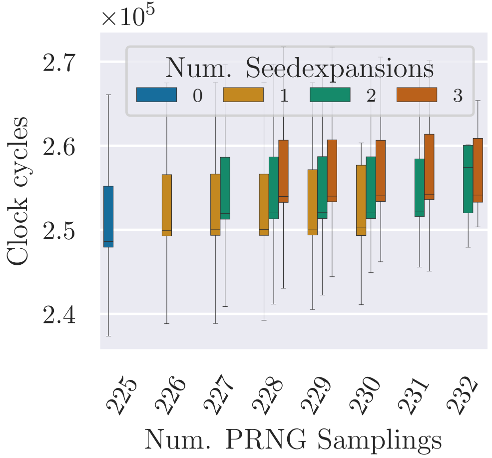


Figure 6: Timing distribution of decapsulation

Attack

Prerequisites

The message $\mathbf{m}$ that a ciphertext decrypts to determines the timing of the message

The ciphertext does not have to be valid

Prerequisites

The message $\mathbf{m}$ that a ciphertext decrypts to determines the timing of the message

The ciphertext does not have to be valid

$\rightarrow$ We can distinguish whether a modified ciphertext decrypts to a message $\mathbf{m}$ or $\mathbf{m}$'!

Recall: HQC encryption/decryption

Encrypt(pk, $\mathbf{m}$)

$\mathbf{e}, \mathbf{r_1}, \mathbf{r_2} \leftarrow\$ \mathcal{R}$ with $\omega(\mathbf{e}) = \omega_e$ and $\omega(\mathbf{r_1}) = \omega(\mathbf{r_2}) = \omega_r$

$\mathbf{u} = \mathbf{r_1} + \mathbf{h} \cdot \mathbf{r_2}$

$\mathbf{v} = \mathbf{mG} + \mathbf{s} \cdot \mathbf{r_2} + \mathbf{e}$

return $c = (\mathbf{u}, \mathbf{v})$

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

return $\mathcal{C}.\,\mathrm{Decode}(\underbrace{\mathbf{v} - \mathbf{u} \cdot \mathbf{y}}_{\mathbf{mG}-\mathbf{y}})$

Set $\mathbf{r_1}$ to $\mathbf{1}$ and $\mathbf{r_2}$ and $\mathbf{e}$ to $\mathbf{0}$ error is secret key!

Recover the error of the ciphertext to ✨🎉win🎉✨.

Additionally: we can add any extra error $\mathbf{e}'$ we want, for a combined error of $\mathbf{e}' - \mathbf{y}$.

Using the distinguisher

Recall: ciphertexts do not have to be valid

Assume $\texttt{timing}(c_1) \neq \texttt{timing}(c_2)$

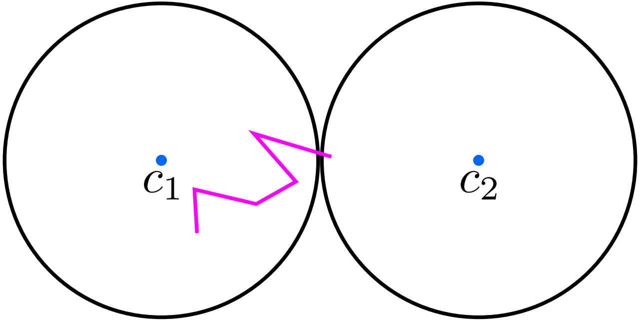


Figure 7: Random walk in ambient space $\mathbb{F}_2^n$ (symbolic image)

Attack[HGS99]

Flip bits until timing changes

Flip bits back to determine if they are an error

Repeat, take a majority vote

Evaluation

6096 attacks performed

Success rate: 87%

Among failed attacks: 86% terminated with less than 20 incorrect bits

866,143 idealized oracle calls (median)

BIKE Side-Channel and Attack

Bit Flipping Key Encapsulation (BIKE)

Algorithm 10:

BIKE.KeyGen

Input: $\cdot$

Output: $\mathsf{sk} = (\mathbf{h_0}, \mathbf{h_1}, \sigma)$

$\mathsf{pk} = \mathbf{h} \in \mathcal{R}$

$(\mathbf{h_0}, \mathbf{h_1}) = \mathsf{Sample}(\mathcal{H}_w)$

$\mathbf{h} = \mathbf{h_1}\mathbf{h_0}^{-1}$

$\sigma = \mathsf{Sample}(\mathcal{M})$

$\mathsf{sk} = (\mathbf{h_0}, \mathbf{h_1}, \sigma)$

$\mathsf{pk} = \mathbf{h}$

BIKE

Algorithm 11:
BIKE.Encaps

Input: $\mathsf{pk} = \mathbf{h}$
Output: K, c

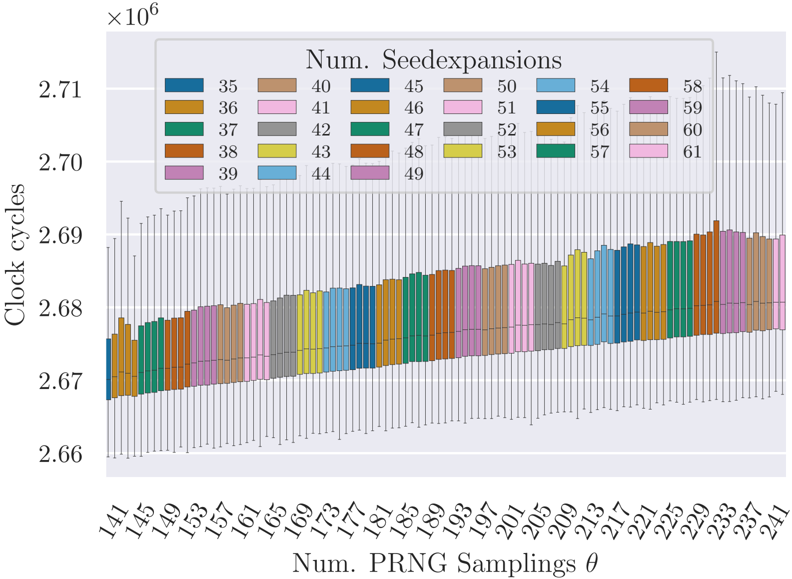
$m = \mathsf{Sample}(\mathcal{M})$
$(\mathbf{e_0}, \mathbf{e_1}) = \mathsf{H}(m)$
$c = (\mathbf{e_0} + \mathbf{e_1}, m \oplus \mathsf{L}(\mathbf{e_0}, \mathbf{e_1}))$
$K = \mathsf{K}(m, c)$

Algorithm 12:
BIKE.Decaps

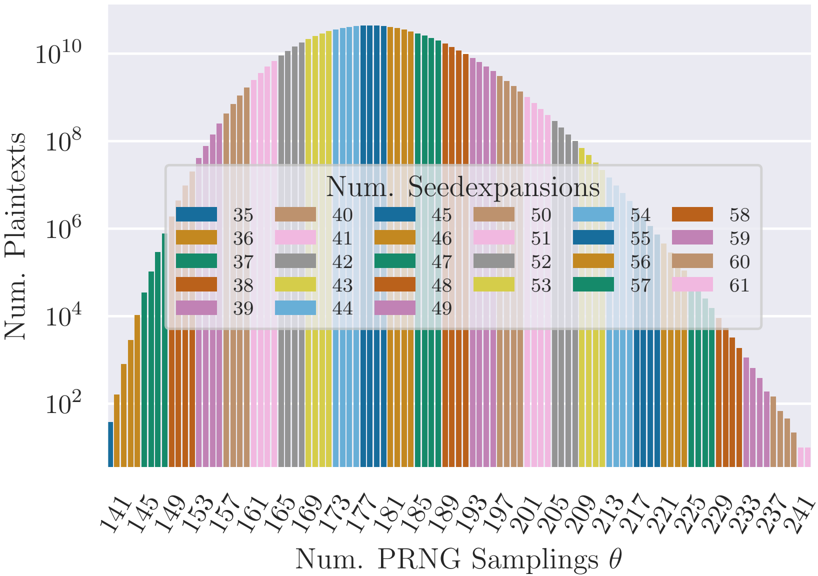
Input: $\mathsf{sk} = (\mathbf{h_0}, \mathbf{h_1}, \sigma)$
$c = (\mathbf{c_0}, c_1)$
Output: K

$\mathbf{e}' = \mathsf{Decode}(\mathbf{c_0}\mathbf{h_0}, \mathbf{h_0}, \mathbf{h_1})$
$m' = c_1 \oplus \mathsf{L}(\mathbf{e}')$
if $\mathbf{e}' = \mathsf{H}(m')$ **then**
    $K = \mathsf{K}(m', c)$
else
    $K = \mathsf{K}(\sigma, c)$
end

BIKE Side-Channel



BIKE Rare Messages



BIKE Attack

Reuse [GJS16] attack and [NJW18]

- Observation: if the distance of an error ocurrs in the secret key, it lowers the decryption failure rate
- Recover distance spectrum of the secret key with side-channel
- Recover the secret key from the distance spectrum using a recursive-backtracking algorithm

Recover Distance Spectrum using the Side-Channel

Simplest version:

Ciphertext with rare timing behavior + added noise

Send ciphertext to timing oracle, check whether decoding failure occurred.

Derive whether a cyclic distance d occurs in the secret key based on the decoding failure rate.

Countermeasures

Constant-Time Random Number Generation

Remove inner rejection sampling:

Sample a large number in steps, reduce modulo n

Constant Number of Iterations

Determine a sufficient number of outer rejection sampling iterations.

“Sufficient”: will not require more iterations with overwhelming probability.

Perform fixed number of iterations.

End Result

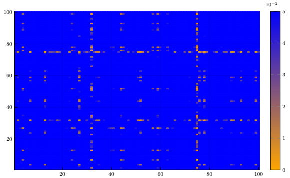


Figure 8: Fixed version

But: heavy performance hit: +29% in cycle count.

Interesting alternative approaches: Constant-time and time-efficient Fisher-Yates[3]

[3] Nicolas Sendrier. "Secure Sampling of Constant-Weight Words -Application to BIKE". In: *eprint Archive* (2021). URL: https://eprint.iacr.org/2021/1631.

References

[Agu+16] Carlos Aguilar, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. *Efficient Encryption from Random Quasi-Cyclic Codes*. Cryptology ePrint Archive, Report 2016/1194. https://eprint.iacr.org/2016/1194. 2016.

[Car+20] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jurjen Bos, Jean-Christophe Deneuville, Jérôme Lacan, Edoardo Persichetti, Jean-Marc Robert, Pascal Véron, Philippe Gaborit, Arnaud Dion, and Gilles Zémor. "Hamming Quasi-Cyclic (HQC) Third Round Version Updated Version 10/01/2020". In: (2020-10-01), p. 47. URL: https://pqc-hqc.org/doc/hqc-reference-implementation_2020-10-01.zip (visited on 2020-11-10).

[GJN20] Qian Guo, Thomas Johansson, and Alexander Nilsson. *A key-recovery timing attack on post-quantum primitives using the Fujisaki-Okamoto* *transformation and its application on FrodoKEM*. Cryptology ePrint Archive, Report 2020/743. https://eprint.iacr.org/2020/743. 2020.

[GJS16] Qian Guo, Thomas Johansson, and Paul Stankovski. *A Key Recovery Attack on MDPC with CCA Security Using Decoding Errors*. 858. 2016. URL: https://eprint.iacr.org/2016/858 (visited on 2020-11-04).

[HGS99] Chris Hall, Ian Goldberg, and Bruce Schneier. "Reaction Attacks against several Public-Key Cryptosystems". In: *ICICS 99: 2nd International Conference on Information and Communication Security*. Ed. by Vijay Varadharajan and Yi Mu. Vol. 1726. Lecture Notes in Computer Science. Sydney, Australia: Springer, Heidelberg, Germany, 1999, pp. 2–12.

[NJW18] Alexander Nilsson, Thomas Johansson, and Paul Stankovski Wagner. "Error Amplification in Code-based Cryptography". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2019.1 (2018). https://tches.iacr.org/index.php/TCHES/article/view/7340,

pp. 238–258. ISSN: 2569-2925. DOI:
10.13154/tches.v2019.i1.238-258.

[PT19] Thales Bandiera Paiva and Routo Terada. "A Timing Attack on the HQC Encryption Scheme". In: *SAC 2019: 26th Annual International Workshop on Selected Areas in Cryptography*. Ed. by Kenneth G. Paterson and Douglas Stebila. Vol. 11959. Lecture Notes in Computer Science. Waterloo, ON, Canada: Springer, Heidelberg, Germany, 2019, pp. 551–573. DOI: 10.1007/978-3-030-38471-5_22.

[Sen21] Nicolas Sendrier. "Secure Sampling of Constant-Weight Words -Application to BIKE". In: *eprint Archive* (2021). URL: https://eprint.iacr.org/2021/1631.

[WT+19] Guillaume Wafo-Tapa, Slim Bettaieb, Loic Bidoux, Philippe Gaborit, and Etienne Marcatel. *A Practicable Timing Attack Against HQC and its*

Countermeasure. Cryptology ePrint Archive, Report 2019/909. https://eprint.iacr.org/2019/909. 2019.

Don't Reject This: Key-Recovery Timing Attacks Due to Rejection-Sampling in HQC and BIKE

Qian Guo[3], Clemens Hlauschek[1,5], Thomas Johansson[3],
Norman Lahr[2], Alexander Nilsson[3,4], and **Robin Leander Schröder**[1]

September 19, 2022

[1]Technische Universität Wien, Austria
[2]Fraunhofer SIT, Darmstadt, Germany
[3]Lund University, Lund, Sweden
[4]Advenica AB, Malmö, Sweden
[5]RISE GmbH, Wien, Austria

Bonus Slides

Attack against RS/RM version

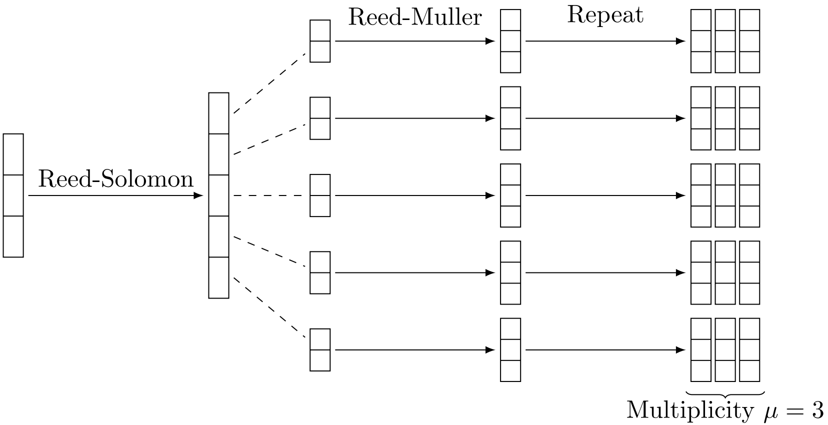


Figure 9: RS/RM Concatenated Code

Optimized Attack

Idea [WT+19]

Exploit the structure of the code generated by $\mathbf{G}$.

The public code $\mathcal{C}$ is either:

- a Bose-Chaudhuri-Hocquenghem (BCH) code tensored with a repetition code
- a Reed-Solomon (RS) code concatenated with a Reed-Muller (RM) code

Idea

We exploit the BCH/Repetition code version.

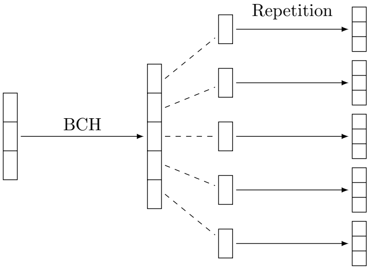


Figure 10: BCH/Repetition Tensor Code

Idea: corrupt δ BCH code blocks s.t. 1 more corruption will cause decoding failure

Then determine out the error in the repetition code block.

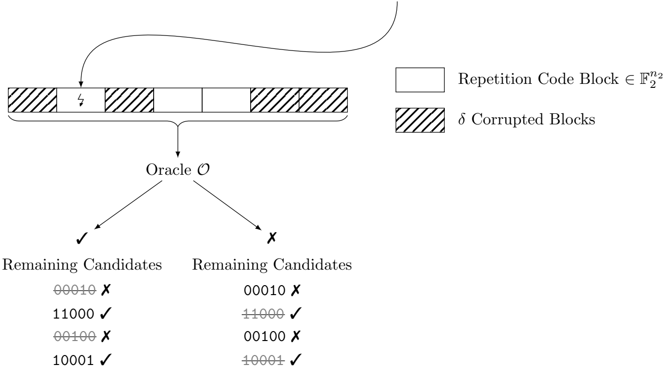
Candidate		Action 1		Decode	Candidate		Action 2		Decode
00010	$\oplus$	11011	=	11001 → 1 ✗	00010	$\oplus$	01000	=	01010 → 0 ✓
11000	$\oplus$	11011	=	00011 → 0 ✓	11000	$\oplus$	01000	=	10000 → 0 ✓
00100	$\oplus$	11011	=	11111 → 1 ✗	00100	$\oplus$	01000	=	01100 → 0 ✓
10001	$\oplus$	11011	=	01010 → 0 ✓	10001	$\oplus$	01000	=	11001 → 1 ✗

min(2✓, 2✗) = 2   min(3✓, 1✗) = 1

$\max(2, 1) = 2$

↓

Select Action 1



Evaluation

163 attacks performed

Success rate: 96.7%

Among failed attacks: less than 4 bits incorrect

19,942 idealized oracle calls (median)

Recover Distance Spectrum using the Side-Channel

Simplest version:

Construct a ciphertext with a message that has a rare timing-behavior and add an error to get close to the decoding limit[4].

Send ciphertext to timing oracle, check whether decoding failure occurred.

Derive whether a cyclic distance d occurs in the secret key based on the decoding failure rate.

For each cyclic distance d in the error:

If decoding success: increment observed$_d$.
If decoding failure: increment failed$_d$.

For each distance d, compute the empirical decoding failure rate, and estimate the multiplicity of the distance based on that.

[4] Ciphertext does not have to be valid!

Distance Spectrum of a Vector

Vector $\mathbf{v}$, length r.[5]

Multi-set of cyclic distances between set bits in vector $\mathbf{v}$.

$$\mathbf{v} = 100001001$$

$$D(\mathbf{v}) = \{\}$$

[5] Graphics heavily inspired by https://youtu.be/Gm--Sm_wJ2w

Distance Spectrum of a Vector

Vector $\mathbf{v}$, length r.[5]

Multi-set of cyclic distances between set bits in vector $\mathbf{v}$.

$$\mathbf{v} = 100001001$$

$$D(\mathbf{v}) = \{1\}$$

[5] Graphics heavily inspired by `https://youtu.be/Gm--Sm_wJ2w`

Distance Spectrum of a Vector

Vector $\mathbf{v}$, length r.[5]

Multi-set of cyclic distances between set bits in vector $\mathbf{v}$.

$$\mathbf{v} = 100001001$$

$$D(\mathbf{v}) = \{1, 4\}$$

[5] Graphics heavily inspired by `https://youtu.be/Gm--Sm_wJ2w`

Distance Spectrum of a Vector

Vector $\mathbf{v}$, length r.[5]

Multi-set of cyclic distances between set bits in vector $\mathbf{v}$.

$$\mathbf{v} = 100001001$$

$$D(\mathbf{v}) = \{1, 3, 4\}$$

[5] Graphics heavily inspired by `https://youtu.be/Gm--Sm_wJ2w`

Distances in the error affect the decoding failure rate

Satisfied parity checks during decoding[6]:

$$\mathbf{h} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \qquad s = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{e} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

[6]Graphics heavily inspired by `https://youtu.be/Gm--Sm_wJ2w`

Recover Secret Key from Distance Sprectrum

Greedy recursive-backtracking algorithm:

Start with empty vector $\mathbf{h} = \mathbf{0}^r$

- Check if already done (w bits already set, and $\mathbf{h}$ is the secret key)
- For each bit position i
  - if all distances to i exist in the distance spectrum
    - set bit i, and recurse