Profiling Side-Channel Attack on HQC Polynomial Multiplication Using Machine Learning Methods

Tomáš Rabas, Jiří Buček, Vincent Grosso, Karolína Zenknerová, Róbert Lórencz

Czech Technical University in Prague, FIT
NCISA CZ
Université Jean Monnet Saint-Etienne, CNRS

CASCADE
April, 2025

Hamming Quasi-Cyclic (HQC) Algorithm

CPA secure PKE based on quasi-cyclic codes

IND-CCA2 secure **KEM** based on (modified) Fujisaki-Okamoto transformation

Submission into **NIST PQC** (2017)

4th round – chosen to be the **new FIPS standard** (March 2025) (from BIKE, Classic McEliece, HQC, and SIKE :))

Attacker Model

KEMs in a ephemeral setting (each message new key-pair)
$\Rightarrow$ **single-trace** scenario

profiling attack with random (not chosen) inputs (keys and ciphertexts)

Experiments done on **ChipWhisperer-Lite**, ARM Cortex M4
– small SRAM $\Rightarrow$ reduced parameters from 17669 bits (level 1) $\rightarrow$ 1234 bits of the private key

Decryption Algorithm of HQC PKE

Focus on the **polynomial multiplication** over $\mathbb{F}_2$

HQC Decryption algorithm:

Require: private key $sk = (x, y)$, ciphertext $c = (u, v)$
Ensure: plaintext m
1: $m = \mathcal{C}.\texttt{Decode}(v - \boxed{u \cdot y})$
2: **return** m

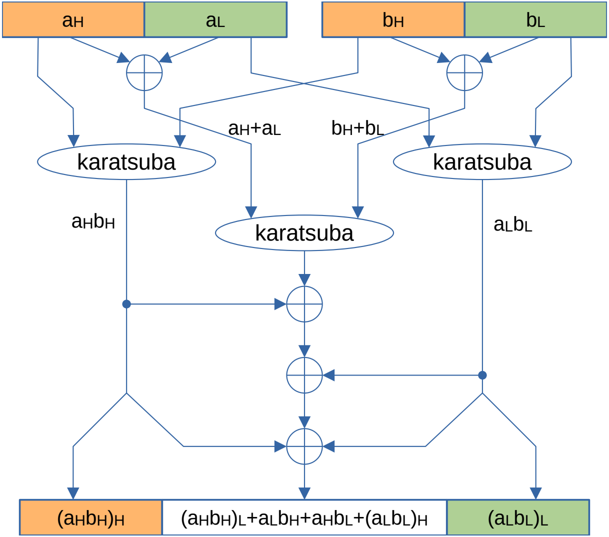
Target Implementation

"additional implementation" from submission package to NIST
– the only pure C code

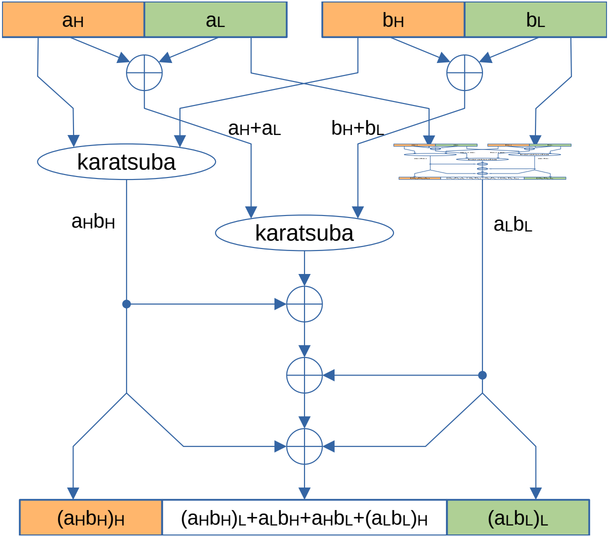
Implements recursive **Karatsuba** algorithm with last step for 64-bit inputs denoted as **base_mul** (constant-time and resistant to cache-timing attacks, using small precomputed table)

The implementation was further adopted (with minor modifications) by various crypto libraries (**PQClean** by Open Quantum Safe, BouncyCastle for `Java` and `C#`). Similar version also is used in PQC algorithm **BIKE**.

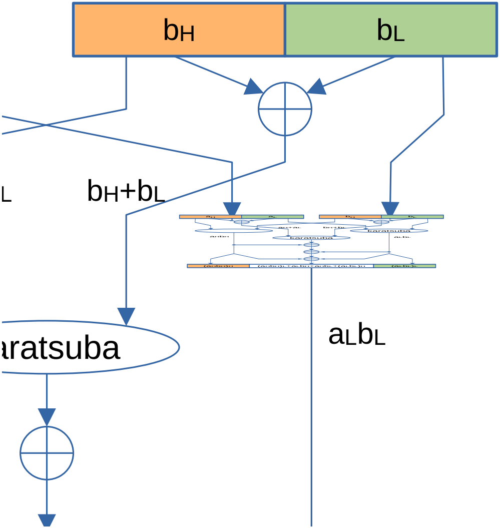
Karatsuba is Recursive



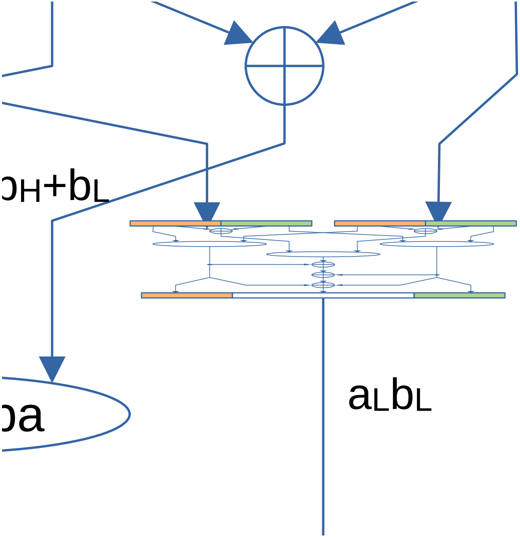
Karatsuba is Recursive



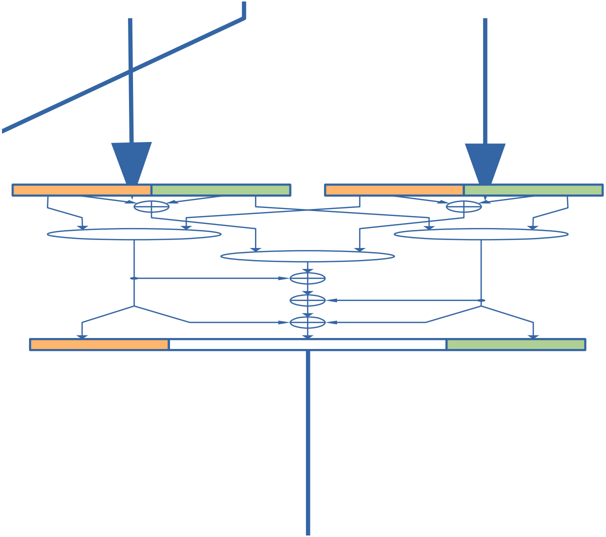
Karatsuba is Recursive



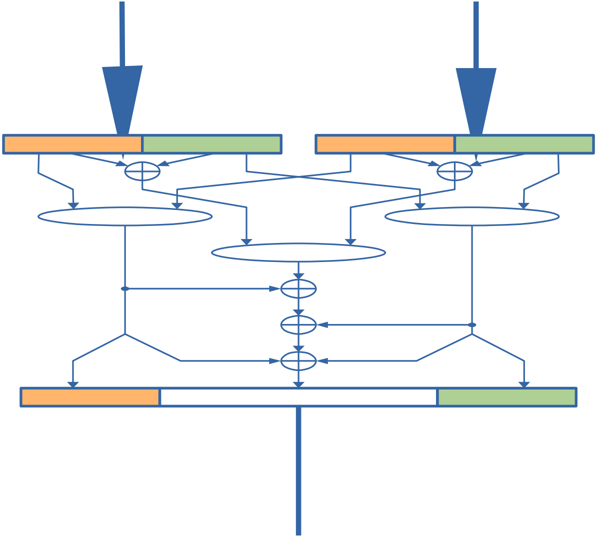
Karatsuba is Recursive



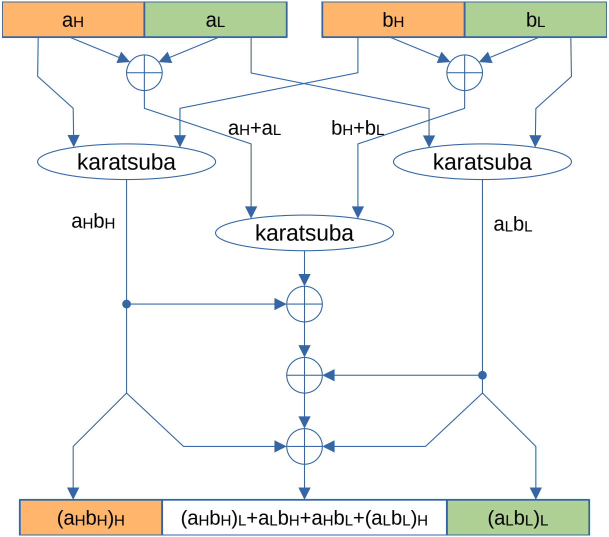
Karatsuba is Recursive



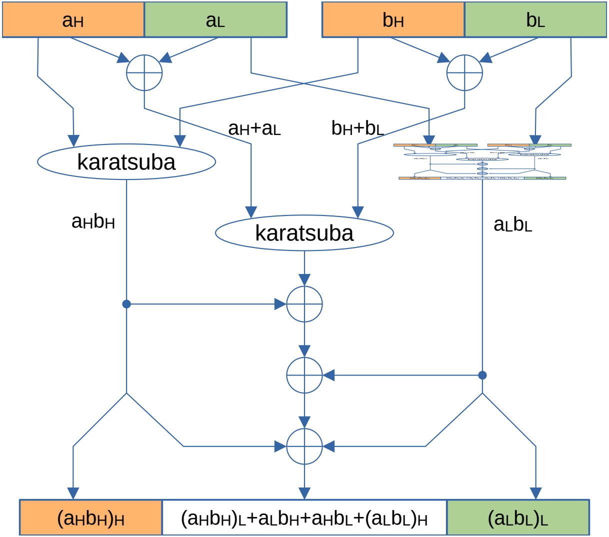
Karatsuba is Recursive



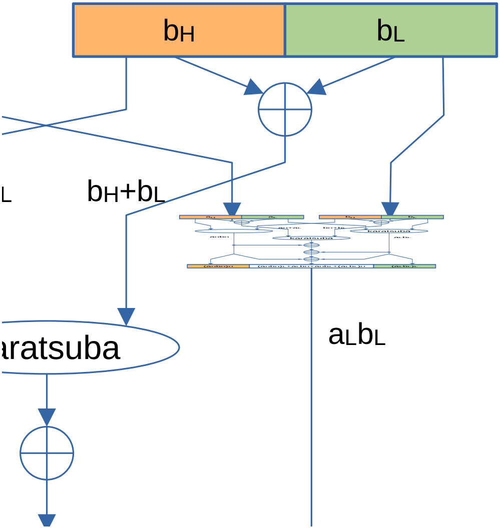
Karatsuba is Recursive (next level)



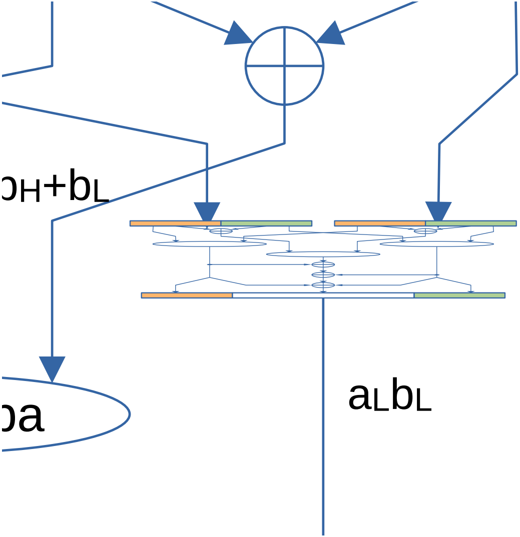
Karatsuba is Recursive (next level)



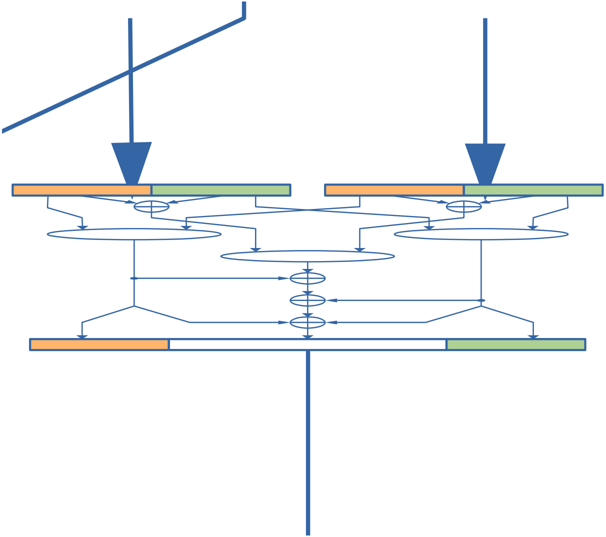
Karatsuba is Recursive (next level)



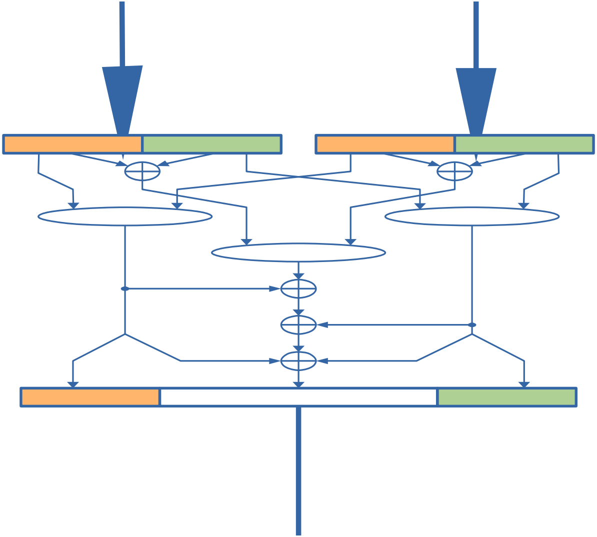
Karatsuba is Recursive (next level)



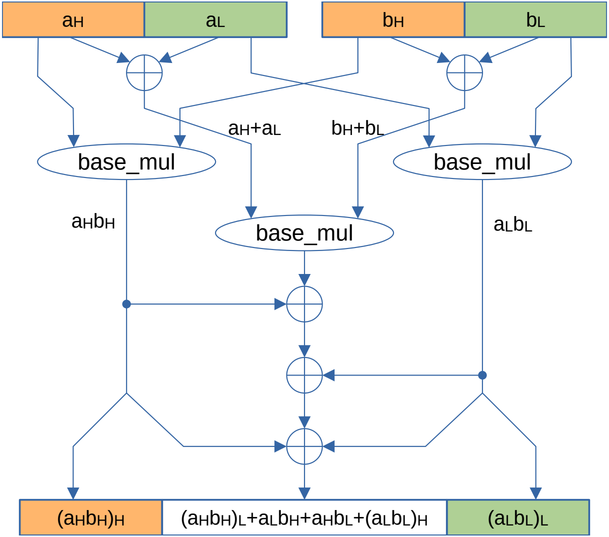
Karatsuba is Recursive (next level)



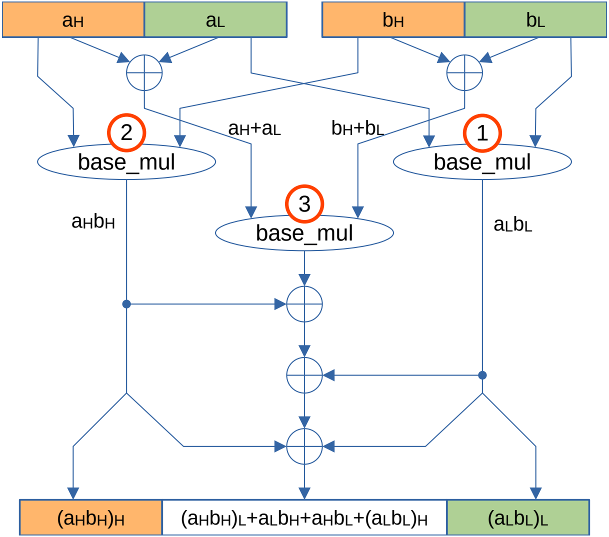
Karatsuba is Recursive (next level)



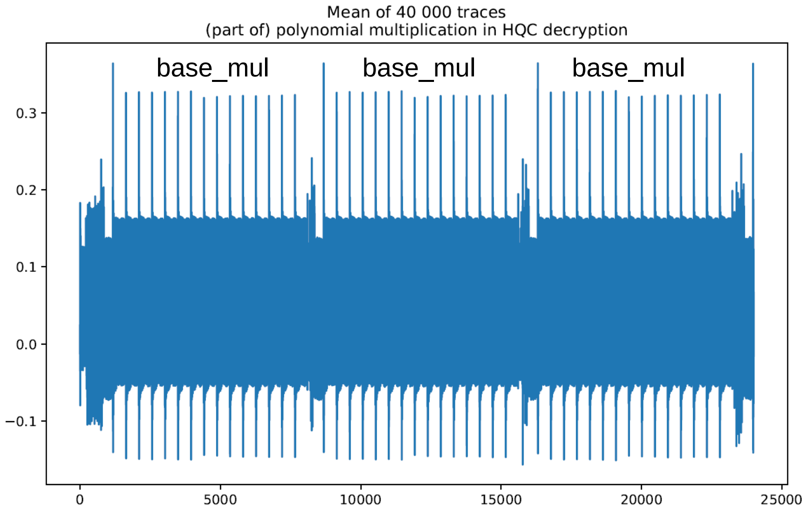
Karatsuba is Recursive (base case)



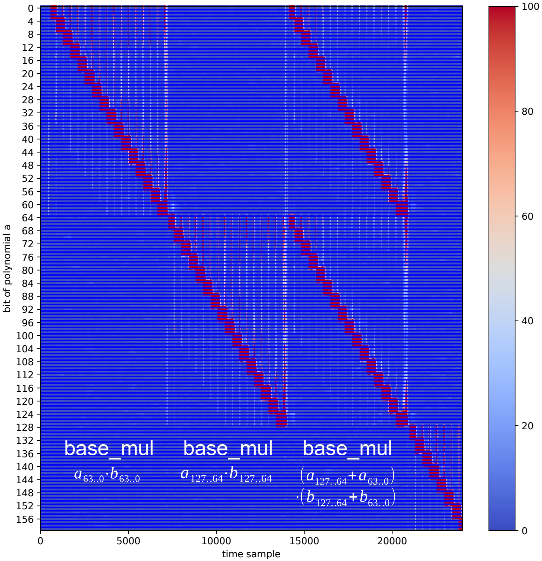
Karatsuba is Recursive (base case)



Traces from ChipWhisperer-Lite



Heat Map of F-statitics (Welch ANOVA)



Attack Scheme

3 datasets:

- 40 000 traces for training (profiling)
- 10 000 traces for validation (profiling)
- **20 000** traces for testing (single-trace attacks)

For each key bit:

- Selecting POIs using Welch ANOVA statistical test
- Training ML-models on these POIs as atributes (using hyper-parameter tuning)
- Test the best model on independent dataset of 20 000 traces.

$\Rightarrow$ we get a ML-model for each key bit.

Results

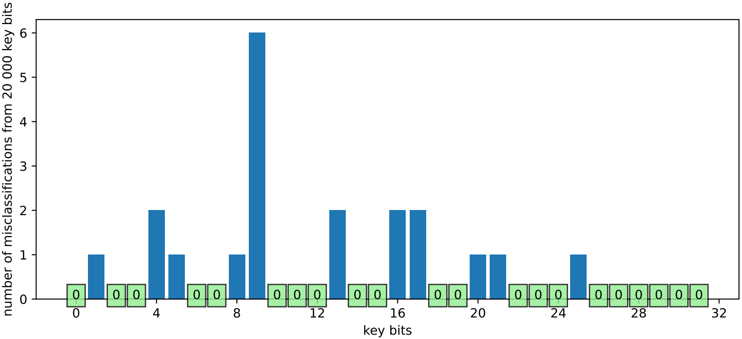


Figure: Results of the attack phase for recovering the individual bits from the private key. The height of the columns corresponds to the misclassifications of individual bits. Most of the bits have been recovered with perfect accuracy. The worst result was for the 9th bit with 6 errors of 20 000, i.e. with the accuracy of 0.9997.

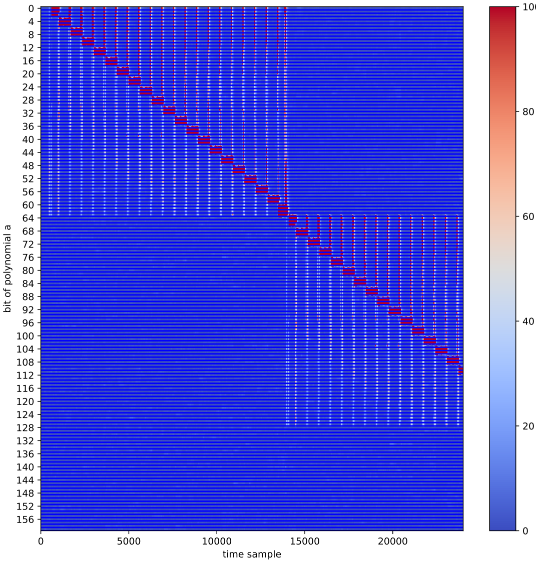
Proposed Countermeasures for *base_mul*

2 proposed countermeasures (details in the paper):

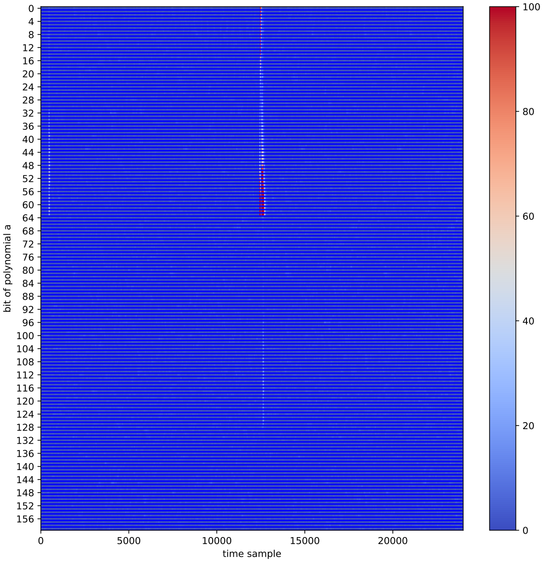
- self-unmasking implementation
  + costs just 20 bits of randomness, – masks only every third bit.
- full mask implementation with two shares
  – costs 64 bits of randomness, + masks all bits

Not extended to full Karatsuba (future work?) $\rightarrow$ we can see leaking bits after unmasking in the end of *base_mul*

Self-unmasking implementation – Heat Map



Full-mask (64b) implementation – Heat Map



Final Results and Comparisons

	Additional implementation	Self-unmasking	Full-mask
# clock cycles	6.5k	14k	14k
random bits	0	20	64
Max F-statistic	198 607	41 226	4 381
Mean bit-error ϵ_{bit}	3.8e-05	0.008	0.048
Naive complexity ($n = 1234$)	10.27	80.86	337.69
Success rate ($n = 1234$)	0.954	4.5e-05	8.22e-27
Naive complexity ($n = 17\,669$)	14.11	1194.30	4868.05
Success rate ($n = 17\,669$)	0.511	7.4e-63	3.2e-374

Conclusion

- First single-trace attack on HQC targeting private key
- Recovering private key with 51.1%
- Proposed two countermeasures

Thanks for Attention

This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS23/211/OHK3/3T/18 funded by the MEYS of the Czech Republic, and also by the Project Barrande No. 8J23FR012, funded by the MEYS of the Czech Republic.