Side Channel Information Set Decoding using Iterative Chunking

Plaintext Recovery from the “Classic McEliece” Hardware Reference Implementation

Norman Lahr [1] Ruben Niederhagen [2] Richard Petri [1] Simona Samardjiska [3]

December 08, 2020

[1] Fraunhofer SIT, Darmstadt, Germany

[2] University of Southern Denmark, Odense, Denmark

[3] Radboud Universiteit, Nijmegen, The Netherlands

Outline

Motivation and Contribution

- Classic McEliece is one of the finalists for KEMs in the 3rd round of the NIST PQC competition.
- It is based on the Niederreiter cryptosystem with binary Goppa codes.
- Classic McEliece is considered to be a conservative choice.
- Defined by the code length n, the code dimension k, the guaranteed error-correction capability t, and the field size m.
- NIST parameter sets: `kem/mceliece348864`, `-460896`, `-6688128`, `-6960119`, and `-8192128`.

Key generation

Private key $g(x) \rightarrow$ random degree-t Goppa polynomial over GF(2^m).
$(\alpha_0, \alpha_1, \ldots \alpha_{n-1}) \rightarrow$ random size-n support list.

Public key $\mathsf{H} \rightarrow$ corresponding binary parity check matrix.

Encryption

Generate a random binary weight-t error vector e of length n.
Compute the syndrome by

$$\mathsf{s} = \mathsf{H} \cdot \mathsf{e}$$

Decryption

Decode the syndrome, get the error-locator polynomial

$$\sigma(x) = \mathit{Decode}(\mathsf{s})$$

Reconstruct the error $\mathsf{e}_i = \{1 \text{ if } \sigma(\alpha_i) = 0 \textit{ or } 0 \text{ otherwise}\}$.

- The security of the Niederreiter scheme relies on the decoding problem:

  Compute e *given* H *and the ciphertext* $\mathsf{s} = \mathsf{H} \cdot \mathsf{e}$, *where* e *is a uniform random weight-*t *vector of length* n.
- The fastest attacks known to break the decoding problem use Information Set Decoding (ISD).
- Current approaches are *Prange*, *Stern*, *May-Meurer-Thomae* (MMT), and *Becker-Joux-May-Meurer* (BJMM).

  $\Rightarrow$ The cost for the Classic McEliece parameters submitted to NIST are infeasible, e.g., $2^{277.6}$ operations for `kem/mceliece8192128` using MMT.

- Reaction-based attacks are known, feasible attacks to code-based cryptosystems.
- Assumpiton: an attacker has access to a decryption oracle which returns success or failure for a given input.
- Idea: reactions caused by prepared input lead to secret key or plaintext recovery.
- If not explicitly, side-channel information, like timing or power consumption, can be used as feedback from a decryption oracle.
- E.g., a decoder leaks exploitable information if its decoding capacity exceeds.

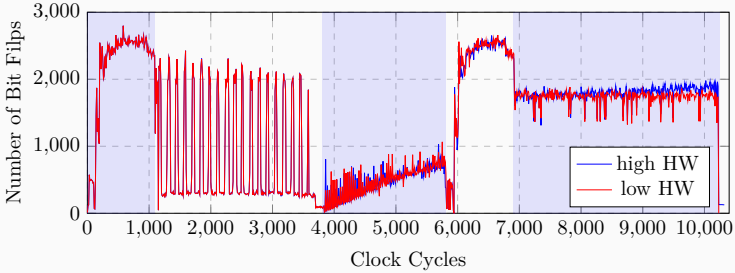
  $\Rightarrow$ Current plaintext recovery approaches, like Shoufan et al., ICISC 2009, reveal bits individually and require a high amount of queries (code length, e.g., 8192).

- Introduction of iterative chunking for plaintext recovery on Classic McEliece.
  - Reaction-based attack strategy with cumulative queries.
  - We determined the optimal parameters and strategy.
  - It significantly reduces the attack effort, e.g., by ~90% for `kem/mceliece8192128`.
- Further improvement by including information set decoding algorithms.
  - We estimate the trade-off between required queries and computational power, e.g., spending 2^{40} operations, further reduction is ~12% for `kem/mceliece8192128`.
- Simulated evaluation and practical side-channel attack using EM leakage on the reference FPGA design.

Reaction-based Attack on Niederreiter

- Access to a decryption oracle.
  - → Niederreiter FPGA implementation of Wang et al., PQCrypto 2018.
- An unlimited number of decryption queries.
- Goal is a plaintext recovery.
  - → No access to the decryption output.
- Presence of in-/direct information leakage.
  - Shoufan et al. mount a timing attack on a McEliece design using the Patterson decoder.
  - → Wang et al. uses a Berlekamp-Massey decoder with a constant-time implementation.

- The design shows a power leakage when exhausting the decoder's capacity t.
- If the syndrome contains $>t$ errors, the computation of an error-locator polynomial fails and the result is a random polynomial.
- The corresponding error vector shows a very low Hamming weight (≈ 0).
- A power simulation reveals a (simple) leakage at the error vector construction.



$$H \cdot e = s$$

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 1 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \\ 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix} =$$

$$H \cdot e = s$$

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 1 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \\ 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix} =$$

$$H \cdot e = s$$

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 1 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \\ 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$H \cdot e = s$$

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 1 & 0 & \cdots & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \\ 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$H \cdot e = s'$$

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 1 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \\ 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

How to recover the entire error vector from a syndrome?

- Idea: iteratively test the error vector positions.
- Add a corresponding column of the binary parity check matrix to the syndrome.
- Ask the oracle whether the capacity t is exceeded.
  - A redundant error ($\mathbf{e}_i = 1$) cancels out.
  - An additional error ($\mathbf{e}_i = 0$) leads to an overflow.

  $\Rightarrow$ Effort: n queries.

$\mathbf{e} \leftarrow (0, \ldots, 0)$

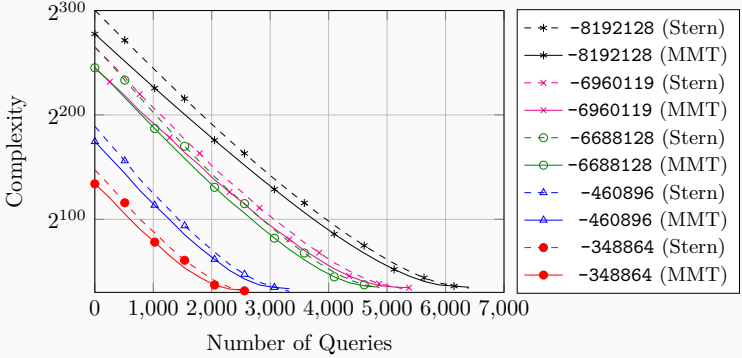
for $i \leftarrow 1$ to n do
  $\mathbf{s}' \leftarrow \mathbf{s} \oplus \mathbf{H}_i$
  if Oracle($\mathbf{s}'$) $=$ *true* then
    $\mathbf{e}_i \leftarrow 1$
  end
end

Reduce the Number of Queries with Information Set Decoding

- The subsequent application of information set decoding can save queries:
  1. Recover just a subset of the positions in $\mathbf{e}$ using a reaction-based attack.
  2. Recover the remaining positions using information set decoding.
- A trade-off between deployed computational power and needed queries is required.
- The default cost of k queries leads to an uniquely solvable linear equation.

  $\rightarrow$ A Gaussian elimination causes a negligible effort.

Impact of Information Set Decoding



Iterative Chunking

Chunking with $\beta = 2$

What happens if we ask for two error positions at once?

- If we add two distinct columns of $\mathbf{H}$ (chunk) there are four different cases.
- The density of '1's is less than 2.1%.
- In the most cases $\mathbf{e}$ consists of '0's.
- We can detect this case!
- Introduce chunk size β, here 2.

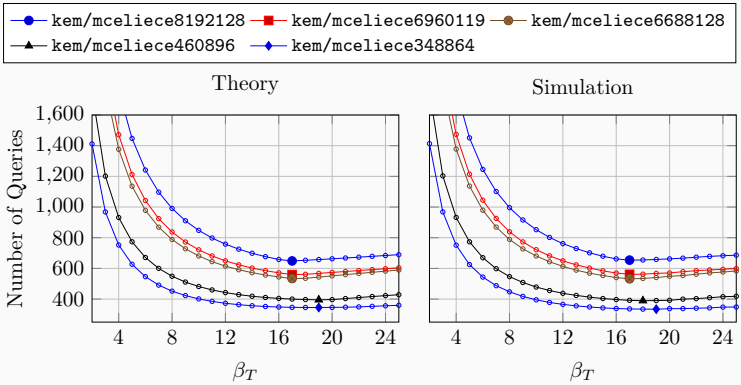
$(\mathbf{e}_i, \mathbf{e}_j)$	$(\mathbf{e}'_i, \mathbf{e}'_j)$	$w(\mathbf{e}')$	Oracle
$(0,0)$	$(1,1)$	$w+2$	false
$(0,1)$	$(1,0)$	w	true
$(1,0)$	$(0,1)$	w	true
$(1,1)$	$(0,0)$	$w-2$	true

What happens if we iteratively increase the input chunk size β?

		s		$\mathsf{s}' = \mathsf{s} \oplus \mathsf{H}'_l\ (\mathsf{e}_l = 1)$	
$(\mathsf{e}_i, \mathsf{e}_j, \mathsf{e}_k)$	$(\mathsf{e}'_i, \mathsf{e}'_j, \mathsf{e}'_k)$	$w(\mathsf{e}')$	Oracle	$w(\mathsf{e}') - 1$	Oracle
$(0,0,0)$	$(1,1,1)$	$w+3$	false	$w+2$	false
$(0,0,1)$	$(1,1,0)$	$w+1$	false	w	true
$(0,1,0)$	$(1,0,1)$	$w+1$	false	w	true
$(1,0,0)$	$(0,1,1)$	$w+1$	false	w	true
$(1,1,0)$	$(0,0,1)$	$w-1$	true	$w-2$	true
$(1,0,1)$	$(0,1,0)$	$w-1$	true	$w-2$	true
$(0,1,1)$	$(1,0,0)$	$w-1$	true	$w-2$	true
$(1,1,1)$	$(0,0,0)$	$w-3$	true	$w-4$	true

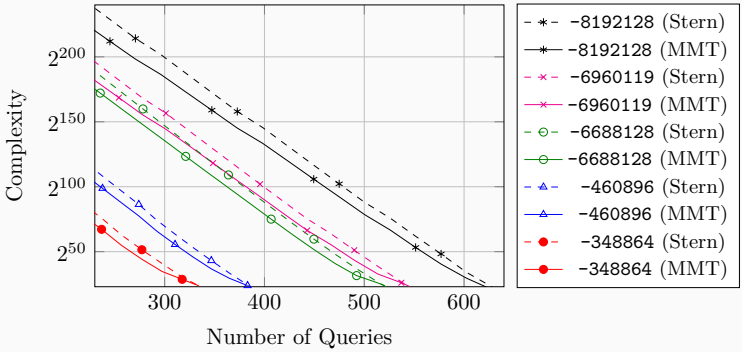
- Iteratively increment the chunk size β if an additional '1' position is identified.
- Reduce the number of errors in the syndrome with known '1' positions.
- Identify the exact position of a '1' in a *'high' chunk* by subchunking (devide & conquer).
- If β gets larger the possibility of a 'high' chunk increases.
  - Leads to more subchunking.
  - There is an optimal β_T which minimizes the number of queries.
- If $\beta = \beta_T$ collect the 'high' chunks in a bucket (e.g., size $n - k$).
- Solve the bucket at the end with information set decoding.

What is the optimal chunk size?



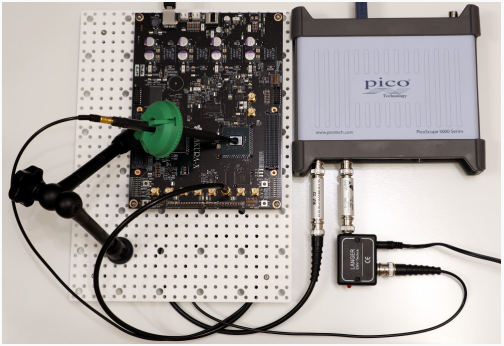
Time vs. Queries: Optimal Iterative Chunking with ISD

Estimate the computational cost to do a trade-off to the number of queries.



Evaluation

- We integrated the FPGA design to Sakura X, Xilinx Kintex 7, @24MHz.
- Langer EM probe (RF-U 5-2).
- Pico Scope 5244D MSO @500MS.
- Python scripts for ...
  - trace acquisition,
  - side-channel decryption oracle, and
  - iterative chunking strategy (same as for simulation).



- EM leakage of error vector construction exploitable with Welch's t-test.
- Requires two reference traces: T_{high}, T_{low}.
- Trace compression for alignment and speedup.
- Update of reference trace T'_{high} for better accuracy.

$$\text{Decrypt}(\mathbf{s}') \rightsquigarrow T_j$$

$$T'_j \leftarrow \text{Compress}(T_j)$$

$$p_\Delta \leftarrow \text{t-test}(T'_j, T'_{high}) - \text{t-test}(T'_j, T'_{low})$$

$$\textbf{return} \begin{cases} \mathit{true} & \mathit{if}\ p_\Delta > 0 \\ \mathit{false} & \mathit{otherwise} \end{cases}$$

Required Number of Queries

kem/ mceliece-	Approach	Simulation/ Experiment avg.	Theory	
			plain	cost $\approx 2^{40}$
348864	$\beta = 1$	2722 $(k+2)$		
	$\beta_T = 19$	334.16	345.06	287.26
460896	$\beta = 1$	3362 $(k+2)$		
	$\beta_T = 18$	389.33	396.95	—
	$\beta_T = 19$	390.33	395.06	337.66
6688128	$\beta = 1$	5026 $(k+2)$		
	$\beta_T = 17$	532.11	534.14	470.91
6960119	$\beta = 1$	5415 $(k+2)$		
	$\beta_T = 17$	561.29	559.87	493.90
8192128	$\beta = 1$	6530 $(k+2)$		
	$\beta_T = 17$	653.97	648.66	576.96

Required Number of Queries

kem/ mceliece-	Approach	Simulation/ Experiment avg.	Theory	
			plain	cost $\approx 2^{40}$
348864	$\beta = 1$	2722 $(k+2)$		
	$\beta_T = 19$	334.16	345.06	287.26
460896	$\beta = 1$	3362 $(k+2)$		
	$\beta_T = 18$	389.33	396.95	—
	$\beta_T = 19$	390.33	395.06	337.66
6688128	$\beta = 1$	5026 $(k+2)$		
	$\beta_T = 17$	532.11	534.14	470.91
6960119	$\beta = 1$	5415 $(k+2)$		
	$\beta_T = 17$	561.29	559.87	493.90
8192128	$\beta = 1$	6530 $(k+2)$		
	$\beta_T = 17$	653.97	648.66	576.96

Conclusion

- Iterative chunking reduces the number of queries by an order of magnitude, e.g., from 6530 to 654 queries (~90%) for `kem/mceliece8192128`.
- The additional use of information set decoding decreases the amount further, e.g., spending 2^{40} operations, reduces queries from 654 to 577 (~11.7%) for `kem/mceliece8192128`.
- Iterative chunking does not depend on a specific kind of information leakage.
- Broader application to other (code-based) schemes for further research.

Thank you for your attention!

Norman Lahr [1] Ruben Niederhagen [2] Richard Petri [1] Simona Samardjiska [3]

December 08, 2020

[1] Fraunhofer SIT, Darmstadt, Germany

[2] University of Southern Denmark, Odense, Denmark

[3] Radboud Universiteit, Nijmegen, The Netherlands