Error Amplification in Code-based Cryptography

Alexander Nilsson\textsuperscript{1,2} Thomas Johansson\textsuperscript{1} Paul Stankovski Wagner\textsuperscript{1}

August 27, 2019

\textsuperscript{1}Dept. of Electrical and Information Technology, Lund University, Sweden
\textsuperscript{2}Advenica AB, Malmö, Sweden
Background

Code-based Cryptography

Previous work

Attack Scenario

Contributions

The Chaining method

Generating $e_0$

Results

Amplification effect

Wrapping it up
Code-based Cryptography

- One of the major branches of cryptographic post-quantum research.
Code-based Cryptography

• One of the major branches of cryptographic post-quantum research.
• Security based on hardness of decoding random linear codes.
• One of the major branches of cryptographic post-quantum research.
• Security based on hardness of decoding random linear codes.
• The McEliece cryptosystem from 1978, using binary Goppa codes, is still secure today.
• One of the major branches of cryptographic post-quantum research.
• Security based on hardness of decoding random linear codes.
• The McEliece cryptosystem from 1978, using binary Goppa codes, is still secure today.
• Large keys!
Quasi-Cyclic Medium Density Parity Check is a variant of the McEliece cryptosystem [Mis+12]:

- More compact keys by using cyclic structures in the key-matrices.
- Encryption simply: $c = mG + e$.
- Uses iterative bitflipping decoding in the decryption stage.
- Decryption Failure Rate (DFR) is non-zero.
Quasi-Cyclic Medium Density Parity Check is a variant of the McEliece cryptosystem [Mis+12]:

- More compact keys by using cyclic structures in the key-matrices.
Quasi-Cyclic Medium Density Parity Check is a variant of the McEliece cryptosystem [Mis+12]:

- More compact keys by using cyclic structures in the key-matrices.
- Encryption simply: $c \leftarrow mG + e$
Quasi-Cyclic Medium Density Parity Check is a variant of the McEliece cryptosystem [Mis+12]:

- More **compact keys** by using cyclic structures in the key-matrices.
- Encryption simply: $c \leftarrow mG + e$
- Uses iterative **bitflipping decoding** in the decryption stage
Quasi-Cyclic Medium Density Parity Check is a variant of the McEliece cryptosystem [Mis+12]:

- More compact keys by using cyclic structures in the key-matrices.
- Encryption simply: $c \leftarrow mG + e$
- Uses iterative bitflipping decoding in the decryption stage
- Decryption Failure Rate (DFR), is non-zero.
A \((n,r,w)\)-QC-MDPC code, is a linear code with an error correcting capability \(t\), length \(n\), codimension \(r\) and with a row weight \(w\) in the parity check matrix \(H\). Additionally we have that \(n = n_0r\).
A \( (n,r,w) \)-QC-MDPC code, is a linear code with an error correcting capability \( t \), length \( n \), codimension \( r \) and with a row weight \( w \) in the parity check matrix \( H \). Additionally we have that \( n = n_0 r \).

Suggested parameters for 80-bit security:

\[
n_0 = 2, \quad n = 9602, \quad r = 4801, \quad w = 90, \quad t = 84
\]
A \((n,r,w)\)-QC-MDPC code, is a linear code with an error correcting capability \(t\), length \(n\), codimension \(r\) and with a row weight \(w\) in the parity check matrix \(H\). Additionally we have that \(n = n_0 r\).

Suggested parameters for 80-bit security:

\[ n_0 = 2, n = 9602, r = 4801, w = 90, t = 84 \]

**Sparse!** \(\approx 99\) bits out of 100 are zero in \(H\).
The secret key $H \in \mathbb{F}^{r \times n}_2$ is constructed as

$$H = [H_0 | H_1 | \ldots | H_{n_0-1}],$$

where $H_i$ is a circulant $r \times r$ matrix.
The secret key $H \in \mathbb{F}_2^{r \times n}$ is constructed as

$$H = [H_0|H_1| \ldots |H_{n_0-1}],$$

where $H_i$ is a circulant $r \times r$ matrix.

For $n_0 = 2$, we get

$$H = \begin{bmatrix}
(h_{0,0} & h_{0,1} & \cdots & h_{0,r-1}) \\
(h_{0,r-1} & h_{0,0} & \cdots & h_{0,r-2}) \\
\vdots & \vdots & \ddots & \vdots \\
(h_{0,1} & h_{0,2} & \cdots & h_{0,0})
\end{bmatrix} \begin{bmatrix}
(h_{1,0} & h_{1,1} & \cdots & h_{1,r-1}) \\
(h_{1,r-1} & h_{1,0} & \cdots & h_{1,r-2}) \\
\vdots & \vdots & \ddots & \vdots \\
(h_{1,1} & h_{1,2} & \cdots & h_{1,0})
\end{bmatrix}$$
The secret key $H \in \mathbb{F}_2^{r \times n}$ is constructed as

$$H = [H_0|H_1| \ldots |H_{n_0-1}],$$

where $H_i$ is a circulant $r \times r$ matrix.

For $n_0 = 2$, we get

$$H = \begin{pmatrix}
    h_{0,0} & h_{0,1} & \cdots & h_{0,r-1} \\
    h_{0,r-1} & h_{0,0} & \cdots & h_{0,r-2} \\
    \vdots & \vdots & \ddots & \vdots \\
    h_{0,1} & h_{0,2} & \cdots & h_{0,0}
\end{pmatrix} \begin{pmatrix}
    h_{1,0} & h_{1,1} & \cdots & h_{1,r-1} \\
    h_{1,r-1} & h_{1,0} & \cdots & h_{1,r-2} \\
    \vdots & \vdots & \ddots & \vdots \\
    h_{1,1} & h_{1,2} & \cdots & h_{1,0}
\end{pmatrix}$$

Knowledge of $h_0$ (the first row of $H_0$) is sufficient for complete key recovery.
Public key $G \in \mathbb{F}_2^{(n-r) \times n}$ is constructed as follows:

$$G = \begin{pmatrix} I \\ (H_{n_0-1}^{-1} \cdot H_0)^T \\ (H_{n_0-1}^{-1} \cdot H_1)^T \\ \vdots \\ (H_{n_0-1}^{-1} \cdot H_{n_0-2})^T \end{pmatrix}$$
Public key \( G \in \mathbb{F}_2^{(n-r)\times n} \) is constructed as follows:

\[
G = \begin{pmatrix}
I \\
\end{pmatrix}
\begin{pmatrix}
(H^{-1}_{n_{0-1}} \cdot H_0)^T \\
(H^{-1}_{n_{0-1}} \cdot H_1)^T \\
\vdots \\
(H^{-1}_{n_{0-1}} \cdot H_{n_{0-2}})^T \\
\end{pmatrix}
\]

Encryption of plaintext \( m \in \mathbb{F}_2^{n-r} \) into \( c \in \mathbb{F}_2^n \) is given by:
Public key $G \in \mathbb{F}_2^{(n-r)\times n}$ is constructed as follows:

$$G = \begin{pmatrix} I & \begin{pmatrix} \left( H^{-1}_{n_0-1} \cdot H_0 \right)^T \\ \left( H^{-1}_{n_0-1} \cdot H_1 \right)^T \\ \vdots \\ \left( H^{-1}_{n_0-1} \cdot H_{n_0-2} \right)^T \end{pmatrix} \end{pmatrix}$$

Encryption of plaintext $m \in \mathbb{F}_2^{n-r}$ into $c \in \mathbb{F}_2^n$ is given by:

1. Generating random $e \in \mathbb{F}_2^n$ with Hamming weight, $\text{wt}(e)$, less than $t$. 
Public key $G \in \mathbb{F}_2^{(n-r) \times n}$ is constructed as follows:

$$G = \begin{pmatrix} I \\ \left( \begin{array}{c} (H_{n_0-1}^{-1} \cdot H_0)^T \\ (H_{n_0-1}^{-1} \cdot H_1)^T \\ \vdots \\ (H_{n_0-1}^{-1} \cdot H_{n_0-2})^T \end{array} \right) \end{pmatrix}$$

Encryption of plaintext $m \in \mathbb{F}_2^{n-r}$ into $c \in \mathbb{F}_2^n$ is given by:

1. Generating random $e \in \mathbb{F}_2^n$ with Hamming weight, $\text{wt}(e)$, less than $t$.
2. Computing $c \leftarrow mG + e$. 
To decrypt $c \in \mathbb{F}_2^n$ into $m \in \mathbb{F}_2^{n-r}$ we need a decoding algorithm, $\Psi_H$, with knowledge of $H$. 
To decrypt $c \in \mathbb{F}_2^n$ into $m \in \mathbb{F}_2^{n-r}$ we need a decoding algorithm, $\Psi_H$, with knowledge of $H$.

1. Decode $mG \leftarrow \Psi_H(mG + e)$
To decrypt $c \in \mathbb{F}_2^n$ into $m \in \mathbb{F}_2^{n-r}$ we need a decoding algorithm, $\Psi_H$, with knowledge of $H$.

1. Decode $mG \leftarrow \Psi_H(mG + e)$
2. Plaintext $m$ is first $(n - r)$ positions of $mG$. 

To decrypt \( c \in \mathbb{F}_2^n \) into \( m \in \mathbb{F}_2^{n-r} \) we need a decoding algorithm, \( \psi_H \), with knowledge of \( H \).

1. Decode \( mG \leftarrow \psi_H(mG + e) \)

2. Plaintext \( m \) is first \((n - r)\) positions of \( mG \).

The decoding algorithms \((\psi_H)\) are based on variants of the original Gallager’s bitflipping algorithm.
• QC-MPDC was previously shown *vulnerable* in [GJS16]$^1$.

---

• QC-MPDC was previously shown vulnerable in [GJS16].
• Key recovery is possible with 250-300 M ciphertexts for 80-bit security parameters.

---

• QC-MPDC was previously shown vulnerable in [GJS16].
• Key recovery is possible with 250-300 M ciphertexts for 80-bit security parameters.
• Attack against CCA secure QC-MDPC.

¹Qian Guo, Thomas Johansson and Paul Stankovski. ”A Key Recovery Attack on MDPC with CCA security Using Decoding Errors”. In: ASIACRYPT 2016
QC-MPDC was previously shown vulnerable in \cite{GJS16}.

Key recovery is possible with 250-300 M ciphertexts for 80-bit security parameters.

Attack against CCA secure QC-MDPC.

The authors discovered a correlation between the distance spectrums of the secret key and of non-decodeable error patterns.

\cite{GJS16} Qian Guo, Thomas Johansson and Paul Stankovski. ”A Key Recovery Attack on MDPC with CCA security Using Decoding Errors”. In: ASIACRYPT 2016
Distance spectrum \((D(\ldots))\): *wrapping* distances between two non-zero bits. The number in each counter counts the occurrence of a specific distance, or its *multiplicity*. 

**Distance Spectrums**

error pattern, \(e\): \[
\begin{array}{cccccc}
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1
\end{array}
\]

distance spectrum, \(D(e)\): \[
\begin{array}{ccccc}
1 & 1 & 2 & 1 & 1 \\
1 & 2 & 3 & 4 & 5
\end{array}
\]
Distance Spectrums

error pattern, $e$: \[1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\]

distance spectrum, $D(e)$: \[1\ 1\ 2\ 1\ 1\]

Distance spectrum ($D(\ldots)$): wrapping distances between two non-zero bits. The number in each counter counts the occurrence of a specific distance, or its multiplicity.

We want to find $D(h_0)$, the distance spectrum of the first row of $H_0$, the first part of the secret key $H$. 
A reaction attack against CCA secure QC-MDPC. [GJS16]
A reaction attack against CCA secure QC-MDPC. [GJS16]

0. Attacker: Initialize $i \leftarrow 0$. 
A reaction attack against CCA secure QC-MDPC. [GJS16]

0. Attacker: Initialize $i \leftarrow 0$.
1. Attacker: $i \leftarrow i + 1$.  

We need many samples to correctly determine $D(h_0)$.  

By combining all $D(e_i)$ vectors we see a non-uniform probability distribution of individual distances that directly correlates to $D(h_0)$. 

We need many samples to correctly determine $D(h_0)$. 

9
A **reaction attack** against CCA secure QC-MDPC. [GJS16]

0. Attacker: Initialize $i \leftarrow 0$.
1. Attacker: $i \leftarrow i + 1$.
2. Attacker: Encrypts $c_i \leftarrow Gm + e_i$, where $e_i$ is a random vector.
A reaction attack against CCA secure QC-MDPC. [GJS16]

0. Attacker: Initialize $i \leftarrow 0$.
1. Attacker: $i \leftarrow i + 1$.
2. Attacker: Encrypts $c_i \leftarrow Gm + e_i$, where $e_i$ is a random vector.
3. Attacker: Sends $c_i$ to the victim.
A reaction attack against CCA secure QC-MDPC. [GJS16]

0. Attacker: Initialize $i \leftarrow 0$.
1. Attacker: $i \leftarrow i + 1$.
2. Attacker: Encrypts $c_i \leftarrow Gm + e_i$, where $e_i$ is a random vector.
3. Attacker: Sends $c_i$ to the victim.
4. Victim: Decrypts $c_i$ (using $\Psi_H$).
A reaction attack against CCA secure QC-MDPC. [GJS16]

0. Attacker: Initialize $i \leftarrow 0$.
1. Attacker: $i \leftarrow i + 1$.
2. Attacker: Encrypts $c_i \leftarrow Gm + e_i$, where $e_i$ is a random vector.
3. Attacker: Sends $c_i$ to the victim.
4. Victim: Decrypts $c_i$ (using $\Psi_H$).
5. Victim: Sends response back to attacker.
A reaction attack against CCA secure QC-MDPC. [GJS16]

0. Attacker: Initialize $i \leftarrow 0$.
1. Attacker: $i \leftarrow i + 1$.
2. Attacker: Encrypts $c_i \leftarrow Gm + e_i$, where $e_i$ is a random vector.
3. Attacker: Sends $c_i$ to the victim.
4. Victim: Decrypts $c_i$ (using $\Psi_H$).
5. Victim: Sends response back to attacker
6. Attacker: If decoding failure detected: Save $D(e_i)$.
A *reaction attack* against CCA secure QC-MDPC. [GJS16]

0. Attacker: Initialize $i \leftarrow 0$.
1. Attacker: $i \leftarrow i + 1$.
2. Attacker: Encrypts $c_i \leftarrow Gm + e_i$, where $e_i$ is a random vector.
3. Attacker: Sends $c_i$ to the victim.
4. Victim: Decrypts $c_i$ (using $\Psi_H$).
5. Victim: Sends response back to attacker.
6. Attacker: **If decoding failure detected**: Save $D(e_i)$.
7. Attacker: Repeat from step 1.
A reaction attack against CCA secure QC-MDPC. [GJS16]

0. Attacker: Initialize $i \leftarrow 0$.
1. Attacker: $i \leftarrow i + 1$.
2. Attacker: Encrypts $c_i \leftarrow Gm + e_i$, where $e_i$ is a random vector.
3. Attacker: Sends $c_i$ to the victim.
4. Victim: Decrypts $c_i$ (using $\Psi_H$).
5. Victim: Sends response back to attacker
6. Attacker: If decoding failure detected: Save $D(e_i)$.
7. Attacker: Repeat from step 1.

By combining all $D(e_i)$ vectors we see a non-uniform probability distribution of individual distances that directly correlates to $D(h_0)$. 
A **reaction attack** against CCA secure QC-MDPC. [GJS16]

0. Attacker: Initialize $i \leftarrow 0$.
1. Attacker: $i \leftarrow i + 1$.
2. Attacker: Encrypts $c_i \leftarrow Gm + e_i$, where $e_i$ is a random vector.
3. Attacker: Sends $c_i$ to the victim.
4. Victim: Decrypts $c_i$ (using $\Psi_H$).
5. Victim: Sends response back to attacker
6. Attacker: **If decoding failure detected**: Save $D(e_i)$.
7. Attacker: Repeat from step 1.

By combining all $D(e_i)$ vectors we see a **non-uniform** probability distribution of individual distances that directly **correlates** to $D(h_0)$. We need **many samples** to correctly determine $D(h_0)$. 
Background

Code-based Cryptography

Previous work

Attack Scenario

Contributions

The Chaining method

Generating $e_0$

Results

Amplification effect

Wrapping it up
New attack

An **adaptive reaction and/or side-channel** attack against CPA secure QC-MDPC:
An adaptive reaction and/or side-channel attack against CPA secure QC-MDPC:

0. Attacker: Initialize $i \leftarrow 0$,
1. Attacker: $i \leftarrow i + 1$.
2. Attacker: Encrypts $c_i \leftarrow Gm + e_i$, where $e_i$ is
3. Attacker: Sends $c_i$ to the victim.
4. Victim: Decrypts $c_i$ (using $\Psi_H$).
5. Victim: Sends response back to attacker
6. Attacker: If decoding failure detected:

7. Attacker: Repeat from step 1.
An adaptive reaction and/or side-channel attack against CPA secure QC-MDPC:

0. Attacker: Initialize $i \leftarrow 0, j \leftarrow 0$.
1. Attacker: $i \leftarrow i + 1$.
2. Attacker: Encrypts $c_i \leftarrow Gm + e_i$, where $e_i$ is derived from $e_j$.
3. Attacker: Sends $c_i$ to the victim.
4. Victim: Decrypts $c_i$ (using $\Psi_H$).
5. Victim: Sends response back to attacker.
6. Attacker: If decoding failure detected:

7. Attacker: Repeat from step 1.
An adaptive reaction and/or side-channel attack against CPA secure QC-MDPC:

0. Attacker: Initialize $i \leftarrow 0$, $j \leftarrow 0$, $e_0$ any non-decodable pattern.
1. Attacker: $i \leftarrow i + 1$.
2. Attacker: Encrypts $c_i \leftarrow Gm + e_i$, where $e_i$ is derived from $e_j$.
3. Attacker: Sends $c_i$ to the victim.
4. Victim: Decrypts $c_i$ (using $\Psi_H$).
5. Victim: Sends response back to attacker.
6. Attacker: If decoding failure detected:

7. Attacker: Repeat from step 1.
An adaptive reaction and/or side-channel attack against CPA secure QC-MDPC:

0. Attacker: Initialize $i \leftarrow 0$, $j \leftarrow 0$, $e_0$ any non-decodable pattern.
1. Attacker: $i \leftarrow i + 1$.
2. Attacker: Encrypts $c_i \leftarrow Gm + e_i$, where $e_i$ is derived from $e_j$.
3. Attacker: Sends $c_i$ to the victim.
4. Victim: Decrypts $c_i$ (using $\Psi_H$).
5. Victim: Sends response back to attacker
6. Attacker: If decoding failure detected:

7. Attacker: Repeat from step 1.
New attack

An adaptive reaction and/or side-channel attack against CPA secure QC-MDPC:

0. Attacker: Initialize $i \leftarrow 0$, $j \leftarrow 0$, $e_0$ any non-decodable pattern.
1. Attacker: $i \leftarrow i + 1$.
2. Attacker: Encrypts $c_i \leftarrow Gm + e_i$, where $e_i$ is derived from $e_j$.
3. Attacker: Sends $c_i$ to the victim.
4. Victim: Decrypts $c_i$ (using $\Psi_H$).
5. Victim: Sends response back to attacker
6. Attacker: If decoding failure detected: $e_j \leftarrow e_i$,

7. Attacker: Repeat from step 1.
An adaptive reaction and/or side-channel attack against CPA secure QC-MDPC:

0. Attacker: Initialize $i \leftarrow 0$, $j \leftarrow 0$, $e_0$ any non-decodable pattern.
1. Attacker: $i \leftarrow i + 1$.
2. Attacker: Encrypts $c_i \leftarrow Gm + e_i$, where $e_i$ is derived from $e_j$.
3. Attacker: Sends $c_i$ to the victim.
4. Victim: Decrypts $c_i$ (using $\Psi_H$).
5. Victim: Sends response back to attacker
6. Attacker: If decoding failure detected: $e_j \leftarrow e_i$, $j \leftarrow j + 1$,

7. Attacker: Repeat from step 1.
New attack

An adaptive reaction and/or side-channel attack against CPA secure QC-MDPC:

0. Attacker: Initialize $i ← 0$, $j ← 0$, $e_0$ any non-decodable pattern.
1. Attacker: $i ← i + 1$.
2. Attacker: Encrypts $c_i ← Gm + e_i$, where $e_i$ is derived from $e_j$.
3. Attacker: Sends $c_i$ to the victim.
4. Victim: Decrypts $c_i$ (using $\Psi_H$).
5. Victim: Sends response back to attacker
6. Attacker: If decoding failure detected: $e_j ← e_i$, $j ← j + 1$, $i ← 0$.

7. Attacker: Repeat from step 1.
New attack

An adaptive reaction and/or side-channel attack against CPA secure QC-MDPC:

0. Attacker: Initialize $i \leftarrow 0$, $j \leftarrow 0$, $e_0$ any non-decodable pattern.
1. Attacker: $i \leftarrow i + 1$.
2. Attacker: Encrypts $c_i \leftarrow Gm + e_i$, where $e_i$ is derived from $e_j$.
3. Attacker: Sends $c_i$ to the victim.
4. Victim: Decrypts $c_i$ (using $\Psi_H$).
5. Victim: Sends response back to attacker
6. Attacker: If decoding failure detected: $e_j \leftarrow e_i$, $j \leftarrow j + 1$, $i \leftarrow 0$. Save $D(e_i)$ regardless.
7. Attacker: Repeat from step 1.
An adaptive reaction and/or side-channel attack against CPA secure QC-MDPC:

0. Attacker: Initialize $i \leftarrow 0$, $j \leftarrow 0$, $e_0$ any non-decodable pattern.
1. Attacker: $i \leftarrow i + 1$.
2. Attacker: Encrypts $c_i \leftarrow Gm + e_i$, where $e_i$ is derived from $e_j$.
3. Attacker: Sends $c_i$ to the victim.
4. Victim: Decrypts $c_i$ (using $\Psi_H$).
5. Victim: Sends response back to attacker
6. Attacker: If decoding failure detected: $e_j \leftarrow e_i$, $j \leftarrow j + 1$, $i \leftarrow 0$.
   Save $D(e_i)$ regardless.
   if $\Psi_H$ not constant time: save time measurement of steps 3-5.
7. Attacker: Repeat from step 1.
An adaptive reaction and/or side-channel attack against CPA secure QC-MDPC:

0. Attacker: Initialize $i \leftarrow 0, j \leftarrow 0, e_0$ any non-decodable pattern.
1. Attacker: $i \leftarrow i + 1$.
2. Attacker: Encrypts $c_i \leftarrow Gm + e_i$, where $e_i$ is derived from $e_j$.
3. Attacker: Sends $c_i$ to the victim.
4. Victim: Decrypts $c_i$ (using $\Psi_H$).
5. Victim: Sends response back to attacker.
6. Attacker: If decoding failure detected: $e_j \leftarrow e_i, j \leftarrow j + 1, i \leftarrow 0$.
   Save $D(e_i)$ regardless.
   if $\Psi_H$ not constant time: save time measurement of steps 3-5.
7. Attacker: Repeat from step 1.

We call deriving $e_i$ from $e_j$ the chaining method, by which we significantly amplify the DFR.
Error Amplification is gained by generating a chain of related non-decodable error patterns:
The Chaining method

Error Amplification is gained by generating a chain of related non-decodable error patterns:

- From $e_0$ we can find another error pattern by randomly swapping a '1' and a '0' in the bit pattern (MUTATE).
Error Amplification is gained by generating a chain of related non-decodable error patterns:

- From $e_0$ we can find another error pattern by randomly swapping a ’1’ and a ’0’ in the bit pattern (MUTATE).
- Decoding success: $e^j_i \Rightarrow \Delta D^j_i \leftarrow D(e_j) - D(e^j_i)$
The Chaining method

Error Amplification is gained by generating a chain of related non-decodable error patterns:

- From $e_0$ we can find another error pattern by randomly swapping a '1' and a '0' in the bit pattern (MUTATE).
- Decoding success: $e_j^{ij} \Rightarrow \Delta D_j^{ij} \leftarrow D(e_j) - D(e_j^{ij})$
- Decoding failure: $e_{j+1} \Rightarrow \Delta D_j \leftarrow D(e_j) - D(e_{j+1})\}$ vectors!
Generating $e_0$

By using timing information we can distinguish the number of iterations required.

![Graph showing the number of nanoseconds required for decoding against the number of iterations required for decoding. The graph compares three decoders: Decoder $B$, Decoder $F$, and Decoder $Q$.]
By using timing information we can distinguish the number of iterations required.

We use the chaining method to find harder and harder patterns $e'_0$. 
Generating $e_0$

By using timing information we can distinguish the number of iterations required.

We use the chaining method to find harder and harder patterns $e_0'$.

- $e_0'$ is replaced each time a more difficult pattern is encountered!
By using timing information we can distinguish the number of iterations required.

We use the chaining method to find harder and harder patterns $e'_0$.

- $e'_0$ is replaced each time a more difficult pattern is encountered!
- Keep going until a decryption failure $e_0$ is found.
Background

Code-based Cryptography

Previous work

Attack Scenario

Contributions

The Chaining method

Generating $e_0$

Results

Amplification effect

Wrapping it up
We see that the vector

\[ \Delta D = \sum_{k=0}^{j} \Delta D_k \]

settle into multiplicity layers for large \( j \) (long chains).
We see that the vector \[ \Delta D = \sum_{k=0}^{j} \Delta D_k \]
settle into multiplicity layers for large \( j \) (long chains). Also using the successful decodings \( (\Delta D_k^j) \), inverted, improves the results.
Results

We see that the vector

$$\Delta D = \frac{\sum_{k=0}^{j} \Delta D_k}{j}$$

settle into multiplicity layers for large $j$ (long chains).

Also using the successfull decodings ($\Delta D_{ik}^j$, inverted), improves the results.

We can reconstruct the secret key using [GJS16]!
Amplification effect

Random samples

Chaining method

DFR indicated by horizontal lines.

Note the logarithmic scale on the y-axis!
Amplification effect

Random samples

Chaining method

DFR indicated by horizontal lines.

Note the logarithmic scale on the y-axis!
Background

Code-based Cryptography

Previous work

Attack Scenario

Contributions

The Chaining method

Generating $e_0$

Results

Amplification effect

Wrapping it up
Conclusions

• **Improvement** over the original (CPA-version) attack with a factor 20-30.
Conclusions

- **Improvement** over the original (CPA-version) attack with a factor 20-30.
- Low DFR’s as a protective measure might not be enough if we have **side-channels**.
Conclusions

- **Improvement** over the original (CPA-version) attack with a factor 20-30.
- Low DFR’s as a protective measure might not be enough if we have **side-channels**.
- Attacker selection of error patterns makes attacks possible and **efficient**.
Conclusions

• **Improvement** over the original (CPA-version) attack with a factor 20-30.

• Low DFR’s as a protective measure might not be enough if we have *side-channels*.

• Attacker selection of error patterns makes attacks possible and **efficient**.
  
    • Knowledge of a **single** non-decodable error pattern can be used as *leverage* for generating more.
Conclusions

- **Improvement** over the original (CPA-version) attack with a factor 20-30.
- Low DFR’s as a protective measure might not be enough if we have side-channels.
- Attacker selection of error patterns makes attacks possible and efficient.
  - Knowledge of a single non-decodable error pattern can be used as leverage for generating more.
  - IND-CCA secure schemes are not vulnerable to the chaining method.
Thank you!

(Questions?)
