Error Amplification in Code-based Cryptography

Alexander Nilsson[1,2] Thomas Johansson[1] Paul Stankovski Wagner[1]

August 27, 2019

[1]Dept. of Electrical and Information Technology, Lund University, Sweden

[2]Advenica AB, Malmö, Sweden

- One of the major branches of cryptographic post-quantum research.

- 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 McElice 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 McElice 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]:

QC-MDPC (1/5)

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 $n = n_0 r$.

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:

$$n_0 = 2, n = 9602, r = 4801, w = 90, t = 84$$

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 $H \in \mathbb{F}_2^{r \times n}$ is constructed as

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

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

The secret key $H \in \mathbb{F}_2^{r \times n}$ is constructed as

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

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

For $n_0 = 2$, we get

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

The secret key $H \in \mathbb{F}_2^{r \times n}$ is constructed as

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

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

For $n_0 = 2$, we get

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

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:

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

Public key $G \in \mathbb{F}_2^{(n-r)\times n}$ is constructed as follows:

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

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:

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

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, $\mathrm{wt}(e)$, less than t.

Public key $G \in \mathbb{F}_2^{(n-r)\times n}$ is constructed as follows:

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

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, $\mathrm{wt}(e)$, 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, Ψ_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, Ψ_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, Ψ_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, Ψ_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 (Ψ_H) are based on variants of the original Gallager's bitflipping algorithm.

- QC-MPDC was previosly shown vulnerable in [GJS16][1].

[1]Qian 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 [GJS16][1].
- Key recovery is possible with 250-300 M ciphertexts for 80-bit security parameters.

[1]Qian Guo, Thomas Johansson and Paul Stankovski. ”A Key Recovery Attack on MDPC with CCA security Using Decoding Errors”. In: ASIACRYPT 2016

Previous Work

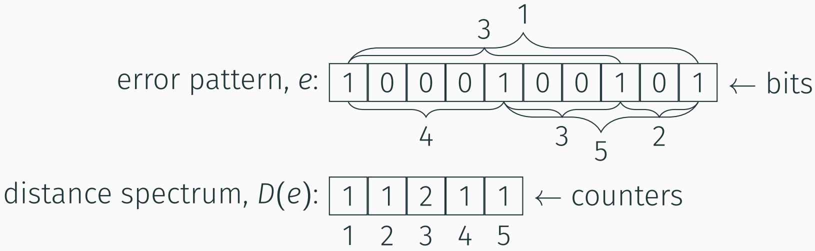
- QC-MPDC was previosly shown vulnerable in [GJS16][1].
- Key recovery is possible with 250-300 M ciphertexts for 80-bit security parameters.
- Attack against CCA secure QC-MDPC.

[1] Qian 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 [GJS16][1].
- 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.

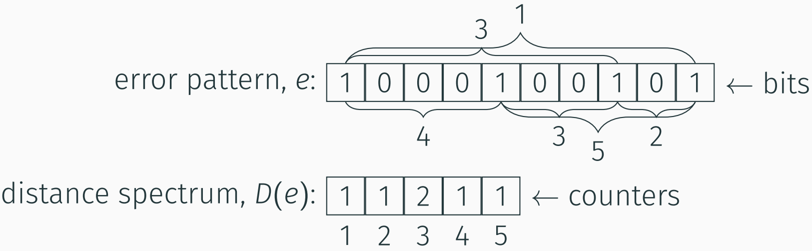
[1]Qian Guo, Thomas Johansson and Paul Stankovski. ”A Key Recovery Attack on MDPC with CCA security Using Decoding Errors”. In: ASIACRYPT 2016

Distance Spectrums



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

Distance Spectrums



Distance spectrum ($D(\ldots)$): *wrapping* distances between two non-zero bits. The number in each counter counts the occurence 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.

Previous attack

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

Previous attack

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: $i \leftarrow i + 1$.

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.

Previous attack

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 $\mathbf{\Psi}_H$).

Previous attack

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 Ψ_H).
5. Victim: Sends response back to attacker

Previous attack

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 $\mathbf{\Psi}_H$).
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 $\boldsymbol{\Psi}_H$).
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 $\mathbf{\Psi}_H$).
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 $\boldsymbol{\Psi}_H$).
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 $D(h_0)$.

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 $\mathbf{\Psi}_H$).
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
3. Attacker: Sends c_i to the victim.
4. Victim: Decrypts c_i (using Ψ_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
3. Attacker: Sends c_i to the victim.
4. Victim: Decrypts c_i (using Ψ_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 derrived from e_j.
3. Attacker: Sends c_i to the victim.
4. Victim: Decrypts c_i (using $\mathbf{\Psi}_H$).
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 derrived from e_j.
3. Attacker: Sends c_i to the victim.
4. Victim: Decrypts c_i (using $\mathbf{\Psi}_H$).
5. Victim: Sends response back to attacker
6. Attacker: If decoding failure detected: $e_j \leftarrow e_i$,
7. Attacker: Repeat from step 1.

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 derrived from e_j.
3. Attacker: Sends c_i to the victim.
4. Victim: Decrypts c_i (using Ψ_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 $i \leftarrow 0$, $j \leftarrow 0$, 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 derrived from e_j.
3. Attacker: Sends c_i to the victim.
4. Victim: Decrypts c_i (using Ψ_H).
5. Victim: Sends response back to attacker
6. Attacker: If decoding failure detected: $e_j \leftarrow e_i$, $j \leftarrow j + 1$, $i \leftarrow 0$.
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 derrived from e_j.
3. Attacker: Sends c_i to the victim.
4. Victim: Decrypts c_i (using Ψ_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.

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 derrived from e_j.
3. Attacker: Sends c_i to the victim.
4. Victim: Decrypts c_i (using Ψ_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 Ψ_H not constant time: save time measurment of steps 3-5.
7. Attacker: Repeat from step 1.

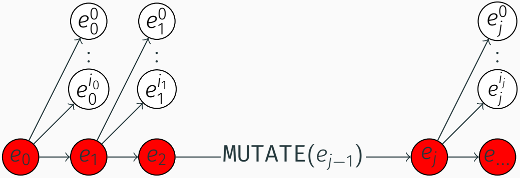
New attack

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

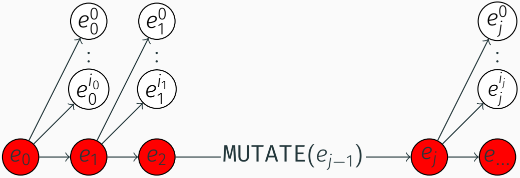
0. Attacker: Initialize $i \leftarrow 0$, $j \leftarrow 0$, e_0 any non-decodable pattern.
1. Attacker: $i \leftarrow i + 1$.
2. Attacker: Encrypts $c_i \leftarrow Gm + e_i$, where e_i is derrived from e_j.
3. Attacker: Sends c_i to the victim.
4. Victim: Decrypts c_i (using $\boldsymbol{\Psi}_H$).
5. 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 $\boldsymbol{\Psi}_H$ not constant time: save time measurment 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.

The Chaining method



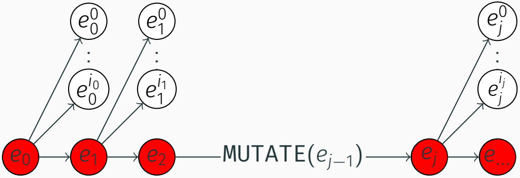
Error Amplification is gained by generating a chain of related non-decodable error patterns:



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

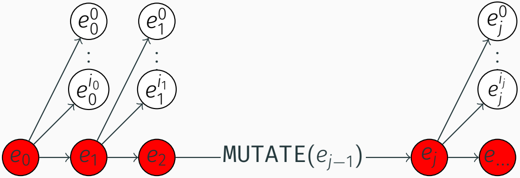
- From e_0 we can find another error pattern by randomly swapping a '1' and a '0' in the bit pattern (MUTATE).

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: $e_j^{i_j} \Rightarrow \Delta D_j^{i_j} \leftarrow D(e_j) - D(e_j^{i_j})$



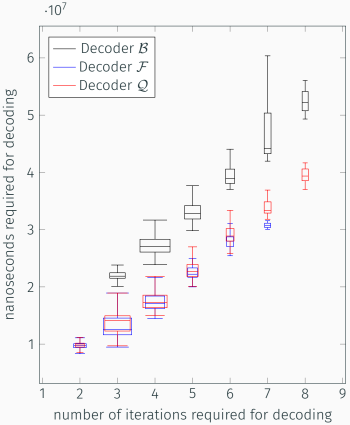
Error Amplification is gained by generating a chain of related non-decodable error patterns:

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

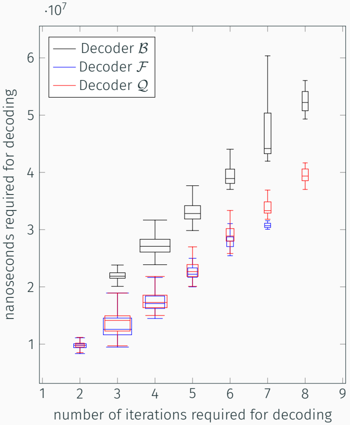
} vectors!

Generating e_0

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

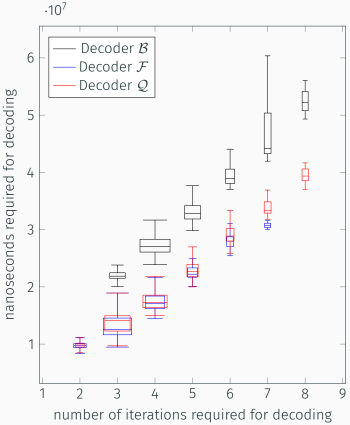


By using timing information we can distinguish the number of iterations required.



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

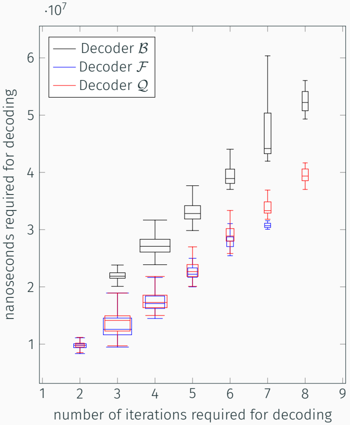
By using timing information we can distinguish the number of iterations required.



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.



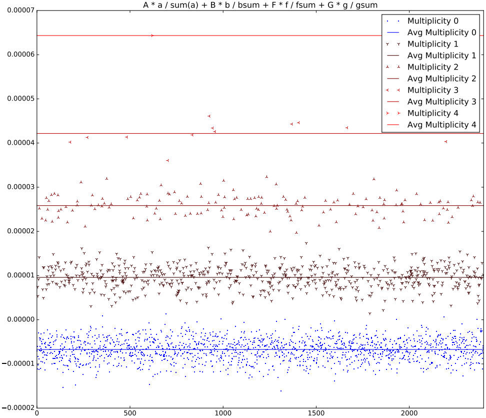
We use the chaining method to find harder and harder patterns e_0'.

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

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).



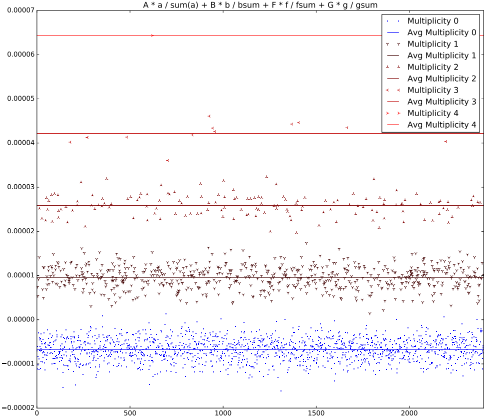
Results

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 ($\Delta D_k^{i_k}$), inverted, improves the results.



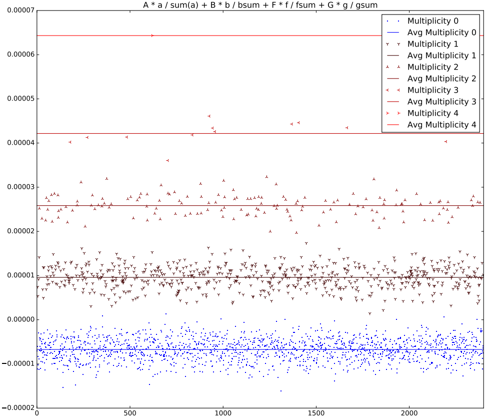
Results

We see that the vector

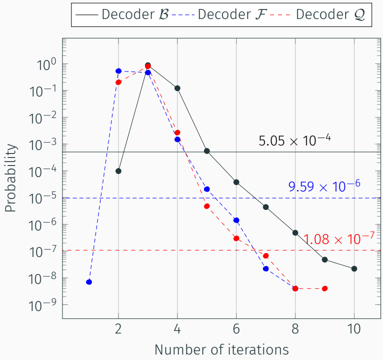
$$\Delta D = \frac{\sum_{k=0}^{j} \Delta D_k}{j}$$

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

Also using the successfull decodings ($\Delta D_k^{i_k}$), inverted, improves the results.



We can reconstruct the secret key using [GJS16]!



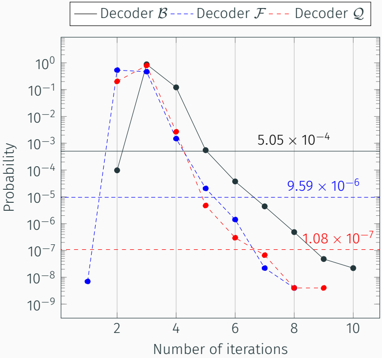
Random samples

Chaining method

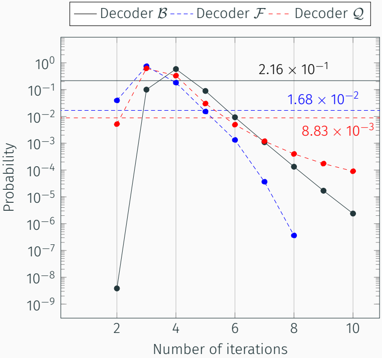
DFR indicated by horizontal lines.

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!

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?)

[GJS16] Qian Guo, Thomas Johansson, and Paul Stankovski. "A Key Recovery Attack on MDPC with CCA Security Using Decoding Errors". In: *ASIACRYPT 2016, Part I*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. LNCS. Springer, Heidelberg, Dec. 2016, pp. 789–815. DOI: 10.1007/978-3-662-53887-6_29.

[Mis+12] Rafael Misoczki et al. *MDPC-McEliece: New McEliece Variants from Moderate Density Parity-Check Codes*. Cryptology ePrint Archive, Report 2012/409. http://eprint.iacr.org/2012/409. 2012.

[NJW18] Alexander Nilsson, Thomas Johansson, and Paul Stankovski Wagner. "Error Amplification in Code-based Cryptography". In: *IACR TCHES* 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.