Differential Power Analysis Attack on the Secure Bit Permutation in the McEliece Cryptosystem

Martin Petrvalsky, Tania Richmond, Milos Drutarovsky, Pierre-Louis Cayrel, Viktor Fischer

26th Conference Radioelektronika 2016, April 19-20, Košice, Slovak Republic

Differential Power Analysis Attack on the Secure Bit Permutation in the McEliece Cryptosystem

Martin Petrvalsky[1], *Tania Richmond*[2,3], *Milos Drutarovsky*[1], *Pierre-Louis Cayrel*[2], *Viktor Fischer*[2]

[1] Dept. of Electronics and Multimedia Communications
Technical University of Kosice, Kosice, Slovakia
{martin.petrvalsky,milos.drutarovsky}@tuke.sk
[2] Hubert Curien Laboratory, Jean Monnet University
18, Rue du Prof. B. Lauras, 18, 42000 Saint-Etienne, France
{tania.richmond,pierre.louis.cayrel,fischer}@univ-st-etienne.fr
[3] IMATH Laboratory, University of Toulon
Avenue de l'Université, BP 20132, 83957 La Garde Cedex, France
tania.richmond@univ-tln.fr

ABSTRACT

The segment of post-quantum cryptography rises its importance with increasing improvements in the quantum computing. Cryptographic post-quantum algorithms have been proposed since 1970s. However, side-channel attack vulnerabilities of these algorithms are still in focus of the recent research. In this paper, we present a differential power analysis attack on the McEliece public-key cryptosystem. We demonstrate that a part of a private key, permutation matrix, can be recovered using the power analysis. We attack a software implementation of a secure bit permutation that was proposed by Strenzke et al. at PQCrypto 2008. The cryptosystem is implemented on a 32-bit ARM based microcontroller. We provide details of the attack and results using power consumption measurements of the device. In addition, we outline a novel countermeasure against the introduced attack. The countermeasure uses properties of the linear codes and does not require large amount of random bits which can be profitable for low-cost embedded devices.

Index Terms— Differential power analysis, Goppa codes, McEliece cryptosystem, secure bit permutation, side-channel attack

1. INTRODUCTION

Code-based cryptography offers interesting properties - fast computation, favorable theoretical complexity and robustness of the underlying non-deterministic polynomial-time hard problem (NP-hard problem) which can not be broken by quantum algorithms in polynomial-time [1].

This work was performed in the framework of the COST Action IC1204 (Trustworthy Manufacturing and Utilization of Secure Devices). It was supported by the Slovak Research and Development Agency, project number APVV-0586-11 and in part by NATO's Public Diplomacy Division in the framework of "Science for Peace", SPS Project 984520.

The first public-key cryptosystem (PKC) based on error-correcting codes was proposed by McEliece in 1978 [2]. In his paper, McEliece proposed to use the family of Goppa codes. Permuted Goppa codes can provide a couple of advantages - they look like random codes and they can be decoded efficiently. A Goppa decoder can be used as a trapdoor in code-based cryptography. In this context, we are interested in implementations of the McEliece PKC and associated leakages in order to recover the private key.

Side-channel attacks (SCAs) exploit a physical phenomenon of an implementation, e.g. running time in software or power consumption in hardware. The first SCA against the McEliece PKC was proposed in 2008 [3] and more papers have followed during the last years. Most of those attacks target the Patterson's algorithm [4] and they are based on timing attack [3], [5, 6, 7, 8]. However, we are interested only in power consumption analysis attacks in this paper.

Attacks against the McEliece PKC with Goppa codes using a simple power analysis (SPA) have been proposed in [9, 10]. However, to the best of our knowledge, this is the first differential power analysis (DPA) attack on the McEliece PKC using Goppa codes (the DPA in [11] is against the McEliece PKC using QC-MDPC codes).

There are four profiles used for the first steps of the McEliece decryption [9, Section 3]. Profiles III and IV do not require permutation algorithm and thus they are more secure. However, the first step in profiles I and II permutes input ciphertexts which is convenient for some applications, e.g. for embedded and constraint devices. In this work, we are focused on the profiles I and II where the permutation algorithm must be implemented in a secure way.

The paper is organized as follows. We start by presenting theoretical background on Goppa codes and the McEliece PKC in Section 2. In Section 3, we describe the DPA attack against the ciphertext permutation given in [3, Algorithm

3]. Furthermore, we provide an outline of a countermeasure against the proposed DPA attack in Section 4. Finally, we conclude the paper in Section 5.

2. THEORETICAL BACKGROUND

2.1. Goppa Codes

Goppa proposed a large class of linear error-correcting codes in 1970 [12, 13]. However, our interest is focused exclusively on irreducible binary Goppa codes that are commonly used for encryption. For the sake of simplicity, we will call them Goppa codes.

Definition 1 *Let m and t be positive integers. Let $\mathcal{L} = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ represent a subset of $\mathbb{F}_{2^m}$ such that the α_i are pairwise distinct elements, so $n \leqslant 2^m$. Given a monic (irreducible) polynomial $g(x) \in \mathbb{F}_{2^m}[x]$ such that $\deg(g) = t$ and $g(\alpha_i) \neq 0$ for $i = 1, \ldots, n$, the (irreducible) Goppa code $\Gamma(\mathcal{L}, g)$ is defined as:*

$$\Gamma(\mathcal{L}, g) = \left\{ C = (C_1, C_2, \ldots, C_n) \in \mathbb{F}_2^n \;\middle|\; \sum_{i=1}^{n} \frac{C_i}{x \oplus \alpha_i} \equiv 0 \mod g(x) \right\}. \quad (1)$$

We call the syndrome polynomial, a polynomial associated to $C \in \mathbb{F}_2^n$ given by:

$$\mathcal{S}_C(x) = \sum_{i=1}^{n} \frac{C_i}{x \oplus \alpha_i}. \quad (2)$$

To decode a binary Goppa codeword containing errors, one commonly adopted solution is to use the so-called Patterson's algorithm [4]. The input of a decoding algorithm consists of a product of a Goppa code parity-check matrix denoted $\mathcal{H}$ and the codeword C with at most t errors, i.e. $S = C \cdot \mathcal{H}^T$. The result of this operation is called the syndrome and it can be viewed as a polynomial as $S_C(x) = [x^{t-1}, \ldots, x, 1] \cdot S$, i.e. (2).

2.2. The McEliece Cryptosystem

McEliece proposed the first public-key code-based encryption scheme in 1978 [2]. The McEliece PKC, using Goppa codes as in the original paper, is performed using the three following algorithms - key generation, plaintext encryption and ciphertext decryption.

2.2.1. Key Generation.

The key generation consists of the determination of the Goppa code according to Definition 1 given in Section 2.1. Since Goppa codes are linear, they can be generated by a so-called $k \times n$ generator matrix denoted $\mathcal{G}$. We randomly choose a non-singular $k \times k$ matrix $\mathcal{S}$ and a $n \times n$ permutation matrix $\mathcal{P}$. Then we compute the public $k \times n$ generator matrix given by $\tilde{\mathcal{G}} = \mathcal{S} \cdot \mathcal{G} \cdot \mathcal{P}$. The key generation provides the secret key $\mathsf{sk} = (\Gamma(\mathcal{L}, g), \mathcal{S}, \mathcal{P})$ and the public key $\mathsf{pk} = (m, t, \tilde{\mathcal{G}})$. Notice that k and n are also public because $\tilde{\mathcal{G}}$ is a $k \times n$ matrix.

2.2.2. Plaintext Encryption.

During the plaintext encryption, the message M is encrypted using the public generator matrix. This operation can be expressed by $C = M \cdot \tilde{\mathcal{G}}$. Then an error vector E of length n and weight t is randomly selected and added to the codeword C, giving the ciphertext $\tilde{C} = C \oplus E$.

2.2.3. Ciphertext Decryption.

At first, the product $\tilde{C}_p = \tilde{C} \cdot \mathcal{P}^{-1}$ is computed during the decryption of the ciphertext $\tilde{C}$. The attack described in Section 3 targets this phase of the ciphertext decryption. This step is leading to a codeword containing an error which can be mathematically described as $\underbrace{M \cdot \mathcal{S} \cdot \mathcal{G}}_{\text{codeword}} \oplus \underbrace{E \cdot \mathcal{P}^{-1}}_{\substack{\text{error vector} \\ \text{of weight } t}}$.

Next, a decoding algorithm (the Patterson's algorithm in our case) must be applied on the obtained secret code. Let $\mathcal{G}_r^{-1}$ be the $\mathcal{G}$ right-side inverse, i.e. $\mathcal{G} \cdot \mathcal{G}_r^{-1} = \mathcal{I}_k$, where $\mathcal{I}_k$ is the $k \times k$ identity matrix. Thereafter the obtained codeword $M \cdot \mathcal{S} \cdot \mathcal{G}$ is multiplied by $\mathcal{G}_r^{-1}$ on the right-side, in order to find $\tilde{M} = M \cdot \mathcal{S}$. Finally we compute $M = \tilde{M} \cdot \mathcal{S}^{-1}$ to recover the plaintext.

3. DPA ATTACK ON THE SECURE BIT PERMUTATION

3.1. Measurement Setup for the DPA Attack

We attack a software implementation of the McEliece PKC decryption algorithm running on an STM32F103 microcontroller (MCU) [14]. The MCU features a 32-bit ARM Cortex-M3 (clocked at 72 MHz). We acquire instantaneous power consumption traces (traces) by measuring voltage drop on a 1Ω resistor placed in series between a grounding pin of the MCU and the ground. We assume a constant voltage on the MCU power supply thus the measured voltage drop is proportional to power consumption. We do not take uncertainties, precise calibrations and small deviations of the power supply into an account since it has insignificant or no effect while using DPA and CPA attacks.

Traces are acquired using the Agilent Technologies oscilloscope DSO9404A [15] (using 2 of 4 analog channels @ 1 MΩ). A workflow of the measurement process is depicted in Fig. 1. All traces needed for attacking protected device are acquired at sample rate 250×10^6 samples per second (250 MS/s), vertical resolution 12-bit averaged, filtered DC signal component and with a range +/- 30 mV. Two 500 MHz passive probes were connected directly to *SubMiniature version A* (SMA) connectors available on the testing board with the MCU.

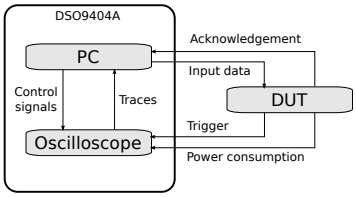

Fig. 1. Workflow of the measurement process. The acquisition of traces is controlled by the PC embedded in the oscilloscope.

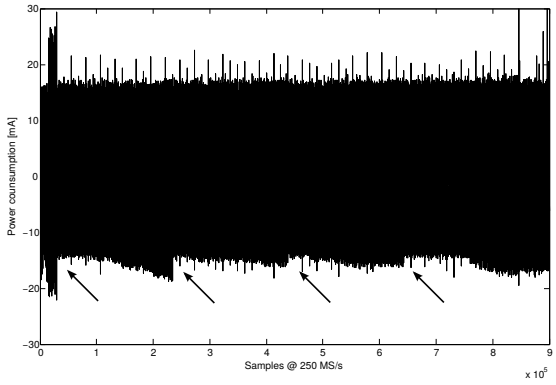

Fig. 2. An example of the power consumption trace.

Data acquisition is controlled by the software running on the oscilloscope's internal PC. The software sets up the oscilloscope, sends the ciphertext to the MCU – design under test (DUT) via UART and waits for the acknowledgment. The DUT rises a trigger and starts the ciphertext decryption. The oscilloscope measures power consumption during the first step of the decryption. Once the acquisition is finished, the PC stores the measured trace to the hard disk. The measurement process is repeated depending on the desired number of traces. An average speed for a measurement of one trace with 9×10^5 samples is 1.5 s. In traces (Fig. 2), we can distinguish four patterns of rising amplitude (especially in negative half waves). These patterns are caused by implementation which stores 64-bit ciphertexts in four 16-bit words. It allows us to distinguish and attack (or protect) each word separately. Later, we can use this fact to generalize our statement for any length of the ciphertexts.

3.2. Attacked Secure Bit Permutation Algorithm

We attack the secure bit permutation algorithm proposed by Strenzke et al. [3, Algorithm 3] (recalled in this paper in Algorithm 1). We found a flaw in the algorithm which leaks information about the private permutation matrix in the McEliece PKC. Using a DPA attack, we are able to reconstruct the whole matrix. We chose to attack the permutation algorithm with $n = 64$. We store values $n = 64$ as four 16-bit words as we can see in Fig. 2 (LSW, two middle words and MSW). Since we cover all types of words (LSW, middle words and MSW), the proposed attack can be enhanced to any value of $n > 3$ in order to attack more practical implementations ($n = 1024, n = 2048, \ldots$) assuming straightforward implementation.

Algorithm 1. Secure bit permutation $\tilde{C}_p = \tilde{C} \cdot \mathcal{P}^{-1}$ (from [3])

Require: Private $n \times n$ permutation matrix $\mathcal{P}$ represented by lookup-table $t^{\mathcal{P}}$ and ciphertext vector $\tilde{C}$
Ensure: Permuted ciphertext $\tilde{C}_p$ (of length n)
1: **for** $i = 0$ **to** $n - 1$ **do**
2: $\quad j = t_i^{\mathcal{P}}$
3: $\quad \tilde{C}_{p_i} = 0$
4: $\quad$ **for** $h = 0$ **to** $n - 1$ **do**
5: $\quad\quad l = \tilde{C}_{p_i}$
6: $\quad\quad \mu = \tilde{C}_h$
7: $\quad\quad s = j \oplus h$
8: $\quad\quad s \mid= s \gg 1$
9: $\quad\quad s \mid= s \gg 2$
10: $\quad\quad s \mid= s \gg 4$
11: $\quad\quad s \mid= s \gg 8$
12: $\quad\quad s \mid= s \gg 16$
13: $\quad\quad s \,\&= 1$
14: $\quad\quad s = \sim (s - 1)$
15: $\quad\quad \tilde{C}_{p_i} = (s \,\&\, l) \mid ((\sim s) \,\&\, \mu)$
16: $\quad$ **end for**
17: **end for**
18: **return** $\tilde{C}_p$

With notations used in Algorithm 1, the vulnerable operation is 15. Indeed, for $0 \leqslant h < j$, $\tilde{C}_{p_i}$ will be zero and from $h = j$ to $h = n - 1$, $\tilde{C}_{p_i}$ will be $\mu = \tilde{C}_h$. It means that the i-th bit of $\tilde{C}_p$ will become the h-th bit of $\tilde{C}$ only when $j = h$ (i.e. when $s = 0$) and otherwise it will not change (i.e. when $s \neq 0$). These properties allow us to deploy the DPA attack (Fig. 3). During the vulnerable operation (Step 15), we can observe the instant when μ is assigned to $\tilde{C}_{p_i}$ ($j = h$). With the DPA we are able to detect time of this assignment and thus reveal one row of the permutation matrix.

3.3. Principle and Results of the DPA Attack

We deploy the DPA attack based on the Pearson's correlation coefficient [16] with known input ciphertext. The workflow of the DPA is depicted in Fig. 3. We apply a Hamming weight of individual bits leakage model. ($H_i \in \{0, 1\}$). We use (3) for correlation analyses:

$$r_{H,X}(\eta) = \frac{\sum_{i=1}^{N}[(X_i(\eta) - \bar{X}(\eta))(H_i - \bar{H})]}{\sqrt{\sum_{i=1}^{N}[X_i(\eta) - \bar{X}(\eta)]^2 \sum_{i=1}^{N}(H_i - \bar{H})^2}} \quad (3)$$

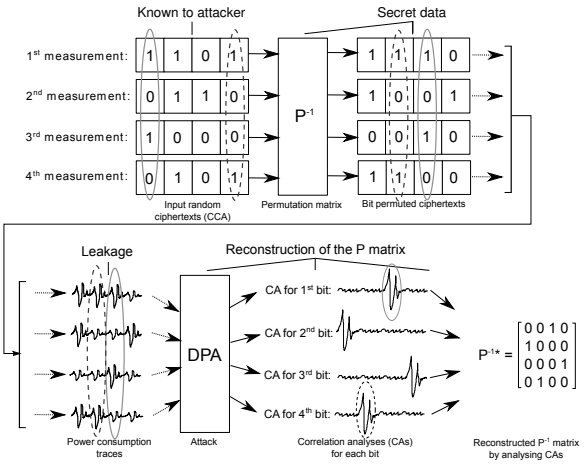

Fig. 3. Main steps of the DPA attack on the secure bit permutation.

where $r_{H,X}(\eta)$ is the Pearson's correlation coefficient for η-th sample (measured during execution of the cryptographic algorithm), N is a number of measured traces, $X_i(\eta)$ is a value of η-th sample measured during i-th measurement (i-th trace), $\bar{X}(\eta)$ is a mean value of corresponding η-th samples (from all traces), H_i is a hypothesis of power consumption for one bit of input data corresponding with i-th measurement (i-th trace) and $\bar{H}$ is a mean value of all hypotheses H_i.

We need to perform the correlation analysis n times for each input bit (in our case 64 times). We obtain positions of permuted bits by searching for correlation peaks during analyses as depicted in Fig. 4. We can clearly distinguish the moment (marked with arrows) when a bit from input ciphertexts is handled for the first time during the permutation algorithm. Since we know the position of bits in input ciphertexts and the position of the same bits in permuted ciphertexts (from correlation analyses), we can reconstruct the permutation matrix $\mathcal{P}^{-1}$. The attack with 500 traces takes several minutes on a 2.4 GHz Core i7 CPU.

4. COUNTERMEASURE OUTLINE AGAINST THE DPA ATTACK

One of the most common countermeasure against DPA attacks is masking [17]. Two main disadvantages of the masking technique are requirement of random bits and increased computational complexity. In our novel method of countermeasure (Algorithm 2), we are able to use the masking with decreased effects of mentioned disadvantages. We benefit from properties of the McEliece PKC decryption:

- syndrome computation is scheduled right after bit permutation,
- every Goppa codeword has a zero syndrome,

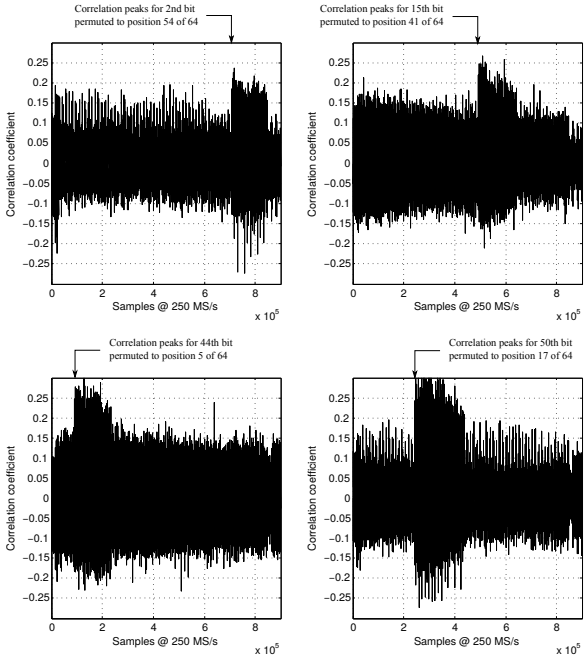

Fig. 4. Four selected correlation analyses using 500 traces.

- rows in generator matrix $\mathcal{G}$ are codewords.

Since the syndrome computation is a linear operation and the syndrome equals zero for all codewords, the main idea for our countermeasure is to add a codeword to the permuted ciphertext and compute the syndrome with this new word $\tilde{C}'_p = \tilde{C}_p \oplus B$, where $B \in \Gamma(\mathcal{L}, g)$ (for example one row in $\mathcal{G}$). The syndrome is the same since $S = \tilde{C}'_p \cdot \mathcal{H}^T = (\tilde{C}_p \oplus B) \cdot \mathcal{H}^T = \tilde{C}_p \cdot \mathcal{H}^T \oplus \underbrace{B \cdot \mathcal{H}^T}_{=0} = \tilde{C}_p \cdot \mathcal{H}^T$.

Hence we need to add the mask to the input ciphertext $\tilde{C}$ *before* the permutation algorithm, we add a permuted codeword ${}_pB$ such that ${}_pB \cdot \mathcal{P}^{-1} = B$. After the permutation, we get $\tilde{C}'_p = (\tilde{C} \oplus {}_pB) \cdot \mathcal{P}^{-1} = (\tilde{C} \cdot \mathcal{P}^{-1}) \oplus ({}_pB \cdot \mathcal{P}^{-1}) = \tilde{C}_p \oplus B$ which is input for the syndrome computation in the previous paragraph.

In regular masking method, we need to generate random masks with width at least equal to the width of masked data. After, we perform calculations with mask and masked data. In the end, we merge mask and masked data into resulting data of the desired operation. In our case, we have to obtain a permuted Goppa codeword ${}_pB$. There are several possibilities to do so. It can be done either by choosing a random Goppa codeword or we can use a row in the generator matrix $\mathcal{G}$ (and linear combinations) and modify them using the permutation matrix. Another possibility is to use linear combinations of rows from public matrix $\tilde{\mathcal{G}}$ (this option is potentially vulnerable to an attack since an adversary knows the $\tilde{\mathcal{G}}$ matrix). For constraint devices we can pre-compute sufficient amount of

modified codewords ${}_pB$ and use their linear combinations for each decryption. After we add ${}_pB$ to the ciphertext, we get correct result from the McEliece PKC decryption *with no additional steps*.

Compared to masking scheme, this method is more effective in terms of computational time and resources (memory space, random bits). Using these properties, we avoid storing multiple values (masks and masked data) and we compute operations of permutation and parity-check matrix multiplication only once. Relations between intermediate values and input data are broken and thus the overall first-order DPA leakage is reduced. Indeed, a higher order DPA attack [18] is still possible if an adversary attacks multiple vulnerable data. Mutual information of ${}_pB$ and 15 in Algorithm 2 can lead to a successful second order DPA attack. On the other hand, higher order DPA attacks are more complex than the first order DPA.

In the best of our knowledge, this is the first time that these properties are used as a countermeasure against an SCA in the McEliece PKC. The proposed DPA attack resistant bit permutation is sketched in Algorithm 2 (with underlined changes compared to Algorithm 1).

Algorithm 2. Secure bit permutation $\tilde{C}'_p = (\tilde{C} \oplus {}_pB) \cdot \mathcal{P}^{-1}$ with assumed DPA resistance

Require: Private permutation matrix $\mathcal{P}$ represented by lookup-table $t^{\mathcal{P}}$, ciphertext vector $\tilde{C}$ and the private code $\Gamma(\mathcal{L}, g)$
Ensure: Permuted ciphertext with added Goppa codeword $(\tilde{C}'_p = \tilde{C}_p \oplus B)$

1: Choose ${}_pB$
2: $\tilde{C}' = \tilde{C} \oplus {}_pB$
3: **for** $i = 0$ **to** $n-1$ **do**
4: $\quad j = t_i^{\mathcal{P}}$
5: $\quad \tilde{C}'_{p_i} = 0$
6: $\quad$ **for** $h = 0$ **to** $n-1$ **do**
7: $\quad\quad l = \tilde{C}'_{p_i}$
8: $\quad\quad \mu = \tilde{C}'_h$
9: $\quad\quad s = j \oplus h$
10: $\quad\quad s \mid= s \gg 1$
11: $\quad\quad s \mid= s \gg 2$
12: $\quad\quad s \mid= s \gg 4$
13: $\quad\quad s \mid= s \gg 8$
14: $\quad\quad s \mid= s \gg 16$
15: $\quad\quad s \;\&= 1$
16: $\quad\quad s = \sim (s-1)$
17: $\quad\quad \tilde{C}'_{p_i} = (s \;\&\; l) \mid ((\sim s) \;\&\; \mu)$
18: $\quad$ **end for**
19: **end for**
20: **return** $\tilde{C}'_p$

We add permuted Goppa codes ${}_pB$ to input ciphertext in Algorithm 2. From the point of view of an adversary, we summate input ciphertexts with random values. In Step 17, the adversary can not survey which of the variables l and μ is assigned to $\tilde{C}'_{p_i}$ as it is in Step 15 of Algorithm 1. The reason is that the hypotheses which the adversary would create are completely distorted by the addition of the mask ${}_pB$ as shown in Fig 5.

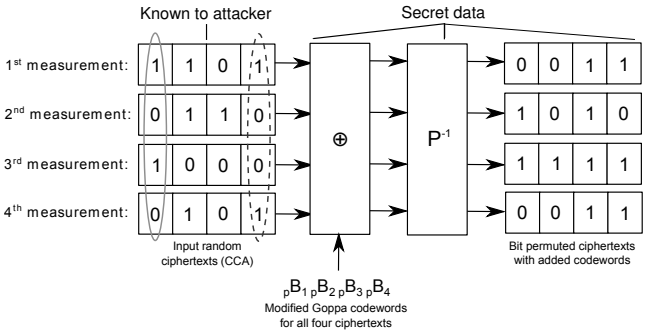

Fig. 5. Secure bit permutation with the proposed countermeasure. Modified Goppa codes are added to ciphertexts as masks before the permutation. During and after the permutation, an adversary is unable to find patterns (gray and dashed elipses) in permuted ciphertexts with added Goppa codewords.

5. CONCLUSION

In this paper, we successfully deployed the DPA attack targeting the McEliece PKC decryption. We attack the bit permutation including the countermeasure proposed by Strenzke et al. The permutation was implemented and attacked on the ARM Cortex-M3 based MCU. We showed that without a countermeasure, we were able to recover the whole 64×64 permutation matrix as a toy example. The DPA attack can be extended to usual McEliece PKC parameters, e.g. $n = 2048$.

Secondly, we proposed an outline of a countermeasure against this DPA attack. We developed a method which is based on a masking technique. We add Goppa codewords (random or rows of $\mathcal{G}$) to ciphertexts during the permutation algorithm as masks. By using properties of the McEliece PKC, we reduced the number of random bits that are needed for the masking. After the codeword addition, there is no change in the decryption algorithm. With these properties, we decreased a computational complexity as well (compared to the regular masking technique). In theory, the masking countermeasure significantly reduce the overall leakage - this fact has yet to be evaluated.

In the future, we will expand the DPA attack and the countermeasure for usual McEliece PKC parameters. Furthermore, we will implement the outlined countermeasure and we well test its robustness. Next, we will apply the DPA attack on other implementations of the McEliece PKC where we expect DPA vulnerabilities. We will also focus on other parts of the McEliece decryption algorithm such as syndrome computation.

6. REFERENCES

[1] Elwyn R. Berlekamp, Robert James McEliece, and Henk C. A. van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Transactions on Information Theory*, vol. 24, no. 3, pp. 384–386, May 1978.

[2] Robert James McEliece, "A public-key cryptosystem based on algebraic coding theory," Tech. Rep. 44, California Inst. Technol., Pasadena, CA, January 1978.

[3] Falko Strenzke, Erik Tews, H. Gregor Molter, Raphael Overbeck, and Abdulhadi Shoufan, "Side channels in the McEliece PKC," in *The Second International Workshop on Post-Quantum Cryptography (PQCrypto 2008)*, Johannes Buchmann and Jintai Ding, Eds., vol. 5299 of *Lecture Notes in Computer Science*, pp. 216–229. Springer, Berlin Heidelberg, October 2008.

[4] Nicholas J. Patterson, "The algebraic decoding of Goppa codes," *IEEE Transactions on Information Theory*, vol. 21, no. 2, pp. 203–207, March 1975.

[5] Abdulhadi Shoufan, Falko Strenzke, H. Gregor Molter, and Marc Stöttinger, "A timing attack against Patterson algorithm in the McEliece PKC," in *Proceedings of the 12th International Conference on Information, Security and Cryptology (ICISC 2009)*, Donghoon Lee and Seokhie Hong, Eds., vol. 5984 of *Lecture Notes in Computer Science*, pp. 161–175. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[6] Falko Strenzke, "A timing attack against the secret permutation in the McEliece PKC," in *Proceedings of the Third international conference on Post-Quantum Cryptography (PQCrypto 2010)*, Nicolas Sendrier, Ed., vol. 6061 of *Lecture Notes in Computer Science*, pp. 95–107. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[7] Roberto M. Avanzi, Simon Hoerder, Dan Page, and Michael Tunstall, "Side-channel attacks on the McEliece and Niederreiter public-key cryptosystems," *Journal of Cryptographic Engineering*, vol. 1, no. 4, pp. 271–281, November 2011.

[8] Falko Strenzke, "Timing attacks against the syndrome inversion in code-based cryptosystems," in *The 5th International Workshop on Post-Quantum Cryptography (PQCrypto 2013)*, Philippe Gaborit, Ed., vol. 7932 of *Lecture Notes in Computer Science*, pp. 217–230. Springer, Berlin Heidelberg, 2013.

[9] Stefan Heyse, Amir Moradi, and Christof Paar, "Practical power analysis attacks on software implementations of McEliece," in *Proceedings of the Third international conference on Post-Quantum Cryptography (PQCrypto 2010)*, Nicolas Sendrier, Ed., vol. 6061 of *Lecture Notes in Computer Science*, pp. 108–125. Springer, Berlin Heidelberg, 2010.

[10] H. Gregor Molter, Marc Stöttinger, Abdulhadi Shoufan, and Falko Strenzke, "A simple power analysis attack on a McEliece cryptoprocessor," *Journal of Cryptographic Engineering*, vol. 1, no. 1, pp. 29–36, April 2011.

[11] Cong Chen, Thomas Eisenbarth, Ingo von Maurich, and Rainer Steinwandt, "Differential power analysis of a McEliece cryptosystem," Cryptology ePrint Archive, Report 2014/534, 2014.

[12] Valerii Denisovich Goppa, "A new class of linear error-correcting codes," *Problemy Peredachi Informatsii*, vol. 6, no. 3, pp. 24–30, September 1970.

[13] Elwyn R. Berlekamp, "Goppa codes," *IEEE Transactions on Information Theory*, vol. 19, no. 5, pp. 590–592, September 1973.

[14] ST Microelectronics, "STM32 product information, software and datasheets," `http://www.st.com/web/en/catalog/mmc/FM141/SC1169`, [online] Cited 12/12/2015.

[15] Agilent Technologies, "DSO9404A datasheet and product information," `http://www.keysight.com/en/pd-1632456-pn-DSO9404A/oscilloscope-4-ghz-4-analog-channels?&cc=SK&lc=eng`, [online] Cited 12/12/2015.

[16] Eric Brier, Christophe Clavier, and Francis Olivier, "Correlation power analysis with a leakage model," in *CHES*, 2004, pp. 16–29.

[17] Xiaoan Zhou, Juan Peng, and Liping Guo, "An improved AES masking method smartcard implementation for resisting DPA attacks," *International Journal of Computer Science Issues (IJCSI)*, vol. 10, no. 01, pp. 118–122, Mar. 2013.

[18] Matthieu Rivain, Emmanuel Prouff, and Julien Doget, "Higher-order masking and shuffling for software implementations of block ciphers.," in *CHES*, Christophe Clavier and Kris Gaj, Eds. 2009, vol. 5747 of *Lecture Notes in Computer Science*, pp. 171–188, Springer.