Countermeasure against the SPA attack on an embedded McEliece cryptosystem

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

Countermeasure against the SPA Attack on an Embedded McEliece Cryptosystem

Martin Petrvalsky*, Tania Richmond†, Milos Drutarovsky*, Pierre-Louis Cayrel† and Viktor Fischer†
*Department of Electronics & Multimedia Communications, Technical University of Kosice
Park Komenskeho 13, 041 20 Kosice, Slovakia
Email: {martin.petrvalsky,milos.drutarovsky}@tuke.sk
†Hubert Curien Laboratory, Jean Monnet University
18, Rue du Prof. B. Lauras, 18, 42000 Saint-Etienne, France
Email: {tania.richmond,fischer,pierre.louis.cayrel}@univ-st-etienne.fr

***Abstract*—In this paper, we present a novel countermeasure against a simple power analysis based side channel attack on a software implementation of the McEliece public key cryptosystem. First, we attack a straightforward C implementation of the Goppa codes based McEliece decryption running on an ARM Cortex-M3 microprocessor. Next, we demonstrate on a realistic example that using a "chosen ciphertext attack" method, it is possible to recover the complete secret permutation matrix. We show that this matrix can be completely recovered by an analysis of a dynamic power consumption of the microprocessor. Then, we estimate the brute-force attack complexity reduction depending on the knowledge of the permutation matrix. Finally, we propose an efficient software countermeasure having low computational complexity. Of course, we provide all the necessary details regarding the attack implementation and all the consequences of the proposed countermeasure especially in terms of power consumption.**

I. Introduction

The code-based cryptosystems are very attractive because of their robustness regarding attacks based on the use of quantum computers. The first code-based cryptosystem was proposed by R. McEliece in 1978 [1]. However, it appeared that the code-based cryptosystems are as vulnerable to side channel attacks (SCA) proposed by Kocher in 1996 [2] as other cryptosystems. The first known SCA against the McEliece public key cryptosystem (PKC) appeared in 2008 [3].

Since the first published attack, several other vulnerabilities were discovered [4]–[12]. In this paper, we propose a new countermeasure against a variant of the attack described in [6]. In our attack, we target a straightforward C implementation of the syndrome computation on the ARM Cortex-M3 microprocessor [13]. We explain on a realistic example using simple power analysis (SPA) and chosen ciphertext attack (CCA) that the secret permutation matrix can be completely recovered.

After introducing the context, we start the paper by presenting Goppa codes and the McEliece PKC in Section II. Next, we briefly introduce the state of the art in Section III. We describe all necessary implementation details regarding considered SPA attack in Section IV. Furthermore, we provide an efficient countermeasure against the implemented attack in Section V. Finally, we conclude the paper in Section VI.

II. Theoretical background

A. Goppa codes

Goppa codes represent a large class of linear error-correcting codes proposed in 1970 [14], [15]. However, our interest is focused exclusively on irreducible binary Goppa codes that are commonly used in cryptography. For the sake of simplicity, we will call them simply Goppa codes. Given a monic irreducible polynomial $g(x)$ over $\mathbb{F}_{2^m}[x]$ with $\deg(g) = t$ and a set $\mathcal{L} = \{\alpha_1, \alpha_2, .., \alpha_n\}$ representing a subset of $\mathbb{F}_{2^m}$ such that $g(\alpha_i) \neq 0$, the Goppa code is defined as:
$\Gamma(\mathcal{L}, g) = \{C \in \mathbb{F}_2^n \mid \mathcal{S}_C(x) \equiv 0 \mod g(x)\}$.
We call a polynomial associated to $C \in \mathbb{F}_2^n$ the syndrome polynomial:
$\mathcal{S}_C(x) = \sum_{i=1}^{n} \frac{C_i}{x \oplus \alpha_i}$.

For decoding a binary Goppa codeword containing errors, one commonly adopted solution is to use the so-called Patterson's algorithm [16]. We will focus on the first step of this algorithm consisting in computing a product of a parity check matrix of the Goppa code denoted $\mathcal{H}$ and the codeword with less than or equal to t errors denoted C, i.e. $S = C\mathcal{H}^T$. The result of this operation is called the syndrome and it can be viewed as a polynomial as $S_C(x) = [x^{t-1}, \ldots, x, 1]S$.

B. The McEliece cryptosystem

The McEliece PKC using Goppa codes [1] is performed using three operations: the key generation, plaintext encryption and ciphertext decryption.

The key generation consists in the determination of the Goppa code according to the definition given in Section II-A. As the Goppa code is linear, it 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 $\mathcal{P}$. We compute the public $k \times n$ generator matrix as $\tilde{\mathcal{G}} = \mathcal{S}\mathcal{G}\mathcal{P}$. The key generation procedure outputs the secret key $\mathsf{sk} = (\Gamma(\mathcal{L}, g), \mathcal{S}, \mathcal{P})$ and the public key $\mathsf{pk} = (n, t, \tilde{\mathcal{G}})$.

During the plaintext encryption, the message M is encrypted using the public generator matrix. This operation can be expressed as $C = M\tilde{\mathcal{G}}$. Next, an error vector E of length n

and weight t is randomly selected and added to the codeword, giving the ciphertext $\tilde{C} = C \oplus E$.

During the decryption of the ciphertext $\tilde{C}$, the product $\tilde{C}\mathcal{P}^{-1}$ must first be computed giving a codeword containing an error, i.e. $M\mathcal{S}\mathcal{G} \oplus E\mathcal{P}^{-1}$. Then, a decoding algorithm (the Patterson's algorithm in our case) must be applied on the obtained secret code. The attack described in Section IV targets this phase of the ciphertext decryption. The obtained $M\mathcal{S}\mathcal{G}$ is multiplied by $\mathcal{G}_r^{-1}$ on the right side, such that $\mathcal{G}\mathcal{G}_r^{-1} = \mathcal{I}_k$ is the $k \times k$ identity matrix, in order to find $\tilde{M} = M\mathcal{S}$. Finally we compute $M = \tilde{M}\mathcal{S}^{-1}$ to recover the plaintext.

III. Existing side channel attacks

Several side channel attacks against the McEliece PKC were published recently. Most of attacks target the Patterson's decoding algorithm and exploit different weaknesses. The most common are timing attacks aiming either the message [3], [4], [8] or the private key recovery [5], [10]. Some fault injection attacks are also known, e.g. that published in [7], based on two variants of Goppa codes. A combined timing and fault injection attack targeting the message recovery was proposed in [17]. Finally, the attacks using SPA like those published in [6], [9] or [12] (for another type of codes) or attacks using differential power analysis (DPA) [11] (again, for different type of codes) tend to recover the private key.

In this paper, we focus on the kind of power analysis attacks proposed in [6]. Based on this principle, we implement an attack against the syndrome computation. Next, we propose a countermeasure featuring a linear computational complexity, which uses similar idea to that published in [3, Algorithm 3]. However, contrary to our solution, this countermeasure is focused on another type of attack and it has a quadratic computational complexity.

Four implementation profiles were introduced in the paper [6]. In profile I, rows of the parity check matrix are computed as they are needed. Profile II uses precomputed parity check matrix. Profiles III and IV omit multiplication with permutation matrix in the first step of the decryption. In profile III, syndrome is directly computed from permuted support. Profile IV uses precomputed and permuted parity check matrix. Profile I is favorable for embedded devices due to low memory requirements. Our countermeasure can be used for profiles I and II (profile I only if all computations are constant in time). For testing purposes, we use profile II due to simplified measurements. If profiles III and IV are used than the SPA attack and our countermeasure are not applicable because the permutation matrix is merged with the parity check matrix or with the support $\mathcal{L}$.

IV. SPA attack details

A. Measurement setup

We attack a software implementation of the Patterson's decryption algorithm running on the STM32F103 microcontroller [18] featuring a 32-bit ARM Cortex-M3 core. The STM microcontroller is fitted on a custom evaluation board [19] aimed at power analysis attack implementations. The board is equipped with a serial port connector for data transfers. It has two SMA connectors for oscilloscope probes, which are used to get the power consumption traces. We measure power consumption near the grounding pin of the microcontroller. The on-chip PLL of the microcontroller generates the core clock of 72 MHz from an external quartz oscillator oscillating on 8 MHz. The main features of the selected microcontroller are:

- Instruction set Thumb2,
- 20 kB RAM, 128 kB Flash,
- ADC, USB, RTC, USART, 7 timers, ...

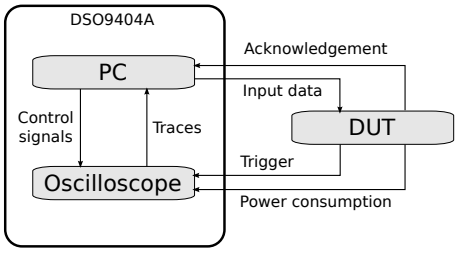

Fig. 1. Workflow of the power traces acquisition – the whole process is controlled by the oscilloscope's PC

The power traces were acquired using the Agilent Technologies oscilloscope DSO9404A [20] with four analog channels having 4 GHz bandwidth at 50Ω. The power traces acquisition workflow is depicted in Fig. 1

All power traces needed for attacking unprotected device are acquired at 1 GS/s sample rate. Power traces obtained after the countermeasure deployment was collected at 100 MS/s sample rate in order to reduce the acquired data to the acceptable level for the oscilloscope. The 500 MHz passive probes were connected directly to the SMA connectors available on the evaluation board.

The data acquisition process is controlled by the software running on the oscilloscope's internal PC. The software sets up the oscilloscope, sends the ciphertext to the microcontroller (design under test – DUT in Fig. 1) via UART and waits for the acknowledgment from the microcontroller. DUT rises a trigger and starts the ciphertext decryption. The oscilloscope measures the power consumption during the ciphertext decryption. Once the acquisition is finished, the PC stores the measured power consumption trace to the hard disk. The measurement process is repeated depending on the desired number of traces. The acquired traces are ready for further processing.

B. Principle of the SPA attack on the syndrome computation

As explained earlier, our idea is derived from the attack published in [6]. It is a kind of the SPA attack, which can recover the secret permutation matrix. Its principle is illustrated in Fig. 2.

The permutation matrix is the first private information used in the McEliece decryption. The input ciphertext is permuted using a permutation matrix. This operation is implemented as a binary vector-matrix multiplication. The next step after the permutation is the syndrome computation $S = C\mathcal{H}^T$, which is also implemented as a binary vector-matrix multiplication. In a straightforward implementation, the binary multiplication

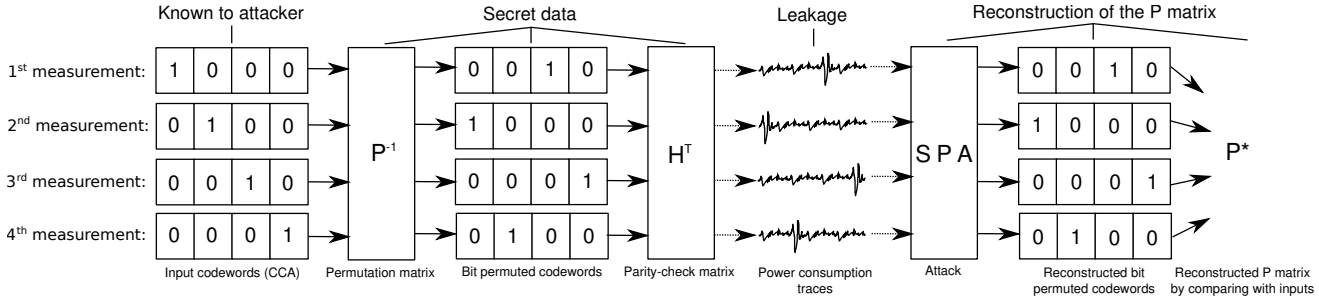

Fig. 2. Principle of the SPA attack on the syndrome computation: the chosen ciphertext has one bit equal to one in a row, data are then permuted and multiplied with the parity check matrix in order to get the syndrome – the permutation matrix can be reconstructed by analyzing power traces of different chosen ciphertexts

is realized by adding modulo 2 only the rows of the matrix that have indexes of the vector bits equal to one. Rows having indexes where the vector bits are equal to zero are omitted.

The SPA on the syndrome computation is based on detecting special parts of the algorithm in power consumption traces. The sequences, where the row of the matrix is added modulo 2 and sequences where the row addition is omitted have different patterns in power consumption traces as it can be seen in Fig. 3b).

We generated subsequent input ciphertexts having a Hamming weight equal to 1. For each measurement, the isolated bits equal to 1 were placed in a different position. Once the permutation matrix was applied on the input ciphertext, the bit was permuted to another (unknown) position for the new measurement.

Observing the traces of the syndrome computation can help in detecting the modulo 2 additions. If the attacker is able to locate every position of the modulo 2 additions during the syndrome computation for every possible input ciphertext containing only one bit equal to one, the reconstruction of the permutation matrix is possible as shown in Fig 2.

C. From the knowledge of $\mathcal{P}$ to the knowledge of $g(x)$

Once the attacker found the private permutation matrix $\mathcal{P}$, the order of elements in the support $\mathcal{L}$ is revealed. The only remaining part of the private key is the Goppa polynomial $g(x)$. Indeed, the knowledge of the scrambling matrix $\mathcal{S}$ is useless because $\mathcal{SG}$ and $\mathcal{G}$ generate the same code.

A way to find $g(x)$ from $\mathcal{P}$ is to see the Goppa code as an alternant code [21] i.e. a sub-code of a generalized Reed-Solomon (GRS) code on the sub-field $\mathbb{F}_2$. The public Goppa code, defined by $\mathcal{SGP} = \tilde{\mathcal{G}}$, is equivalent to the private Goppa code, defined by $\mathcal{G}$. Since the dual code of a GRS code is another GRS code [21, chapter 10], the attacker can compute a parity check matrix denoted $\tilde{\mathcal{H}}$ from the public generator matrix $\tilde{\mathcal{G}}$. The $\tilde{\mathcal{H}}$ matrix is a generator matrix of the dual code, so $\tilde{\mathcal{G}}\tilde{\mathcal{H}}^T = 0$. This relation provides a system of linear equations with n unknowns After solving this system, these unknowns correspond to the evaluation of the Goppa polynomial on each element on the support $\mathcal{L}$. By a Lagrange polynomial interpolation, the attacker can recover all coefficients of the Goppa polynomial $g(x)$..

The complexity of solving the system of linear equations with n unknowns is roughly $\mathcal{O}(n^3)$ and that of the Lagrange interpolation is about $\mathcal{O}(n^2)$. The cost of multiplications in $\mathbb{F}_{2^m}$ is m^2. This means that the complexity of the given attack is $m^2(n^3 + n^2)$ binary operations. In example given in the following subsection, the attack complexity can be decreased to 2^{37} binary operations from the original complexity of 2^{62} [22]. This represents a critical threat for the decryption algorithm.

D. Realistic example of the CCA based on SPA

Usual implementations of the McEliece PKC use codewords of $n = 1,024$, $n = 2,048$ and more bits. In this work, we chose the codeword of $n = 1,024$ bits. Parameter $n = 1,024$ provides the level of security sufficient only for short-term uses. For more secure embedded implementation, at least $n = 2,048$ should be used. Of course, the attack can be easily modified for any other code lengths.

As explained in the previous section, we use the chosen ciphertext method for attacking the syndrome computation. We measure 1,024 traces corresponding to processing of input ciphertexts chosen as follows:

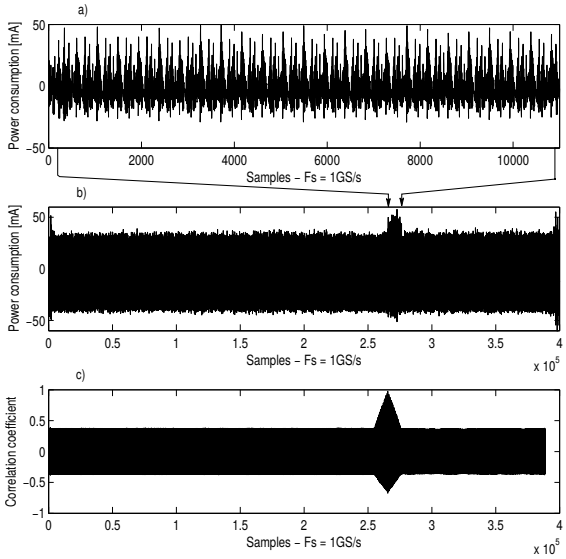

Fig. 3. Typical power traces and correlation traces of the SPA using chosen ciphertext: Top panel (a) – power trace of 1024 consecutive additions modulo 2 featuring a typical pattern (zoomed from the middle panel (b)); Middle panel (b) – power trace of one chosen ciphertext decryption; Bottom panel (c) – correlation trace obtained from the power trace of the middle panel.

- 10000000000...0 for the first measurement,
- 01000000000...0 for the second measurement,
- ...,
- 000...000000001 for the last measurement.

An example of traces acquired for 1,024 bit codewords is presented in Fig. 3. Fig. 3b) shows one typical 400,000-sample power trace of the ARM Cortex-M3 during one syndrome computation. In all other traces, the pattern of the modulo 2 addition appeared at different time. This pattern appearance depends on the position of the value '1' in corresponding row of the permutation matrix. Indeed, we consider that the permutation is not seen as a vector of positions but as a permutation matrix corresponding to the linear application. This means that there is just one '1' in each row and each column. Our aim is to find all values of '1' in this matrix.

The correlation traces were obtained by computing correlation coefficients between modulo 2 additions pattern from Fig. 3a) and a sliding window of currently measured power trace (e.g. that depicted in Fig. 3b). As can be observed, it is possible to locate the modulo 2 addition pattern at around 270,000$^{\text{th}}$ sample, for which the correlation coefficient is equal almost to 1.

We developed a software for automatic evaluation of the traces. It computes correlation traces as shown in Fig. 3c), determines the exact location of the beginning of the modulo 2 addition pattern and finally it sorts these positions in the whole trace set in order to reconstruct the $\mathcal{P}$ matrix.

In a straightforward implementation, the permutation matrix $\mathcal{P}$ could successfully be deduced from positions of modulo 2 additions during the syndrome computation for all measured traces.

In most experiments, we were able to extract the permutation matrix stored in the Flash memory of the ARM Cortex-M3 with a 100% success rate. There were only few cases where we missed only one position of value '1' in the whole 1,024×1,024 matrix. Due to permutation matrix properties, if we miss one value another value is also pushed out of its correct position. Incorrect position detection can be minimized by improving the data processing algorithm - trace aligning, filtration, averaging, choosing better modulo 2 addition pattern and using more robust correlation analysis.

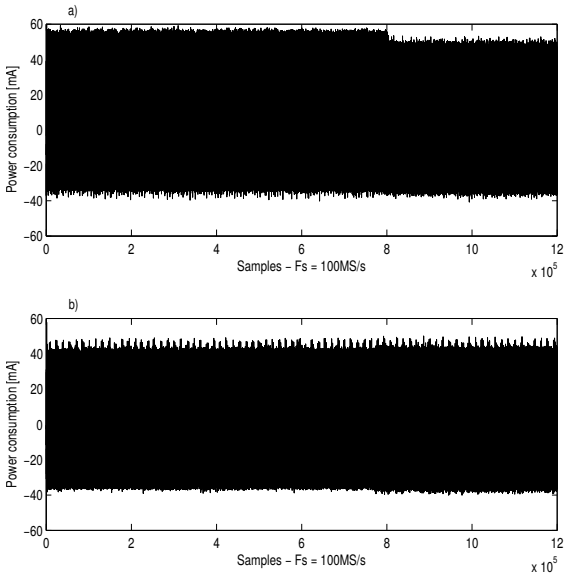

Fig. 4. Power traces after a straightforward countermeasure (a) and an advanced countermeasure (b) against the simple power analysis attack on syndrome computations

V. SPA RESISTANT COUNTERMEASURE

Our countermeasure is based on a basic principle of avoiding branch statements and data dependent timings. Next, we provide a C implementation of both insecure and protected syndrome computation:

```
for(i=0; i<DIM_N; i++)                    // insecure s = H*cp
 if (MPL_TESTBIT(cp, i) == TRUE)          // @72MHz: 4ms avg
  MPL_XOR(s, s, H[i], j, DIM_NK_WORD);// range: 0.4ms - 8ms
//-------------------------------------------------
for(j=0; j<DIM_NK_WORD; j++)              // mask syndrome
 s[j]^=0xAAAA;
for(i=0; i<DIM_N; i++){                   // protected s = H*cp
 mul = 0 - (MPL_TESTBIT(cp, i));          // @72MHz: 12ms
 for(j=0; j<DIM_NK_WORD; j++) s[j]^=H[i][j] & mul;
}
for(j=0; j<DIM_NK_WORD; j++)                // unmask syndrome
 s[j]^=0xAAAA;
```

In the code, s is the computed syndrome of size $(n-k)$ bits, cp is the bit-permuted ciphertext of n bits, $\mathcal{H}$ is the parity check matrix of $n \times n$ bits, DIM_N is the parameter $n = 1024$, DIM_NK_WORD is the number of words needed to store $(n-k)$-bit variable ($380/sizeof(s)$ corresponding to $k = 624$, $m = 10$ and $t = 38$), mul is a temporary variable indicating that the row of the binary matrix should be added (all bits '0' for no addition; all bits '1' for corresponding row addition).

In a straightforward unprotected implementation, one scans the codewords bit by bit. If he finds a '1' value, he adds the corresponding row of the multiplied matrix to the result (the syndrome). These additions create characteristic patterns in power traces, which can serve as basis for the SPA attack. In our solution, we perform identical steps for each bit of the codeword, which removes special patterns in power traces such as patterns in Fig. 3b). Furthermore, we need a special initialization of the syndrome variable, in order to remove the effect presented in Fig. 4.

Figure 4a) shows 1,200,000-sample power trace after applying simple above mentioned countermeasure. However, we noticed that appearance of '1' values in bit-permuted ciphertext was still visible. It can be seen in top panel of Fig. 4 that the positive peaks of the power traces reach about 55 mA. Around the 800,000$^{\text{th}}$ sample (where the value '1' was handled), these values drop down to 50 mA. This power decrease is caused by changing the syndrome variable. In a straightforward implementation, this variable is initialized to zero at the beginning of the algorithm. In every iteration, the syndrome is loaded and then saved into the memory. Registers in the ARM Cortex-M3 are pre-charged to value 0xFFFFFFFF and then the new values are written to them. If the current written value is zero, the consumption is the highest because of the flip-flop switching and vise versa. As a basis for the new countermeasure, we used the fact that the rows and columns of the parity check matrix $\mathcal{H}$ have approximately the same amount of ones and zeros. We

successfully used this property to hide the visible amplitude decrease from Fig. 4a) and initialized the syndrome to the masked value '10101010...1010' before the multiplication. The result of application of this new countermeasure is shown in Fig. 4b) where no clearly visible voltage drops are present.

VI. Conclusion

In this paper, we successfully deployed an SPA attack targeting the computation of the private permutation matrix of the McEliece PKC on an ARM Cortex-M3 based microcontroller. We showed that without a countermeasure, we were able to recover the whole 1,024 $\times$ 1,024 bit permutation matrix using just 1,024 power traces. Next, we quantified the security threat represented by the revelation of the permutation matrix in terms of the computational complexity. Finally, we proposed, implemented and successfully tested an efficient countermeasure against this SPA attack.

The disadvantage of our implementation of the countermeasures is that it needs longer time for the multiplication – our implementation is about 3 times slower than a straightforward unprotected implementation on average. However, in comparison with [3, Algorithm 3], our solution provides a linear time complexity compared to a quadratic time complexity proposed in the previous paper.

The countermeasure proposed here ensures that the power consumption and execution time are constant, which prevents SPA attacks and timing attacks. On the other hand, the constant time algorithm does not avoid other kinds of more sophisticated attacks (such as DPA) to be deployed.

Acknowledgment

This work was performed in the framework of the COST Action IC1204 (Trustworthy Manufacturing and Utilization of Secure Devices). It was supported by APVV-0586-11 grant and in part by NATO's Public Diplomacy Division in the framework of "Science for Peace", SPS Project 984520. The authors would also like to thank Alain Couvreur for his helpful advices.

References

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

[2] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Advances in Cryptology (CRYPTO'96)*, ser. LNCS, N. Koblitz, Ed., vol. 1109. Springer, 1996, pp. 104–113. [Online]. Available: http://dx.doi.org/10.1007/3-540-68697-5_9

[3] F. Strenzke, E. Tews, H. G. Molter, R. Overbeck, and A. Shoufan, "Side channels in the McEliece PKC," in *The Second International Workshop on Post-Quantum Cryptography (PQCrypto 2008)*, ser. LNCS, J. Buchmann and J. Ding, Eds. Springer, October 2008, vol. 5299, no. 5299/2008, pp. 216–229. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88403-3_15

[4] A. Shoufan, F. Strenzke, H. G. Molter, and M. 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)*, ser. LNCS, D. Lee and S. Hong, Eds. Springer, 2010, vol. 5984, pp. 161–175. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-14423-3_12

[5] F. 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)*, ser. LNCS, N. Sendrier, Ed. Springer, 2010, vol. 6061, pp. 95–107. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-12929-2_8

[6] S. Heyse, A. Moradi, and C. Paar, "Practical power analysis attacks on software implementations of McEliece," in *Proceedings of the Third international conference on Post-Quantum Cryptography (PQCrypto 2010)*, ser. LNCS, N. Sendrier, Ed. Springer, 2010, vol. 6061, pp. 108–125. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-12929-2_9

[7] P.-L. Cayrel and P. Dusart, "McEliece/Niederreiter PKC: Sensitivity to fault injection," in *5th International Conference on Future Information Technology (FutureTech 2010)*, May 2010, pp. 1–6.

[8] R. M. Avanzi, S. Hoerder, D. Page, and M. 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. [Online]. Available: http://dx.doi.org/10.1007/s13389-011-0024-9

[9] H. G. Molter, M. Stöttinger, A. Shoufan, and F. Strenzke, "A simple power analysis attack on a McEliece cryptoprocessor," *Journal of Cryptographic Engineering*, vol. 1, no. 1, pp. 29–36, April 2011. [Online]. Available: http://dx.doi.org/10.1007/s13389-011-0001-3

[10] F. Strenzke, "Timing attacks against the syndrome inversion in code-based cryptosystems," in *The 5th International Workshop on Post-Quantum Cryptography (PQCrypto 2013)*, ser. LNCS, P. Gaborit, Ed. Springer, 2013, vol. 7932, pp. 217–230, http://eprint.iacr.org/2011/683. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-38616-9_15

[11] C. Chen, T. Eisenbarth, I. von Maurich, and R. Steinwandt, "Differential power analysis of a McEliece cryptosystem," Cryptology ePrint Archive, Report 2014/534, 2014, http://eprint.iacr.org/2014/534.

[12] I. von Maurich and T. Güneysu, "Towards side-channel resistant implementations of QC-MDPC McEliece encryption on constrained devices," in *Post-Quantum Cryptography*, ser. LNCS, M. Mosca, Ed. Springer, October 2014, vol. 8772, pp. 266–282. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-11659-4_16

[13] ARM, "ARM Cortex-M product information, software and datasheets." [Online]. Available: http://www.arm.com/products/processors/cortex-m/

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

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

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

[17] F. Strenzke, "Message-aimed side channel and fault attacks against public key cryptosystems with homomorphic properties," *Journal of Cryptographic Engineering*, vol. 1, no. 4, pp. 283–292, 2011. [Online]. Available: http://dx.doi.org/10.1007/s13389-011-0020-0

[18] ST Microelectronics, "STM32 product information, software and datasheets." [Online]. Available: http://www.st.com/web/en/catalog/mmc/FM141/SC1169

[19] M. Petrvalsky, M. Drutarovsky, and M. Varchola, "Differential power analysis attack on ARM based AES implementation without explicit synchronization," in *Radioelektronika (RADIOELEKTRONIKA), 2014 24th International Conference*, April 2014, pp. 1–4.

[20] Keysight (Agilent Technologies), "DSO9404A datasheet and product information." [Online]. Available: http://www.keysight.com/en/pd-1632456-pn-DSO9404A/oscilloscope-4-ghz-4-analog-channels?&cc=SK&lc=eng

[21] F. J. MacWilliams and N. J. A. Sloane, *The theory of error-correcting codes*, N.-H. M. Library, Ed. North-Holland, 2006.

[22] D. Bernstein, T. Lange, and C. Peters, "Attacking and defending the mceliece cryptosystem," in *Post-Quantum Cryptography*, ser. Lecture Notes in Computer Science, J. Buchmann and J. Ding, Eds. Springer Berlin Heidelberg, 2008, vol. 5299, pp. 31–46. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88403-3_3