A Side-Channel Attack Against the Secret Permutation on an Embedded McEliece Cryptosystem

Tania Richmond, Martin Petrvalsky, Milos Drutarovsky

A Side-Channel Attack Against the Secret Permutation on an Embedded McEliece Cryptosystem

Tania Richmond
Hubert Curien Laboratory
Jean Monnet University
Rue du Prof. Benoit Lauras, 18
42000 Saint-Etienne
France
Email: tania.richmond@univ-st-etienne.fr

Martin Petrvalsky
and Milos Drutarovsky
Department of Electronics & Multimedia Communications
Technical University of Kosice
Park Komenskeho 13
041 20 Kosice
Slovakia
Email: {martin.petrvalsky,milos.drutarovsky}@tuke.sk

***Abstract*—In this paper, based on a thorough analysis of the state of the art, we point out a missing solution for embedded devices to secure the syndrome computation. We show that this weakness can open the door to a side-channel attack targeting the secret permutation. Indeed, brute-force attack iterations are dramatically decreased when the secret permutation is recovered. We demonstrate the feasibility of this attack against the McEliece cryptosystem implemented on an ARM Cortex-M3 microprocessor using Goppa codes. We explain how to recover the secret permutation on a toy example. Finally, we propose a promising countermeasure, which can be implemented in embedded devices to prevent this attack.**

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], [5], [6], [7], [8], [9], [10], [11], [12], [13]. 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 [14]. We explain on a toy example using simple power analysis (SPA) and chosen ciphertext attack (CCA) that the secret permutation matrix can be completely recovered.

This paper is organized as follows. We give backgrounds on Goppa codes and the McEliece cryptosystem in Section II to fix notations. Then we briefly explain the state of the art in Section III. We detail our proposal, give the general idea and provide a toy example of our attack in Section IV. Finally we conclude this paper in Section V.

II. Theoretical background

A. Goppa codes

Goppa codes represent a large class of linear error-correcting codes proposed in 1970 [15], [16]. 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 [17]. 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 $C = M\mathcal{G}$. 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], [11]. 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 [18]. Finally, the attacks using SPA like those published in [6], [9] or [13] (for another type of codes) or attacks using differential power analysis (DPA) [12] (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. Our proposal

We developed our idea from [6] where the SPA attack on the permutation matrix was realized. We proposed that this attack can be avoided by changing algorithm of matrix multiplication. Firstly, we performed the attack on a toy example. After, we modified the algorithm to secure the multiplication.

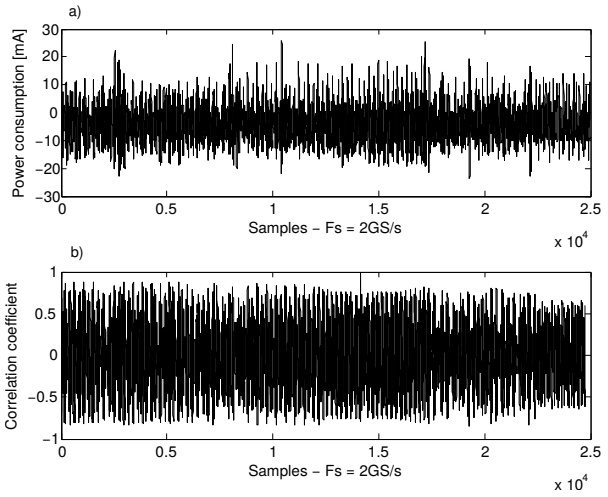

Fig. 1. Figure a) shows 25,000 samples of measured power consumption of ARM Cortex-M3 during matrices multiplication. Figure b) represents correlation trace which was obtained by computing correlation coefficient between $\oplus$ instruction pattern and sliding window of currently measured power trace. As we can see, we can locate $\oplus$ operation around 14,000th sample where the coefficient almost equal to 1. In straightforward implementation, we can deduce permutation matrix $\mathcal{P}$ from positions of $\oplus$ instruction (from syndrome computation) in measured traces.

A. Toy example

Straightforward implementation of permutation and parity-check matrices multiplication was developed for ARM Cortex-M3 microcontroller. We chose length of the ciphertext $n = 8$ bits as a first approach. Afterwards, a power consumption measurements and SPA attack was performed on the implementation (Fig. 1). We managed to recover permutation matrix which was stored in Flash memory of the microcontroller with 100% success rate. Process of recovering $\mathcal{P}$ matrix is depicted in Fig. 2.

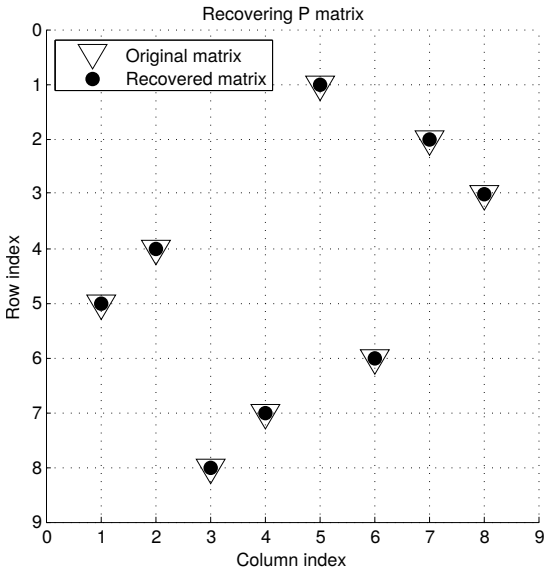

Fig. 2. In each measurement we sent all possible 8-bit ciphertexts with Hamming weight equal to 1. In power consumption traces the $\oplus$ pattern appears in different times depending on input ciphertext. By sorting positions of the $\oplus$ pattern appearences we can extract $\mathcal{P}$ matrix.

B. Countermeasure

For our toy example we propose algorithm which can avoid the SPA attack. The basic principle is to avoid branch statements and data dependent running time. We provide example of unsecure and secure syndrome computations:

```
for(i=0; i<DIM_N; i++) // STRAIGHTFORWARD s = H*cp
 if ((cp>>i)&0x01) s^=H[DIM_N-1-i];
//-----------------------------------------------
for(i=0; i<DIM_N; i++) // SECURE s = H*cp
 s^=H[DIM_N-1-i]*((cp>>i)&0x01);
```

where s is computed syndrome, DIM_N is equal to 8, cp is 8-bit long permuted ciphertext and $\mathcal{H}$ is parity-check matrix.

V. Conclusion

In this paper we managed to perform SPA attack on the toy example which targets the secret permutation matrix in the McEliece cryptosystem on a microcontroller implementation. We also proposed coutermeasure which can efficiently avoid the SPA and timing analysis attacks.

In future research, we will apply the same principles for full range McEliece algorithm with $n = 1024$ and $n = 2048$ bits. Afterwards, we will examine other possible attacks on our implementation of permutation and parity-check matrices multiplications.

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.

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. Lecture Notes in Computer Science (LNCS), N. Koblitz, Ed., vol. 1109. Berlin, Heidelberg: Springer, 1996, pp. 104–113. [Online]. Available: http://www.springerlink.com/content/4el17cvre3gxt4gd/

[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. Lecture Notes in Computer Science (LNCS), J. Buchmann and J. Ding, Eds. Berlin Heidelberg: 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. Lecture Notes in Computer Science (LNCS), D. Lee and S. Hong, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 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. Lecture Notes in Computer Science (LNCS), N. Sendrier, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 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. Lecture Notes in Computer Science (LNCS), N. Sendrier, Ed. Berlin Heidelberg: 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, "Fast and secure root finding for code-based cryptosystems," in *Cryptology and Network Security*, ser. Lecture Notes in Computer Science (LNCS), J. Pieprzyk, A.-R. Sadeghi, and M. Manulis, Eds. Springer Berlin Heidelberg, December 2012, vol. 7712, pp. 232–246. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35404-5_18

[11] ——, "Timing attacks against the syndrome inversion in code-based cryptosystems," in *The 5th International Workshop on Post-Quantum Cryptography (PQCrypto 2013)*, ser. Lecture Notes in Computer Science (LNCS), P. Gaborit, Ed. Berlin Heidelberg: 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

[12] 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.

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

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

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

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

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

[18] 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