

# Fault injection's sensitivity of the McEliece PKC

Pierre-Louis Cayrel<sup>1</sup> and Pierre Dusart<sup>2</sup>

1 - Université Paris VIII  
Département de Mathématiques  
2, rue de la Liberté  
93526 - SAINT-DENIS cedex 02, France  
pierreloiscayrel@gmail.com  
2 - Université de Limoges, XLIM-DMI,  
123, Av. Albert Thomas  
87060 Limoges Cedex, France.  
pierre.dusart@xlim.fr

June 3, 2009

**Abstract.** The McEliece public key cryptosystem (PKC) is supposed secure in a post quantum world [2] because there is no efficient quantum algorithm for the underlying problems, which this cryptosystem is built upon. The purpose of this article is to describe in what the structure of the McEliece PKC is sensitive to fault injection. We present the injection fault in the McEliece scheme using Goppa codes and in a variant using quasi-cyclic alternant codes, and describe the main difference of those two constructions in this context.

## 1 Introduction

The McEliece cryptosystem presents a good alternative to classic number theory encryption scheme. Its security has been studied for years and a lot of distinct constructions have been proposed in order to reduce the huge size of public keys. After several improvements, the use of quasi-cyclic alternant codes leads to a public key of 6500 bits (see [1]). In the real world, it's important to study side channel attacks in hardware environment. A first study of the McEliece scheme against side channel attacks has been done in [6]. We present here an other kind of attack, the fault injection attack and show how the quasi-cyclic construction presents a weakness in this context. We focus our analysis on the encryption scheme (that can be the case for satellite communication where only the encryption is realized).

### Our contribution

In their paper [6], the authors deal with side channel attacks against McEliece scheme (timing attacks) but they don't deal with fault attack.

We propose in this paper the study of the security of the McEliece PKC against a specific side channel attack, the fault attack. Due to its structure based on error correcting codes the scheme correct by himself the fault injected. We will focus on the parameters use to be secure against several fault injections.

This contribution shows one of the good properties of the McEliece scheme in a real context (except with the use of quasi-cyclic matrices). The classic scheme is secure by construction against this kind of attack.

The study of the decryption algorithm, a decoding algorithm of error correcting codes, against fault injection is out of the scope of this article.

### Organisation

This paper is constructed as follows. Section 2 presents as preliminaries the McEliece PKC in brief. Section 3 details fault attack. Section 4 outlines the security of the McEliece PKC against this kind of attacks. Section 5 concludes the paper.

## 2 Preliminaries

### 2.1 The McEliece PKC

The McEliece PKC [5] represents one of the oldest public-key cryptosystem ever designated. It is also the first public-key cryptosystem based on linear error-correcting codes. The principle is to select a linear code of length  $n$  and dimension  $t$  that is able to efficiently correct  $t$  errors. The core idea is to transform it to a random-looking linear code. A description of the original code and the transformations can serve as the private key while a description of the modified code serves as the public key. McEliece's original proposal uses a generator matrix of a binary Goppa code. The encryption function encodes a message according to the public code and adds an error vector of weight  $t$ . The decryption function basically decodes the ciphertext by recovering the secret code through the trapdoor which consists of the transformations. Niederreiter [7] also proposed a public-key cryptosystem based on linear codes in which the public key is a parity-check matrix. It is proved in [4] that these two systems are equivalent in terms of security.

The McEliece cryptosystem [5] uses error-correcting codes that have an efficient decoding algorithm in order to build trapdoor one-way functions.

McEliece proposed binary Goppa codes as the underlying family of codes. The parameters of a binary Goppa code are  $[2^m, 2^m - mt, \geq 2t + 1]$  where  $m$  and  $t$  are non-negative integers. Additionally, a binary Goppa code can be decoded by an efficient  $t$ -bounded decoding algorithm. The principle of the McEliece cryptosystem is to randomly pick a code  $\mathcal{C}$  among the family of binary Goppa codes. The private key is the Goppa polynomial of  $\mathcal{C}$ . The public key will be a generator matrix that is obtained from the private key and by two random linear transformations: the *scrambling* transformation  $S$ , which sends the secret matrix  $G$  to another generator matrix, and a permutation transformation which reorders the columns of the secret matrix. Figure 1 and Figure 2 give details of the three algorithms.

**Fig. 1.** Key generation algorithm of the McEliece cryptosystem

KeyGen( $1^\kappa$ ) ( $\kappa$  is the security parameter)

1. Choose  $n$ ,  $k$  and  $t$  according to  $\kappa$
2. Randomly pick a generator matrix  $G_0$  of an  $[n, k, 2t + 1]$  binary Goppa code  $\mathcal{C}$
3. Randomly pick a  $n \times n$  permutation matrix  $P$
4. Randomly pick a  $k \times k$  invertible matrix  $S$
5. Calculate  $G = S \times G_0 \times P$
6. Output  $\text{pk} = (G, t)$  and  $\text{sk} = (S, G_0, P, \gamma)$  where  $\gamma$  is a  $t$ -bounded decoding algorithm of  $\mathcal{C}$

**Fig. 2.** Encryption and decryption algorithms of the McEliece cryptosystem

- |  |   |
|--|---|
| <p>Encrypt(<math>\text{pk}, m \in \mathbb{F}_2^k</math>)</p> <ol style="list-style-type: none"> <li>1. Randomly pick <math>e</math> in <math>\mathbb{F}_2</math> of weight <math>t</math></li> <li>2. Calculate <math>c = m \times G + e</math></li> <li>3. Output <math>c</math></li> </ol> | <p>Decrypt(<math>\text{sk}, c \in \mathbb{F}_2^n</math>)</p> <ol style="list-style-type: none"> <li>1. Calculate <math>z = c \times P^{-1}</math></li> <li>2. Calculate <math>y = \gamma(z)</math></li> <li>3. Output <math>m = y \times S^{-1}</math></li> </ol> |
|--|---|

**Parameters of the McEliece PKC (using Goppa Codes)** The security parameters  $\kappa, m \in \mathbb{N}$  and  $t \in \mathbb{N}$  with  $t \ll 2m$  have to be chosen in order to set up a McEliece PKC. An example for secure values would be  $m = 11, t = 50$  for a Goppa code of parameters  $[2^m, 2^m - mt, \geq 2t + 1]$ .

**Parameters of the McEliece PKC (using quasi-cyclic alternant codes)** In [1], the authors present a new way to describe the public key matrix using quasi-cyclic alternant code  $\mathcal{C}$  of parameters  $[459, 255, 101]_{2^8}$  for a public key of 8,100 bits and a security of  $2^{80}$  binary operations and a code of parameters  $[612, 408, 101]_{2^8}$  for a public key of 13,000 bits and a security of  $2^{100}$  binary operations.

The next tables from [1] describes this parameters

**Table 1.** Suggested parameters of alternant codes for different security levels

$q^m - 1 = N = N_0 \times l$			Public code $\mathcal{C}[n, k, \geq 2t + 1]_q$					
$\ell$	$N_0$	$t$	Parameter	$n$	$k$	$q$	security	Public key size (bits)
51	1,285	50	$A_{16}$	459	255	$2^8$	80	8,100
51	1,285	50	$B_{16}$	510	306	$2^8$	90	9,700
51	1,285	50	$C_{16}$	612	408	$2^8$	100	13,000
51	1,285	50	$D_{16}$	765	510	$2^8$	120	20,000
75	13,981	56	$A_{20}$	450	225	$2^{10}$	80	6,800
93	11,275	63	$B_{20}$	558	279	$2^{10}$	90	8,500
93	11,275	54	$C_{20}$	744	372	$2^{10}$	110	14,600

### 3 Fault Injection

Recently, in [3], an implementation of a code based cryptosystem (the Stern scheme) on a low resource device has been done. The securisation of the implemented scheme against side channel attack is the next step in a real world context. Furthermore, a description of small public key has been proposed in [1], the construction opens the way of an implementation on smart card.

The securisation of the McEliece scheme against side channel attacks has been studied in [6], we describe in this section the sensitivity of the McEliece PKC against fault injection and specially the weakness of the quasi-cyclic construction in this context.

#### 3.1 Description of fault injection

A system could make wrong computations. These computational errors are internal and can be created by internal components (bad quality of component with makes fuzzy electric noise) or external stress (specific areas are bombarded of a circuit board with heavy radiation). The external stress can be intensional (pirate's attack) or due to an specific environment (space radiations for satellite). Many hardware systems are sensible to this kind of attacks. In case of pirate's attack, the fault injection is the first step of the fault attacks. This fault can be exploit to find confidential data in secure devices or to compromise the system by creating a denial of service (DoS). Apply to a encryption device, a failure is randomly returning for some of the calls. Next it may propagate to the system boundary and be observable by the attacker. He can exploit theses faults to find secret keys in cryptographic system.

In case of disturbed environment, the device should be fault tolerant : the system should accomplish its mission in spite of the faults (possibly in degraded mode). If this device is satellite on board, there is no possibility to repair or change it. The system must be tested before the launching. The fault injection is often used with stress testing and is widely considered to be an important part of developing robust software.

Hence for specific environment or security context, the fault injection must be take into account.

#### 3.2 Countermeasures

The classical way to correct errors in encryption system is to compute the inverse function : if a message is encrypted then decrypted correctly, the encryption is correct (without errors). This solution is good but

impossible in some cases : public key encryption systems where the device knows the public key only or computed the decrypted value is too long. Another way is to have dedicated hardware (in fact two identical devices) which computes the encryption twice.

## 4 Fault Injection on the McEliece Encryption

### 4.1 Fault model

The fault model is the following : we suppose that there is one fault on the variables used. This fault is a memory error of one bit (in fact, it's a flip on a memory cell).

### 4.2 Fault on variables of McEliece Encryption

The McEliece encryption works in the following way : starting with a vector-message  $m$ , the product with encryption matrix  $G$  is computed and the error  $e$  is added. Hence

$$c = m \times G + e.$$

There are three cases to study depending on the variable touched by the fault :  $m$ ,  $G$  or  $e$ .

- If the input message  $m$  is touched by the fault and transformed to  $m'$ , the computation will be made like on another input. Hence the decryption operation will compute  $m'$  and not the good value  $m$ . Hence an error on the input message will be not corrected by the scheme.
- If we introduce a fault within a bit of the matrix (let us replace one 0 by one 1 at the  $i^{th}$  line,  $j^{th}$  column for example) then one has an error at the level of the product  $m \times G$  if and only if the  $i^{th}$  coordinate of  $m$  is not equal to zero.

The probability that the fault has a consequence on the product  $m \times G$  is approximatively of  $1/4$ . There is one possibility on two that the matrix position  $(i, j)$  to be zero (we suppose  $G$  random) and one possibility on two that the  $i^{th}$  coordinate of  $m$  is 1 (we suppose also that  $m$  is random).

Supposing that the fault is an incidence on the product  $m \times G$  (that we shall note  $m \times \tilde{G}$ ) the encoding of McEliece makes then the sum  $m \times \tilde{G}$  with an error  $e$  of fixed weight  $t$  (the used code corrects up to  $t$  errors).

If the error has its  $i^{th}$  coordinate equal to 1 (probability of  $\frac{t}{n}$ ) the fault has no effect and we do nothing more than decode  $t - 1$  errors.

**Remark :** when we try to decode with at  $t$  errors but if a word with less than  $t$  errors agrees, it should be chosen by the algorithm).

If the error has its  $i^{th}$  coordinate equal to 0 then the fault has an effect and we have to decode  $t + 1$  errors.

If the choice of  $t$  is such as  $t + 1$  do not give a unique word of code, we do not find the original message  $m$  but several possible candidates. To avoid this scenario, it is enough to choose  $t'$  in such way that addition of  $r$  errors (due to the faults) do not disturb the correction of errors ( $t' + r \leq t$ ).

- If the fault is made on the vector  $e$ . We have two cases : if the fault decreases the weight of  $e$ , the scheme will decrypt  $c$  correctly the  $t - 1$  leading errors (as the scheme can correct up to  $t$  errors) to retrieve the correct  $m$ . If the fault increases the weight of  $e$ , the scheme will decrypt  $c$  incorrectly as  $t + 1$  errors are leading.

With our fault model (a random fault on one bit) with restricted perimeter (fault on  $G$  or  $e$  but not on  $m$ ), the scheme decodes without error a faulty encryption in approximatively  $1/4$  of cases : if a fault occurs, it can be on 1 bit of the  $nk$  bits of  $G$  or on 1 bit of the  $n$  bits of  $e$ . On  $G$ , a flip 0 to 1 has an effect if  $m_i = 1$  and  $e_i = 0$ , a flip 1 to 0 has an effect if  $m_i = 1$  and  $e_i = 1$ . On  $e$ , a flip has an effect if  $e_i = 0$  (flip 0 to 1) but

no effect in the other case ( $e_i = 1$  and flip 1 to 0). Hence the probability that the scheme does not correct a fault is :

$$P(\text{error on } G)[P(G_{ij} = 0)P(m_i = 1)P(e_j = 0) + P(G_{ij} = 1)P(m_i = 1)P(e_j = 1)] + P(\text{error on } e)[P(e_j = 0)]$$

$$\frac{kn}{kn+n} \times \left( \frac{1}{2} \times \frac{1}{2} \times (1 - t/n) + \frac{1}{2} \times \frac{1}{2} \times (t/n) \right) + \frac{n}{kn+n} \times (1 - t/n) \approx 1/4.$$

**Remark :** If we consider the Niederreiter version where the encryption is a product  $H'm^T$  with  $H' = SHP$  (a masked Goppa code matrix),  $m$  the message of weight  $t$ ; the probability that a fault injection (in the matrix  $H'$ ) has an effect is  $t/n$  instead of the  $1/4$  in the McEliece case.

### 4.3 In the case of quasi-cyclic alternant codes version Niederreiter

In this case, we just store in memory the first row of the matrix and the vector and we shift this vector to obtain the matrix-vector product. So in this context, a fault injection will have an effect in several rows of the matrix-vector product. Let use the notation proposed in [1], the length of the code is  $N = q^m - 1 = l \times N_0$  and with one row we construct  $l$  rows (using circular permutation). The probability that a block with size  $l$  has no error position is  $\frac{\binom{N-l}{t}}{\binom{N}{t}}$ . So the probability that the fault has an effect is:  $1 - \frac{\binom{N-l}{t}}{\binom{N}{t}}$ . With the parameters proposed in §2.1 (see [1]), we obtain a probability greater than 0.97 for all the proposed parameters. In this case, the use of quasi-cyclic matrices presents a weakness regards to the fault injection.

## 5 Conclusion

The McEliece PKC owns good properties face to fault injection. Due to the fact that we have to decode to decrypt, the scheme corrects directly fault that we may introduce. We just have to choose parameters in order to counter this fault injection. Hence, not use the maximal capacity of the code is a good way to correct faults due to external attacks or mistakes in device's calculation.

In the case of quasi-cyclic matrices, the construction permits to implement the scheme on low resource device due to the small size of the public key, but the scheme is more sensitive to a fault injection. The use of compact matrices (quasi-cyclic) implies the use of the stored row several times and permits the diffusion of the fault that we may introduce.

For an embedded device, we have to find a compromise between the size of the public key and the security in regard to fault injection.

## References

1. T. Berger, P.-L. Cayrel, P. Gaborit and A. Otmani, *Reducing Key Length of the McEliece Cryptosystem*, AfricaCrypt 2009, LNCS, to appear.
2. D.J. Bernstein, J. Buchmann, E. Dahmen, *Post-Quantum Cryptography*, Springer, Berlin, 2009, ISBN 978-3-540-88701-0.
3. P.-L. Cayrel, P. Gaborit and E. Prouff, *Secure Implementation of the Stern Authentication and Signature Schemes for Low-Resource Devices*, Eighth Smart Card Research and Advanced Application Conference CARDIS 2008 In G. Grimaud and F.-X. Standaert, editors, Lecture Notes in Computer Science, Vol. 5189, pages 191-205, 2008
4. Y. X. Li, R. H. Deng and X.-M. Wang, *On the equivalence of McEliece's and Niederreiter's public-key cryptosystems*, IEEE Transactions on Information Theory, volume 40, number 1, 1994, pages 271-273
5. R. J. McEliece, *A Public-Key System Based on Algebraic Coding Theory*, Jet Propulsion Lab, DSN Progress Report 44, 1978, pages 114-116
6. H. Gregor Molter, Raphael Overbeck, Abdulhadi Shoufan, Falko Strenzke, and Erik Tews, *Side Channels in the McEliece PKC*, The Second international Workshop on Post-Quantum Cryptography PQCRYPTO 2008, Lecture Notes in Computer Science, Vol. 5299.
7. H. Niederreiter, *Knapsack-type cryptosystems and algebraic coding theory*, Problems Control Inform. Theory, Vol. 15, number 2, pages 159-166, 1986