McEliece/Niederreiter PKC: sensitivity to fault injection

PIERRE-LOUIS CAYREL
Center for Advanced Security Research Darmstadt,
CASED - Mornewegstrasse 32, 64293 Darmstadt, Germany
Email: pierre-louis.cayrel@cased.de

PIERRE DUSART
Université de Limoges, XLIM-DMI,
123 Av. Albert Thomas, 87060 Limoges Cedex, France.
Email: pierre.dusart@xlim.fr

***Abstract*—The McEliece and Niederreiter public key cryptosystems (PKC) are presumed secure in a post quantum world [4] because there is no efficient quantum algorithm that solves the hard problems upon which these cryptosystems are built. The present article indicates, however, a different type of vulnerability for such cryptosystems, namely fault injection. We present the injection fault in the McEliece scheme using Goppa codes and in two variants using quasi-cyclic alternant and quasi-dyadic codes, and describe the main difference of those constructions in this context.**

I. INTRODUCTION

In 1978, R. J. McEliece presented the first version of the cryptosystem which was to become the reference in coding-theory-based public key cryptography [14]. The original McEliece construction uses Goppa codes. However, many other families of codes have in the mean time been proven suitable for the McEliece system.

Against such a scheme, two classes of attacks are efficent: decoding attacks and structural attacks. A good tradeoff between error-correction capability and rate is necessary in order to make codes resistant to direct decoding attacks. Furthermore, a good McEliece-cryptosystem designer must also choose codes which are structurally secure, as such attacks are very efficient whenever applicable (see [18] for example).

The McEliece cryptosystem presents a good alternative to the 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 6,500 bits (see [3]) and 4,096 bits using quasi-dyadic Goppa codes [2]. These two constructions appear to be secure against decoding and structural attacks.

In the real world, it is also important to consider side channel attacks on real implementations and hardware. A first study of side channel attacks on the McEliece construction was done in [15]. We present here a fault injection study of those schemes, and show that the quasi-cyclic and the quasi-dyadic constructions present a weakness in this context. We focus our analysis on the encryption scheme (a direct area of application is satellite communication, which only involves encryption).

Our contribution

In their paper [15], the authors deal with side channel (timing) attacks against the McEliece scheme, but do not consider fault injection. In its classical form, the fault attack features an adversary who tries to find a secret key by analysing faulty outputs, which it obtains by inserting faulty inputs into the algorithm. More concretely, the adversary compares faulty ciphertexts obtained by encrypting different faulty plaintexts, or it compares a faulty ciphertext with an unfaulty one. Our approach is to study the sensitivity of the McEliece cryptosystem to fault injection. In particular, we consider an adversary whose goal is to produce a denial of service. In order to do so, it will try to provoke a fault which cannot be corrected by the error-correcting capacity of the code.

We focus on encryption (in this context, this is equivalent to encoding) and on the parameters necessary for the matrix design. Our intention is to show that code-based cryptosystems are intrinsically fault-resistant; by contrast, in public key RSA systems, injection of public parameters (modulus n and public exponent e) will cause a change in the resulting ciphertext if and only if the plaintext m is not identically 0 or 1. With probability very close to one, therefore, the attacker will be able to modify the ciphertext, thus causing an effective interruption in communication. We show that the resistance of McEliece cryptosystems to such an attack heavily depends on the underlying code.

In particular, the very efficient quasi-cyclic and quasi-dyadic codes are *not* secure against this kind of attack, while the less efficient, original McEliece scheme based on Goppa codes is.

Although a similar study of the decryption algorithm for code-based schemes is already in progress, this analysis is considered out of the scope of the present paper.

Organisation

This paper is constructed as follows. Section 2 presents as preliminaries the McEliece and Niederreiter PKC and their new constructions (using quasi-cyclic alternant codes and quasi-dyadic Goppa codes) which reduce the key size. We describe the parameters for each of the three underlying codes, including the classical Goppa code setting, in Section 3. Section 4 provides further details about fault attacks. Section 5 outlines the security of the McEliece PKC against this kind

978-1-4244-6949-9/10/$26.00 ©2010 IEEE

KeyGen(1^κ) (κ is the security parameter)

1) Choose n, k and t according to κ
2) Randomly pick a generator matrix $\boldsymbol{G}_0$ of an $[n, k, 2t+1]$ binary Goppa code $\mathscr{C}$
3) Randomly pick an $n \times n$ permutation matrix $\boldsymbol{P}$
4) Randomly pick a $k \times k$ invertible matrix $\boldsymbol{S}$
5) Compute $\boldsymbol{G} = \boldsymbol{S} \times \boldsymbol{G}_0 \times \boldsymbol{P}$
6) Output pk := $(\boldsymbol{G}, t)$ and sk := $(\boldsymbol{S}, \boldsymbol{G}_0, \boldsymbol{P}, \gamma)$, where γ is a t-bounded decoding algorithm of $\mathscr{C}$

Fig. 1. Key generation algorithm of the McEliece cryptosystem

of attacks. Section 6 discusses the security of the Niederreiter scheme in this context. Section 7 concludes the paper.

II. Preliminaries

A. The McEliece PKC

The McEliece PKC [14] represents one of the oldest public-key cryptosystem in use today. 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 k that is able to efficiently correct t errors. The chosen code must then be transformed into a random-looking 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 bases the construction on a binary Goppa code, and uses a generator matrix of this code, which is then randomised. The encryption function encodes a message into a codeword of the public-key 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 [17] also proposed a public-key cryptosystem based on linear codes in which the public key is a parity-check matrix. It is proved in [13] that these two systems are equivalent in terms of security.

The McEliece cryptosystem [14] uses error-correcting codes that have an efficient decoding algorithm in order to build trapdoor one-way functions.

The parameters of McEliece's original construction 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 $\mathscr{C}$ among the family of binary Goppa codes. The private key is the Goppa polynomial of $\mathscr{C}$. 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 this resulting matrix.

Figures 1 and 2 outline the key generation, encryption, and respectively decryption algorithms.

Encrypt(pk,$\boldsymbol{m} \in \mathbb{F}_2^k$)	Decrypt(sk,$\boldsymbol{c} \in \mathbb{F}_2^n$)
1) Randomly pick $\boldsymbol{e}$ in $\mathbb{F}_2$ of weight t	1) Compute $\boldsymbol{z} := \boldsymbol{c} \times \boldsymbol{P}^{-1}$
2) Compute $\boldsymbol{c} := \boldsymbol{m} \times \boldsymbol{G} + \boldsymbol{e}$	2) Compute $\boldsymbol{y} := \gamma(\boldsymbol{z})$
3) Output $\boldsymbol{c}$	3) Output $\boldsymbol{m} := \boldsymbol{y} \times \boldsymbol{S}^{-1}$

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

B. Constructions to reduce the key size

A possible solution to the problem of the huge key size of the McEliece scheme, is to take very sparse matrices. This idea has been explored in [16], where the authors examined the implications of using low density parity-check (LDPC) codes. The authors showed that taking sparse matrices for the linear transformations is an unsafe solution. A method originating in [9] tries to minimise key size by using quasi-cyclic codes [9], [1], [11], [10], [6]. This particular family of codes fortunately has a very simple and compact description. The first proposal [9] uses subcodes of a primitive BCH cyclic code. The size of the public key for this cryptosystem is only 12Kbits. Another approach attempts to combine the benefits of quasi-cyclicity and the sparse matrices of LDPC codes appears in [1] . The authors propose a public key size that is about 48Kbits. However, recent results [18] show that the cryptosystems [9], [1] can be completely broken.

Two other constructions (unbroken up to date) use quasi-cyclic alternant codes [3] and quasi-dyadic Goppa codes [2]. We focus on these two constructions.

Remark : In 2009, two new structural attacks have been proposed against quasi-cyclic alternant codes and quasi-dyadic Goppa codes : The first is a Sidelnikov-Shestakov type attack by Umana and Leander [12]; the second was proposed by Faugère, Otmani, Perret, and Tillich [8] and carries the same spirit as an earlier attack against quasi-cyclic BCH codes [18]. It seems that binary Goppa codes, and those codes alone, require an attack strategy that is different from the techniques published so far. Note that no structural attacks have so far been found against binary Goppa codes.

1) Quasi-cyclic alternant codes [3]: Let $q = p^d$ for some $d > 0$, and p a prime power. An alternant code $A(L, D)$ over $\mathbb{F}_p$ is defined by:

- a sequence $L \in (\mathbb{F}_q)^n$ of distinct elements with $n \leq p$;
- a sequence $D \in (\mathbb{F}_q)^n$ of nonzero elements;
- easily decodable ($t/2$ errors) syndromes from $H = T_p(\mathrm{vdm}_t(L) \cdot \mathrm{diag}(D))$.

A Goppa code $G(L, g)$ over $\mathbb{F}_p$ is an alternant code where:

- $L \in (\mathbb{F}_q)^n$ satisfies $g(L) \neq 0$, and $D = (1/g(L))$ for some monic polynomial $g(x) \in \mathbb{F}_q[x]$ of degree t;
- good error correction capability (all t design errors) in characteristic 2.

We can also see an alternant code as a subfield subcode of a Generalized Reed-Solomon code over a finite field. For a more extensive treatment of this topic, we recommend [3].

Another type of alternant codes exhibit a desirable property, namely quasi-cyclicity, and thus allow for the entire parity check matrix to be generated by its first row.

Let l be an integer value. A quasi-cyclic matrix H (in systematic form) is a matrix of the form:

$$H = (I|A) \ ,$$

where A is a *circulant matrix*, that is, a matrix of the form:

$$A = \begin{pmatrix} a_1 & a_2 & a_3 & \cdots & a_l \\ a_l & a_1 & a_2 & \cdots & a_{l-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_2 & a_3 & a_4 & \cdots & a_1 \end{pmatrix} \ ,$$

where $(a_1, a_2, a_3, \cdots, a_l)$ is any vector in $\mathbb{F}_2^l$.

Clearly, H can easily be generated from the vector $(a_1, a_2, a_3, \cdots, a_l)$ (which is the first row of A).

2) Quasi-dyadic Goppa codes [2]: In what follows, all vector and matrix indices are numbered from zero onwards.

Definition 1: Given a ring $\mathcal{R}$ and a vector $h = (h_0, \ldots, h_{n-1}) \in \mathcal{R}^n$, the *dyadic* matrix $\Delta(h) \in \mathcal{R}^{n \times n}$ is the symmetric matrix with components $\Delta_{ij} = h_{i \oplus j}$ where $\oplus$ stands for bitwise exclusive-or on the binary representations of the indices. The sequence h is called its *signature*. The set of dyadic $n \times n$ matrices over $\mathcal{R}$ is denoted $\Delta(\mathcal{R}^n)$. Given $t > 0$, $\Delta(t, h)$ denotes $\Delta(h)$ truncated to its first t rows.

One can recursively characterize a dyadic matrix when n is a power of 2: any 1×1 matrix is dyadic, and for $k > 0$, any $2^k \times 2^k$ dyadic matrix M has the form

$$M \ = \ \begin{bmatrix} A & B \\ B & A \end{bmatrix}$$

where A and B are $2^{k-1} \times 2^{k-1}$ dyadic matrices. It is not hard to see that the signature of a dyadic matrix coincides with its first row. Dyadic matrices form a commutative subring of $\mathcal{R}^{n \times n}$ as long as $\mathcal{R}$ is commutative.

Definition 2: A *dyadic permutation* is a dyadic matrix $\Pi^i \in \Delta(\{0,1\}^n)$ whose signature is the i-th row of the identity matrix.

A dyadic permutation is clearly an involution, i.e. $(\Pi^i)^2 = I$. The i-th row (or equivalently the i-th column) of the dyadic matrix defined by a signature h can be written $\Delta(h)_i = h\Pi^i$.

Definition 3: A *quasi-dyadic* matrix is a (possibly non-dyadic) block matrix whose component blocks are dyadic submatrices.

C. Niederreiter Cryptosystem

An encryption scheme that is a dual of the McEliece cryptosystem and equivalent in terms of security [13] is the Niederreiter construction [17]. The main difference between the McEliece and Niederreiter cryptosystems lies in the description of the codes. The Niederreiter encryption scheme describes codes by means of parity-check matrices. As before, however, this matrix forms part of the secret key, and is transformed by scrambling and permuting into the public key. The encryption algorithm takes as input words of weight t, where t is the number of errors that can be decoded. We denote by $\mathcal{W}_{q,n,t}$ the words of $\mathbb{F}_q^n$ of weight t.

Figures 3 and 4 outline the key generation, encryption, and respectively decryption algorithms.

Fig. 3. Key generation algorithm of the Niederreiter cryptosystem

KeyGen(1^κ)

1) Choose n, k, and t according to the security parameter κ
2) Randomly pick a $(n-k) \times n$ parity-check matrix $\boldsymbol{H}_0$ of an $[n, k, 2t+1]$ binary Goppa code $\mathscr{C}$
3) Randomly pick an $n \times n$ permutation matrix $\boldsymbol{P}$
4) Randomly pick an $(n-k) \times (n-k)$ invertible matrix $\boldsymbol{S}$
5) Compute $\boldsymbol{H} := \boldsymbol{S} \times \boldsymbol{H}_0 \times \boldsymbol{P}$
6) Output pk := $(\boldsymbol{H}, t)$ and sk := $(\boldsymbol{S}, \boldsymbol{H}_0, \boldsymbol{P}, \gamma)$, where γ is a t-bounded decoding algorithm of $\mathscr{C}$

Fig. 4. Encryption and decryption algorithms of the Niederreiter cryptosystem

Encrypt(pk, $\boldsymbol{m} \in \mathcal{W}_{2,n,t}$)	Decrypt(sk, $\boldsymbol{c} \in \mathbb{F}_2^{n-k}$)
1) Compute $\boldsymbol{c} := \boldsymbol{H} \times \boldsymbol{m}^T$	1) Compute $\boldsymbol{z} := \boldsymbol{S}^{-1} \times \boldsymbol{c}$
2) Output $\boldsymbol{c}$	2) Compute $\boldsymbol{y} := \gamma(\boldsymbol{z})$
	3) Output $\boldsymbol{m} := \boldsymbol{y} \times \boldsymbol{P}^{-1}$

III. Parameters

A. Generic construction using binary Goppa codes

A secure encryption construction (security level of 2^{80} binary operations) could be based on a $[512, 256, 128]$ binary Goppa code. Better security can be achieved by increasing the parameters as shown in Table 1.

TABLE I
Suggested parameters of Goppa codes for different security levels

security level	n	k	t	Public key size (bits)
80	512	256	128	460,647
112	640	384	128	1,047,600
128	768	512	128	1,537,536
192	1280	768	256	4,184,415

B. Quasi-cyclic alternant codes over $\mathbb{F}_{2^8}$

In [3], the authors present a new way to describe the public key matrix using a quasi-cyclic alternant code $\mathcal{C}$ of parameters $[459, 255, 101]_{2^8}$ for a public key of 8,100 bits and a security level of 2^{80} binary operations. For a public key of 13000 bits and a security level of 2^{100} binary operations, one should use a $[612, 408, 101]_{2^8}$ code. More values are given in Table II, from [3] which describes these parameters.

The authors of [3] proposed no parameters for binary quasi-cyclic alternant codes, but it seems that such codes are still secure against to structural attacks.

TABLE II
SUGGESTED PARAMETERS OF QUASI-CYCLIC ALTERNANT CODES FOR DIFFERENT SECURITY LEVELS

security level	n	k	t	Public key size (bits)
80	450	225	128	6,800
90	558	279	128	8,500
110	744	372	128	14,600

C. Quasi-dyadic binary Goppa codes

In [2], the authors present a new way to describe the public key matrix using quasi-dyadic Goppa code $\mathcal{C}$ of parameters $[512, 256, 128]$ for a public key of 4,096 bits and a security of 2^{80} binary operations and a code of parameters $[768, 512, 128]$ for a public key of 8,192 bits and a security of 2^{128} binary operations. The next table from [2] describes these parameters:

TABLE III
SUGGESTED PARAMETERS OF QUASI-DYADIC BINARY GOPPA CODES FOR DIFFERENT SECURITY LEVELS

security level	n	k	t	Public key size (bits)
80	512	256	128	4,096
112	640	384	128	6,144
128	768	512	128	8,192
192	1280	768	256	12,288

IV. FAULT INJECTION

Bernstein et al. have recently implemented [5] a code based cryptosystem (the Stern scheme) on a low resource device. Securing the implemented scheme against side channel attack is the next step in a real world context. The two constructions in [3], [2], which feature small descriptions of public keys, are also implementable on smart cards, therefore it is paramount to prove them secure.

A. Description of fault injection

Any system can make mistakes in computation. These errors could be caused internally (due to a poor quality of component, which generates fuzzy electric noise) or externally (specific areas of a circuit board are bombarded with heavy radiation, which disturbs the normal execution of a program). The external stress can be intentional (pirate's attack) or due to the specific environment in which the system is placed (e.g. space radiations for satellite). Many hardware systems are sensitive to this kind of attacks. In case of a pirate attack, the fault injection is the first step in a fault attacks. This fault can be exploited to find confidential data in secure devices or in order to compromise the system by creating a denial of service (DoS). In the case of an encryption device, a failure is randomly returned for some of the faulty calls. These failures may propagate to the system boundary and thus become observable by the attacker. He can exploit theses faults to find secret keys in the cryptographic device.

In case of malfunctions due to environment disturbances, the device should be fault tolerant: should function correctly (though possibly at reduced efficiency) in spite of the faults. Furthermore, in the case of for instance satellite components, it might be impossible to change or to fix a faulty device. 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, fault injection must be also taken into account.

In the following, we study whether fault occurrences on the public key of a coding based scheme could alter the device's capacity to communicate (this information is saved on the internal components of the system). The goal of this attacker is to cause a denial of service, which would render communication useless as decoding becomes incorrect.

B. Fault model

We consider a single one-bit fault on the used variables, corresponding to a flip on a memory cell. We study for each construction the effect of a fault on one variable of the scheme. For all schemes, if the input message m is affected by the fault and transformed to m', encryption will proceed as usual. The decryption operation will thus compute m', and not the good value m. Hence an error on the input message will be not corrected by the scheme. Hence, for the next sections, we say that a fault on m is out of the scope.

C. Countermeasures

Errors in encryption systems are classically corrected by inversion: if a message is encrypted, then decrypted correctly, the encryption was faultless. Though a good solution, this method is inapplicable in some cases, such as public key encryption systems – where the device knows the public key only – or when decryption is too inefficient. In this case, the alternative is to have dedicated hardware (in fact two identical devices) which computes the encryption twice.

Countermeasures to other side channel attacks against the McEliece scheme have been presented in [15].

V. FAULT INJECTION ON THE MCELIECE ENCRYPTION SCHEME

We describe in this section the sensitivity of the McEliece PKC against fault injection.

A. Fault on variables of McEliece Encryption

The McEliece encryption works in the following way: each plaintext vector m is multiplied by the encryption matrix G, and an error e is added to the result. Thus,

$$\boldsymbol{c} := \boldsymbol{m} \times \boldsymbol{G} + \boldsymbol{e}.$$

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

1) If m is corrupted, the scheme can't correct this entry.
2) If the fault affects entry g_{ij} on row i and column j of G, the output of $m \times G$ is faulty if and only if the i-th coordinate of m is not zero.
   We assume the entries of G to be random, therefore there $g_{ij} = 0$ with probability $1/2$, while the i-th entry of m

is 0 w.p. $1/2$. Therefore, the probability that a fault in coordinate g_{ij} will give the wrong output is $1/4$.

Supposing that the fault has an effect on the product $\boldsymbol{m} \times \boldsymbol{G}$ (which we denote for clarity by $\boldsymbol{m} \times \widetilde{\boldsymbol{G}}$) we analyse the consequences of adding an error vector e of fixed weight t (the error correction capability is t) to this product.

- If the error has its i^{th} coordinate equal to 1 (probability of $\frac{t}{n}$) the fault has no effect and we correct $t-1$ errors.
- If the error has its i^{th} coordinate equal to 0, then the fault carries over and we have to correct $t+1$ errors.

If the choice of t is such that correcting $t+1$ errors does not gives a unique codeword, we don't find the original message m, but several possible candidates. This scenario can be prevented if one chooses an error of weight t' such that the presence of up to a certain number r of faults does not affect the decoding efficiency (choose t' and r s.t. $t'+r \leq t$).

3) If the fault appears in the vector e, we have two cases:
   - If the fault decreases the weight of e, the scheme will correctly decrypt the $t-1$ remaining errors and will retrieve the correct m (as the scheme can correct up to t errors).
   - If the fault increases the weight of e, the scheme will decrypt c incorrectly, as $t+1$ errors remain.

With our fault model (a random fault on one bit) with restricted perimeter (fault in G or e, but not on m), the scheme effectively decodes the ciphertext in approximately $1/4$ of the 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 $e_i = 0$, but no effect in the other case ($e_i = 1$).

Hence, the probability that the scheme does not correct a fault is as follows:

$$\begin{aligned}
P_1 &= P(\text{error is effective}) \\
&= P(\text{error is effective} \mid \text{error is on } G)P(\text{error is on } G) \\
&\quad + P(\text{error is effective} \mid \text{error is on } e) \\
&= P(\text{error on } G)[P(G_{ij}: 0 \to 1)P(m_i = 1)P(e_j = 0) \\
&\quad + P(G_{ij}: 1 \to 0)P(m_i = 1)P(e_j = 1)] \\
&\quad + P(\text{error on } e)[P(e_j: 0 \to 1)] \\
&= \frac{kn}{kn+n} \times \left(\frac{1}{2} \times \frac{1}{2} \times (1 - t/n) + \frac{1}{2} \times \frac{1}{2} \times (t/n) \right) \\
&\quad + \frac{n}{kn+n} \times (1 - t/n) \approx 1/4.
\end{aligned}$$

B. Constructions with reduced codes

The advantage of this construction lies in the reduced public-key size. The size $SzPK$ of the public-key (G, t) essentially originates in the description of G; we occasionally refer to a fault in G by the general term public-key fault. We suppose that this error was spread across l rows. If the row i is affected, the fault carries over to the product $m \times G$ only if m_i=1, and the final result is corrupted if the fault appears in bit j of the error e. In the first case, there is no consequence if the l errors are spread over the positions where $m_i = 0$. If we suppose that m has weight w, the probability that the plaintext has exactly k errors is $\frac{\binom{n-w}{l-k}\binom{w}{k}}{\binom{n}{l}}$, as we choose k lines where $m_i = 1$. In the second case, the errors have no consequence if the flip is not correlated to the corresponding element e_j. As the weight of e is t, we have a probability of $1 - \frac{\binom{t}{k}}{\binom{n}{k}}$ that G_{ij} equals $e_j = 0$ and a probability of $1 - \frac{\binom{n-t}{k}}{\binom{n}{k}}$ that G_{ij} equals $e_j = 1$.

Hence the probability that the scheme does not correct a fault is :

$$\begin{aligned}
P_2 &= \frac{SzPK}{SzPK+n} \sum_{k=1}^{l} \left(\frac{\binom{n-w}{l-k}\binom{w}{k}}{\binom{n}{l}} \right) \left(1 - \frac{\binom{t}{k}+\binom{n-t}{k}}{2\binom{n}{k}} \right) \\
&\quad + \frac{n}{SzPK+n} \times (1 - t/n).
\end{aligned}$$

As m is random, we have $w \approx n/2$. Hence,

$$\begin{aligned}
P_2 &\approx \frac{SzPK}{SzPK+n} \sum_{k=1}^{l} \left(\frac{\binom{n/2}{l-k}\binom{n/2}{k}}{\binom{n}{l}} \right) \left(1 - \frac{\binom{t}{k}+\binom{n-t}{k}}{2\binom{n}{k}} \right) \\
&\quad + \frac{n}{SzPK+n} \times (1 - t/n).
\end{aligned}$$

As $kn > SzPK$ and generally $l > 1$, $P_2 > P_1$ and the classical construction with Goppa codes is more secure.

For example, with the parameters $l = 8, n = 512, SzPk = 4096$ and $t = 128$, we find $P_2 = 0.81$.

VI. Fault Injection on the Niederreiter Encryption scheme

A. Construction on Goppa codes

Here, the fault can only be on H. If we consider the Niederreiter version, where the encryption is a product $H'm^T$ with $H' = SH_0P$ (a masked Goppa code parity check matrix) and m is a plaintext of weight t; the probability that a fault injection (in the matrix H') has an effect is t/n instead of $1/4$ as in the McEliece case.

B. Constructions with quasi-cyclic alternant codes

In this case, we simply store the first row of the matrix H, H' and the vector m, and we shift this vector to obtain the matrix-vector product. So in this context, a fault injection will have an effect on several rows of the matrix-vector product.

In the notation of [3], the length of the code is $N := q^m - 1 = l \times N_0$ and we reconstruct the l rows of the matrix from the one we stored (by using a circular permutation).

The probability that a block of size l contains no error is $\frac{\binom{N-l}{t}}{\binom{N}{t}}$.

Therefore, the probability that the fault has an effect is: $1 - \frac{\binom{N-l}{t}}{\binom{N}{t}}$.

With the parameters detailled in Section 2, we obtain a probability of more than 0.97 that fault injection is effective. This result holds for all sets of proposed parameters.

The use of quasi-cyclic matrices presents a weakness with regard to the fault injection.

C. Construction with quasi-dyadic binary Goppa codes

As in the quasi-cyclic case, we just store a row of the parity check matrix in order to construct the whole matrix, thus the injected fault will affect several rows. A fault on a block h_k of the matrix signature results in one fault per line (on line i, the column $j = i \oplus k$ is affected).

The computation of the probability of an effect of a fault injection is the same as described above with the parameters detailled in Section 2, we obtain a probability greater than 0.97 for all the proposed parameters.

Therefore, Niederreiter constructions on quasi-dyadic Goppa codes are vulnerable to fault injection attacks.

VII. Conclusion

The McEliece PKC is resistant to fault injections due to the underlying code's capacity to correct faults that might occur. We just have to choose parameters that are large enough in order to counter this fault injection. Choosing a greater error correcting capacity than strictly necessary allows for the correction of faults due to external or internal stimuli.

In the cases of quasi-cyclic and quasi-dyadic matrices, the construction is implementable on low resource devices due to the small public key size, but the scheme is more sensitive to fault injections. The use of compact matrices, generated by a single stored row, implicitly leads to the diffusion of the fault, as this faulty row is reused several times.

It's now possible[19] to implement McEliece cryptosystem on a smartcard. For an embedded device, we have to find a compromise between the size of the public key and the security with respect to fault injection.

Though considered outside the scope of this paper, the decryption algorithms in the McEliece and Niederreiter constructions are also vulnerable to fault injections. An analysis of this topic is already in progress.

References

[1] M. Baldi and G. F. Chiaraluce, *Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes*, IEEE International Symposium on Information Theory, 2007, Nice, France, pages 2591–2595

[2] P. Barreto and R. Misoczki, *Compact McEliece Keys from Goppa Codes*, Accepted for SAC'2009, http://eprint.iacr.org/2009/187

[3] T. Berger, P.-L. Cayrel, P. Gaborit and A. Otmani, *Reducing Key Length of the McEliece Cryptosystem*, AfricaCrypt 2009, LNCS, page 77–97

[4] D.J. Bernstein, J. Buchmann and E. Dahmen, *Post-Quantum Cryptography*, Springer, Berlin, 2009, ISBN 978-3-540-88701-0.

[5] 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

[6] P.-L. Cayrel, A. Otmani and D. Vergnaud, *On Kabatianskii-Krouk-Smeets Signatures*, Proceedings of the first International Workshop on the Arithmetic of Finite Fields (WAIFI 2007), Springer Verlag Lecture Notes, Madrid, Spain, pages 237–251

[7] P. Dusart, G. Letourneux and O. Vivolo, *Differential Fault Analysis on A.E.S* , Lecture Notes in Computer Science Volume 2846/2003, Applied Cryptography and Network Security, 2003, pp. 293–306

[8] J.-C. Faugère, A. Otmani, L. Perret and J.-P. Tillich, *Algebraic Cryptanalysis of McEliece Variants with Compact Keys*, Eurocrypt 2010.

[9] P. Gaborit, *Shorter keys for code based cryptography* Proceedings of the 2005 International Workshop on Coding and Cryptography (WCC 2005), Bergen, Norway pages 81–91

[10] P. Gaborit and M. Girault, *Lightweight code-based authentication and signature*, IEEE International Symposium on Information Theory (ISIT 2007), Nice, France pages 191–195

[11] P. Gaborit, C. Lauradoux and N. Sendrier, *SYND: a Fast Code-Based Stream Cipher with a Security Reduction*, IEEE International Symposium on Information Theory (ISIT 2007), pages 186–190

[12] V. Gauthier Umana and G. Leander, *Practical Key Recovery Attacks On Two McEliece Variants*, 2009, Preprint, `http://eprint.iacr.org/2009/509.pdf`

[13] 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

[14] R. J. McEliece, *A Public-Key System Based on Algebraic Coding Theory*, Jet Propulsion Lab, DSN Progress Report 44, 1978, pages 114-116

[15] H. G. Molter, R. Overbeck, A. Shoufan, F. Strenzke and E. Tews, *Side Channels in the McEliece PKC*, The Second international Workshop on Post-Quantum Cryptography PQCRYPTO 2008, Lecture Notes in Computer Science, Vol. 5299.

[16] C. Monico, J. Rosenthal and A. Shokrollahi, *Using low density parity check codes in the McEliece cryptosystem*, IEEE International Symposium on Information Theory (ISIT 2000), 2000, Sorrento, Italy, 215

[17] H. Niederreiter, *Knapsack-type cryptosystems and algebraic coding theory*, Problems Control Inform. Theory, Vol. 15, number 2, pages 159-166, 1986

[18] A. Otmani, J.P. Tillich and L. Dallot, *Cryptanalysis of McEliece Cryptosystem Based on Quasi-Cyclic LDPC Codes*, in Proceedings of First International Conference on Symbolic Computation and Cryptography, Beijing, China, LMIB Beihang University, April 28-30 2008, p. 69-81

[19] F. Strenzke, *A Smart Card Implementation of the McEliece PKC*, WISTP'10.

We are grateful to Cristina Onete for her useful rewriting comments.