Timing Attacks against the Syndrome Inversion in Code-based Cryptosystems

Falko Strenzke[1]

[1] ** fstrenzke@crypto-source.de
[2] Cryptography and Computeralgebra, Department of Computer Science, Technische Universität Darmstadt, Germany

Abstract. In this work we present new timing vulnerabilities that arise in the inversion of the error syndrome through the Extended Euclidean Algorithm that is part of the decryption operation of code-based Cryptosystems. We analyze three types of timing attack vulnerabilities theoretically and experimentally: The first allows recovery of the zero-element of the secret support, the second is a refinement of a previously described vulnerability yielding linear equations about the secret support, and the third enables to retrieve non-linear equations about the secret support. Furthermore, we analyze theoretically the limitations applying to actual attacks based on the information gained in such manner.

Keywords: side channel attack, timing attack, post quantum cryptography, code-based cryptography

1 Introduction

The McEliece PKC [1] and Niederreiter [2] Cryptosystems, build on error correcting codes, are considered imune to quantum computer attacks [3], and thus are of interest as candidates for future cryptosystems in high security applications. Accordingly, they have received growing interested from researchers in the past years and been analyzed with respect to efficiency on various platforms [4–8]. Furthermore, a growing number of works has investigated the side channel security of code-based cryptosystems [9–13].

Side channel security is a very important implementation aspect of any cryptographic algorithm. A side channel is given when a physical observable quantity that is measured during the operation of a cryptographic device, allows an attacker to gain information about a secret that is involved in the cryptographic operation. The usual observables used in this respect are the duration of the operation (timing attacks [14]), or the power consumption as a function over the time (power analysis attacks[15]).

So far, timing attacks against the decryption operation of the McEliece PKC targeting the plaintext have been developed [12, 10]. In [11], a timing attack is

** To the most part, this work was done in the author's private capacity, a part of the work was done at[2] described that targets the secret support that is part of the private key in code-based cryptosystems. The attacker learns linear equations about the support in this attack.

This work extends on the analysis given in [11] in multiple ways: first of all, we find that a control flow ambiguity causing leakage in terms of the linear equations is manifest already in the syndrome inversion preceding the solving of the key equations in the decryption operation, and consequently the countermeasure proposed in that work is insufficient. We furthermore give experimental results showing that the attack in terms of collecting these linear equations through timing analysis is practical. We also show that there exists a timing side channel vulnerability in the syndrome inversion that allows the attacker to gain knowledge of the zero-element of the secret support. Again, after a theoretical analysis, we demonstrate that the vulnerability can be exploited practically. As an extension resp. generalization of the attacks yielding linear equations, we derive a practical timing attack that lets the attacker gain non-linear equations. Lastly, we give a theoretical analysis of the limitations applying to the usability of the linear equation system for actual attacks and discuss in how far the employment of the non-linear equations yields improvements.

2 Preliminaries

In this work, we give a brief description of the McEliece PKC, and stress those features of the decryption algorithm, that are necessary to understand the timing attack presented in this paper. A more detailed description and security considerations can be found e.g. in [16].

Goppa Codes. Goppa codes [17] are a class of linear error correcting codes. The McEliece PKC makes use of irreducible binary Goppa codes, so we will restrict ourselves to this subclass.

Definition 1. *Let the polynomial* $g(Y) = \sum_{i=0}^{t} g_i Y^i \in \mathbb{F}_{2^m}[Y]$ *be monic and irreducible over* $\mathbb{F}_{2^m}[Y]$*, and let* m*,* t *be positive integers. Then* $g(Y)$ *is called a* Goppa polynomial *(for an irreducible binary Goppa code).*

Then an irreducible binary Goppa code is defined as $\Gamma(g(Y)) = \{\boldsymbol{c} \in \mathbb{F}_2^n | S_{\boldsymbol{c}}(Y) := \sum_{i=0}^{n-1} \frac{c_i}{Y-\gamma_i} = 0 \bmod g(Y)\}$*, where* $n = 2^m$*,* $S_{\boldsymbol{c}}(Y)$ *is the syndrome of* $\boldsymbol{c}$*, the* γ_i*,* $i = 0, \ldots, n-1$ *are pairwise distinct elements of* $\mathbb{F}_{2^m}$*, and* c_i *are the entries of the vector* $\boldsymbol{c}$*.*

The code defined in such way has length n, dimension $k = n - mt$ and can correct up to t errors. Note that in general, there is a freedom in choosing the ordering of the elements γ_i, $i = 0, \ldots, n-1$. Each such ordering will yield a different code. In this work, we choose lexicographical ordering for the γ_i.

As for any error correcting code, for a Goppa code there exists a generator matrix G and a parity check matrix H [18]. Given these matrices, a message $\boldsymbol{m}$ can be encoded into a codeword $\boldsymbol{c}$ of the code by computing $\boldsymbol{z} = \boldsymbol{m}G$, and the syndrome of a (potentially distorted) codeword can be computed as $\boldsymbol{s} = \boldsymbol{z}H$.

Here, we do not give the formulas for the computation of these matrices as they are of no importance for the understanding of the attack developed in this work. The interested reader, however, is referred to [18].

Overview of the McEliece PKC. In this section we give a brief overview of the McEliece PKC, where we closely follow the one given in [19].

The McEliece *secret key* consists of the Goppa polynomial $g(Y)$ of degree t defining the secret code Γ, an $n \times n$ permutation matrix P and a non-singular $k \times k$ matrix S over $\mathbb{F}_2$. The *public key* is given by the public $n \times k$ generator matrix $G_p = SG_sP$ over $\mathbb{F}_2$, where G_s is a generator matrix of the secret code Γ. The *encryption* operation allows messages $\boldsymbol{m} \in \mathbb{F}_2^k$. A random vector $\boldsymbol{e} \in \mathbb{F}_2^n$ with hamming weight $\mathrm{wt}(\boldsymbol{e}) = t$ has to be created. Then the ciphertext is computed as $\boldsymbol{z} = \boldsymbol{m}G_p + \boldsymbol{e}$.

McEliece *decryption* is performed in the following way: First, compute $\boldsymbol{z}' = \boldsymbol{z}P^{-1} = \boldsymbol{m}SG_s + \boldsymbol{e}P^{-1}$. The right hand side of this equation demonstrates that applying error correction using the secret code Γ will recover the permuted error vector $\boldsymbol{e}' = \boldsymbol{e}P^{-1}$, since $\boldsymbol{m}SG_s$ is a code word of Γ. The error correction procedure is detailed in Section 2.

If the matrix S is chosen in such way that the public generator matrix is in reduced row echelon form, i.e. $G_p = [\mathbb{I}|G_2]$, then, in the decryption operation, $\boldsymbol{m}$ can be recovered by extracting the first k bits of $\boldsymbol{m}SG_s$. This would be a security problem if the McEliece PKC was used as proposed in [1]. But since the system has been proven to be insecure against adaptive chosen ciphertext attacks, a so called CCA2-Conversion [20, 21] has to be applied in any case. Consequently, using the reduced row echelon form is not a problem [22]. In this case, the matrix S does not have to be part of the private key. This choice for the matrix S will be assumed for the remainder of the paper. In this work however, we describe the McEliece PKC without such a CCA2-Conversion, as this is of no relevance for the side channel attack devised in this work.

Appropriate choices for the security parameters of the scheme, n and t, would e.g. be $n = 2048$ and $t = 50$ for about 100 bit security [19].

The Decryption Operation in the McEliece PKC. As mentioned above, the first step in the decryption operation is computing $\boldsymbol{z}' = \boldsymbol{z}P^{-1}$. Then, the syndrome associated with this ciphertext has to be computed. This can be done using a parity check matrix H corresponding to the secret code Γ. Algorithm 1 depicts the top level McEliece decryption. It makes use of the error correction algorithm, given by the Patterson Algorithm [23], shown in Algorithm 3. Please note that in Step 1 of this algorithm, the multiplication with the coefficient vector is used to turn the syndrome vector into the syndrome polynomial $S(Y)$. The Patterson Algorithm, in turn, uses an algorithm for finding roots in polynomials over $\mathbb{F}_{2^m}$ (root_find()), and the Extended Euclidean Algorithm (XGCD) for polynomials with a break condition based on the degree of the remainder, given in Algorithm 4. The root finding can e.g. be implemented as an exhaustive search on $\mathbb{F}_{2^m}$. Please note that all polynomials appearing in the algorithms have coeffients in $\mathbb{F}_{2^m}$.

In the following, we turn to those details, that are relevant for the side channel issues we are going to address in Section 3. Please note that the error locator polynomial $\sigma(Y)$, which is determined in Step 4 of Algorithm 3, has the following form[3]:

$$\sigma(Y) = \sigma_t \prod_{j \in \mathcal{E}'} (Y - \gamma_j) = \sum_{i=0}^{t} \sigma_i Y^i. \tag{1}$$

where $\mathcal{E}'$ is the set of those indexes i, for which $e'_i = 1$, i.e. those elements of $\mathbb{F}_{2^m}$ that correspond to the error positions in the permuted error vector. The determination of the error vector in Step 6 of Algorithm 3 makes use of this property. Accordingly, $\deg(\sigma(Y)) = \mathrm{wt}(\boldsymbol{e})$ if $\mathrm{wt}(\boldsymbol{e}) \leqslant t$ holds.

Algorithm 2 The McEliece Decryption Operation

```
Require: the McEliece ciphertext $z$
Ensure: the message $m$
1: $\boldsymbol{z}' \leftarrow \boldsymbol{z}P^{-1}$
2: $\boldsymbol{e}' \leftarrow \mathrm{err\_corr}(\boldsymbol{z}', g(Y))$
3: $\boldsymbol{e} \leftarrow \boldsymbol{e}'P$
4: $\boldsymbol{m}' \leftarrow \boldsymbol{z} + \boldsymbol{e}$
5: $\boldsymbol{m} \leftarrow$the first $k$ bits of $\boldsymbol{m}'$
6: return $\boldsymbol{m}$
```

Algorithm 3 The McEliece error correction with the Patterson Algorithm ($\mathrm{err\_corr}(\boldsymbol{z}', g(Y))$)

```
Require: the permuted ciphertext $\boldsymbol{z}'$, the secret Goppa polynomial $g(Y)$
Ensure: the permuted error vector $\boldsymbol{e}'$
1: $S(Y) \leftarrow \boldsymbol{z}'H^\top \left(Y^{t-1}, \cdots, Y, 1\right)^\top$
2: $\tau(Y) \leftarrow \sqrt{S^{-1}(Y) + Y} \bmod g(Y)$
3: $(\alpha(Y), \beta(Y)) \leftarrow \mathrm{XGCD}\left(\tau(Y), g(Y)\right)$
4: $\sigma(Y) \leftarrow \alpha^2(Y) + Y\beta^2(Y)$
5: $\mathcal{E}' = \{E_0, \ldots, E_{t-1}\} \leftarrow \mathrm{rootfind}(\sigma(Y))$ // if $\gamma_i$ is a root, then $\mathcal{E}'$ contains $i$
6: $\boldsymbol{e}' \leftarrow \boldsymbol{v} \in \mathbb{F}_2^n$ with $v_i = 1$ if and only if $i \in \mathcal{E}'$
7: return $\boldsymbol{e}'$
```

[3] Usually, the error locator polynomial is defined to be monic, i.e $\sigma_t = 1$. But as a matter of fact the error locator polynomial generated in Step 4 of Algorithm 3 generally is not monic.

3 Analysis of Timing Side Channels in the Syndrome Inversion

3.1 Properties of the Syndrome Inversion

Algorithm 4 The Extended Euclidean Algorithm (EEA($r_{-1}(Y)$, $r_0(Y)$, d))

```
Require: the polynomials $r_{-1}(Y)$ and $r_0(Y)$, with $\deg(r_0(Y)) < \deg(r_{-1}(Y))$
Ensure: two polynomials $r_N(Y)$, $b_N(Y)$ satisfying $r_N(Y) = b_N(Y) r_0(Y) \bmod r_{-1}(Y)$
    and $\deg(r_0(Y)) \leqslant \lfloor \deg(r_{-1})/2 \rfloor$
1: $b_{-1} \leftarrow 0$
2: $b_0 \leftarrow 1$
3: $i \leftarrow 0$
4: while $\deg(r_i(Y)) > d$ do
5:     $i \leftarrow i + 1$
6:     $(q_i(Y), r_i(Y)) \leftarrow r_{i-2}(Y)/r_{i-1}(Y)$ // polynomial division with quotient $q_i$ and
       remainder $r_i$
7:     $b_i(Y) \leftarrow b_{i-2}(Y) + q_i(Y) b_{i-1}(Y)$
8: end while
9: $N \leftarrow i$
10: return $(r_N(Y), b_N(Y))$
```

The syndrome polynomial is defined as

$$S(Y) \equiv \sum_{i=1}^{w} \frac{1}{Y \oplus \epsilon_i} \equiv \frac{\Omega(Y)}{\sigma(Y)} \bmod g(Y) \tag{2}$$

The identification of the error locator polynomial $\sigma(Y)$ in the denominator is simply a result of the form of the common denominator of all sum terms. Based on this form, it is possible to find the $\sigma(Y)$ in case of $w \leqslant t/2$ by invoking the Alg. 4 as EEA$(g(Y), S(Y), (t/2)-1)$, i.e. with $r_{-1}(Y) = g(Y)$ and $r_0(Y) = S(Y)$ and breaking once $\deg(r_i(Y)) \leqslant (t/2) - 1$. Then, it returns $\delta\sigma(Y) = b_N(Y)$ [18].

Given this form of the $S(Y)$, we can make a statement about the maximal possible number of iterations in the EEA used to compute $S^{-1}(Y) \equiv \sigma(Y)/\Omega(Y) \bmod g(Y)$ with $\Omega(Y)$ and $\sigma(Y)$ given through $S(Y) \equiv \Omega(Y)/\sigma(Y) \bmod g(Y)$. To this end, note that the actual invocation for the syndrome inversion is EEA$(g(Y), S(Y), 1)$, since in code based cryptosystems based on Goppa Codes we use Pattersons's Algorithm which can correct up to t errors. But the above explained fact that we could stop at $\deg(r_i(Y)) \leqslant (t/2) - 1$ means that we there is one iteration in the EEA where $r_i(Y) = \delta\Omega(Y)$ and $b_i(Y) = \delta\sigma(Y)$, in case of $w \leqslant (t/2) - 1$.

Corollary 1. *Assume a Goppa Code defined by $g(Y)$ and some arbitrary support Γ. When Alg. 4 is invoked as* EEA$(g(Y), S(Y), 1)$ *with $S(Y) = \frac{\Omega(Y)}{\sigma(Y)}$, and the* *error vector* $\boldsymbol{e}$ *corresponding to* $S(Y)$ *satisfies* $\mathrm{wt}(\boldsymbol{e}) \leqslant (\deg(g(Y))/2) - 1$*, then for the number of iterations in Alg. 4 we find:*

$$M \leqslant M_{\max} = \deg(\Omega(Y)) + \deg(\sigma(Y))$$

Proof. Regard the iteration where $r_j(Y) = \delta\Omega(Y)$ and $b_j(Y) = \delta\sigma(Y)$. Since according to Alg. 4 the degree of $b_j(Y)$, starting from zero, increases at least by one in each iteration, we find $j \leqslant \deg(\sigma(Y))$. From here on, the degree of $r_j(Y) = \delta\Omega(Y)$ is decreased by at least one in each subsequent iteration down to $\deg(r_M(Y)) = 0$, i.e. $M - j \leqslant \deg(\Omega(Y))$, giving $M = M - j + j \leqslant \deg(\Omega(Y)) + \deg(\sigma(Y))$. □

3.2 Leakage of the Zero Element of the Support

Algorithm 5 Polynomial Division poly_div$(n(Y), d(Y))$

```
Require: the polynomials $n(Y), d(Y))$ with $\deg(n(Y)) \geqslant \deg(d(Y))$
Ensure: two polynomials $s(Y)$, $q(Y)$ with $q(Y)d(Y) + s(Y) = n(Y)$ and $\deg(s(Y)) < \deg(d(Y))$
 1: $s_{-1}(Y) \leftarrow n(Y)$
 2: $s_0(Y) \leftarrow d(Y)$
 3: $q_0(Y) \leftarrow 0$
 4: $i \leftarrow 0$
 5: while $\deg(s_i(Y)) \geqslant \deg(d(Y))$ do
 6:     $i \leftarrow i + 1$
 7:     $a_i \leftarrow s_{i-2,\deg(s_{i-2}(Y))} / s_{i-1,\deg(s_{i-1}(Y))}$
 8:     $f_i \leftarrow \deg(s_{i-2}(Y)) - \deg(s_{i-1}(Y))$
 9:     $q_i(Y) = q_{i-1} + a_i Y^{f_i}$
10:     $s_i \leftarrow s_{i-2}(Y) - a_i s_{r-1}(Y) Y^{f_i}$
11: end while
12: return $(q_i(Y), s_i(Y))$
```

For $w = 1$ the whole control flow in Patterson's Algorithm is very simple and unambigious on a high level:

$$S(Y) \equiv \frac{1}{Y \oplus \epsilon_1} \bmod g(Y),$$

$$S^{-1}(Y) = Y \oplus \epsilon_1,$$

$$\tau(Y) = \sqrt{\epsilon_1}$$

$$a(Y) = \tau(Y)$$

$$b(Y) = 1$$

$$\sigma(Y) = Y \oplus \epsilon_1$$

The polynomial inversion is, according to Cor. 1, performed in exactly one iteration. But there is an ambigous control flow within the polynomial division (Alg. 5) that is executed within this EEA iteration: We find $q_1(Y) = Y$ because there is no alternative to $\deg(S(Y)) = t - 1$.

If $\epsilon_1 = 0$, then the division has to stop at this point. Otherwise, a second iteration is performed giving $q_2(Y) = \delta(Y \oplus \epsilon_1)$. Thus, if the timing difference resulting from the different number of iterations in the division is detectable, the secret support element α_0 can be found.

This explicit connection between the value of σ_0 and the number of iterations in a division is only given when a single iteration is performed in the Syndrome Inversion EEA. But information about $\alpha_0 = 0$, i.e. the position of the zero element of the secret support, can be obtained in other cases as well.

For $w = 2$ we find $\delta\sigma_0 = 1 + q_{2,0}q_{1,0} = \delta\epsilon_1\epsilon_2$. Thus, if w.l.o.g $\epsilon_1 = 0$, then $\sigma_0 \neq 0$ and consequently neither $q_{2,0}$ nor $q_{1,0}$, both being linked to the number of division iterations in the above discussed manner, can be zero. If an attacker finds from the running time that both divisions are executed in two iterations, then he knows $\epsilon_1 \neq 0 \wedge \epsilon_2 \neq 0$. These inequations certainly are less useful than the explicit information obtainable for $w = 1$, but still leakage is present.

The same type of inequations can be found for certain special cases of $w = 3$: Assume the syndrome inversion is performed with one less iteration and that this happens before the j-iteration where $r_j(Y) = \delta\Omega(Y)$, i.e. $j = 2$ in this case. We leave it open as to whether it is possible to detect this case through timing side channels alone. But given this case is detectable, further refinement in the side channel analysis might reveal wether $q_{1,0} = 0$ or $q_{2,0} = 0$. In either of these two cases, $\delta\sigma_0 = \delta\epsilon_1\epsilon_2\epsilon_3 = q_{2,0}q_{1,0} + 1 \neq 0$.

3.3 The Case $w = 4$

In the following, we give an analysis of the control flow of the first application of EEA during the McEliece decryption, which is used to invert the syndrome polynomial in $\mathbb{F}_{2^m}[X]/g(Y)$. By "control flow analysis", we basically mean giving statements about the number of iterations that are performed in the EEA invocations that an attacker can use to gain information about the secret key, specifically the code support.

Control Flow Analysis of Syndrome Inversion The vulnerability we aim at is given if ciphertexts with even error weights $w \leqslant (t/2) - 1$. The reason that the vulnerability is not exploitable for odd values of w will be given in the course of the analysis.

The error weight $w = 4$ is the lowest value of the w were the exploitable control flow vulnerability occurs. Thus we will start with the analysis of this concrete instance and then give arguments as to why the same control flow vulnerabilities occur for higher even value of w.

In the case of $w = 4$ the syndrome polynomial is of the form:

$$S(Y) \equiv \sum_{i=1}^{4} \frac{1}{Y \oplus \epsilon_i} \equiv \frac{aY^2 \oplus b}{Y^4 \oplus aY^3 \oplus cY^2 \oplus bY \oplus d} \bmod g(Y), \tag{3}$$

where $\epsilon_i \in \mathbb{F}_{2^m}, i \in 1, \dots, 4$ denote the the four elements of the support associated with the error positions. Furthermore, in the right hand side of Equ. 3, which is found by bringing all four sum terms to their common denominator, we have

$$\begin{aligned} a &= \epsilon_1 \oplus \epsilon_2 \oplus \epsilon_3 \oplus \epsilon_4, \\ b &= \epsilon_2\epsilon_3\epsilon_4 \oplus \epsilon_1\epsilon_3\epsilon_4 \oplus \epsilon_1\epsilon_2\epsilon_4 \oplus \epsilon_1\epsilon_2\epsilon_3, \\ c &= \epsilon_1\epsilon_2 \oplus \epsilon_1\epsilon_3 \oplus \epsilon_1\epsilon_4 \oplus \epsilon_2\epsilon_3 \oplus \epsilon_2\epsilon_4 \oplus \epsilon_3\epsilon_4, \\ d &= \epsilon_1\epsilon_2\epsilon_3\epsilon_4. \end{aligned}$$

We will now show that this effect can be exploited for an attack. Regarding $\Omega(Y)$ for the case $w = 4$ we find that the coefficent to the highest power of Y is given by $a = \epsilon_1 \oplus \epsilon_2 \oplus \epsilon_3 \oplus \epsilon_4$. If $a = 0$, then the degree of $\Omega(Y)$ is zero, otherwise it is two. This means that in the case of $a = 0$ the number of iterations in the Inversion is four, in contrast to six in the general case. As a matter of fact, the EEA executes in the maximal number of iterations in the majority of the cases, as numerous experiments have shown.

We now show that the vulnerability is only given for even error weights. In the general case, the syndrome polynomial is given as

$$S(Y) \equiv \sum_{i=1}^{w} \frac{1}{Y \oplus \epsilon_i} \bmod g(Y) \tag{4}$$

Accordingly, the polynomial $\Omega(Y)$ and $\sigma(Y)$ with $\frac{\Omega(Y)}{\sigma(Y)} \equiv S(Y) \bmod g(Y)$ are found by bringing the right hand side of Equ. 4 to the common denominator. We have

$$S(Y) \equiv \sum_{i=1}^{w} \frac{Y^{w-1} \oplus \eta_{i,w-2}Y^{w-2} \oplus \eta_{i,w-3}Y^{w-3} \oplus \dots \oplus \eta_{i,0}}{\sigma(Y)} \bmod g(Y), \tag{5}$$

where $\eta_{i,j} \in \mathbb{F}_{2^m}$. In case of an even value of w, the sum over i results in the cancelation of the Y^{w-1} terms in the nominator. Thus, deg $(\Omega(Y)) \leqslant w-2$ and the coefficient Ω_{w-2} is is a function of the $\{\epsilon_i\}$, specifically $\Omega_{w-2} = \sum_{i=1}^{w} \epsilon_i$, giving us the control flow vulnerability, i.e. deg $(\Omega(Y))$ is smaller when this latter sum is zero.

In contrast, in case of odd values of w, deg $(\Omega(Y)) = w - 1$ and its highest coefficient is 1. Thus, for odd w, there is no according control flow vulnerability related to the degree of $\Omega(Y)$.

Furthermore, for even w, whenever we have $\Omega_{w-2} = 0$ the maximal degree of $\Omega(Y)$ is decreased by two, since we find

$$\Omega_{w-3} = \sum_{i=1}^{w} \eta_{i,w-3} = \sum_{i=1}^{w} \pi_{k,l}^{(1,w),\{i\}}(\epsilon_k\epsilon_l) = 0. \tag{6}$$

Here, we introduce the notation

$$\pi_{j,k}^{(1,x),\{i\}}(\epsilon_j\epsilon_k) := \sum_{j=2,k<j;j,k\neq i}^{x} \epsilon_j\epsilon_k,$$

$$\pi_{j,k,l}^{(1,x),\{i\}}(\epsilon_j\epsilon_k\epsilon_l) := \sum_{j=3,k<j,l<k;j,k,l\neq i}^{x} \epsilon_j\epsilon_k\epsilon_l,$$

$$\dots$$

The equality to zero in Equ. 6 is given because each combination of k and l is found in every summand term from the outer sum with the two exceptions $i = k$ and $i = l$, and consequently the remaining number these summands is even and thus they all cancel out.

Control Flow Analysis of the Key Equation EEA The control flow for the second EEA invocation, i.e. the solving of the Key Equation, for the case $w = 4$ has been analyzed in [11], there it is shown that in the case of $\sigma_3 = 0$, which conincides with the case $\deg(\Omega(Y)) = 0$, the number of iterations is zero, whereas in the case $\sigma_3 \neq 0$ it is one. In that work, a countermeasure is proposed that removes the possiblity to exploit the according timing differences in the second EEA invocation. However, due the fact that, as shown in Section 3.3, timing differences reveal $\sigma_3 = 0$ already at first EEA invocation, the countermeasure proposed in [11] is insuffient.

The Case $w = 6$ As for the case $w = 4$, the coefficient $\Omega_{w-2} = \Omega_4 = \sum_{i=1}^{6} \epsilon_i$, and $\Omega_{w-3} = \Omega_3 = 0$. Thus the same vulnerability reveals $\sum_{i=1}^{6} \epsilon_i = 0$. But for $w = 6$ a further vulnerability arises in the syndrome inversion. This is because also $\Omega_{w-5} = \sum_{i=1}^{w} \pi_{j,k,l,p}^{(1,w),\{i\}}(\epsilon_j\epsilon_k\epsilon_l\epsilon_p) = 0$ for even w because of the same arguments as for Ω_{w-3}, implying $\Omega_1 = 0$ for $w = 6$.

Thus, if for $w = 6$ we have $\Omega_4 = 0$ and also $\Omega_2 = 0$, $\deg(\Omega(Y)) = 0$ the maximal number of iterations in the syndrome inversion EEA is further reduced by two, making the case $\Omega_4 = 0 \wedge \Omega_2 = 0$ distinguishable from the case $\Omega_4 = 0 \wedge \Omega_2 \neq 0$. What remains is to show the information gained in this case.

Expanding Ω_2 we find

$$\Omega_2 = \sum_{i=1}^{6} \pi_{j,k,l}^{(1,6),\{i\}}(\epsilon_j\epsilon_k\epsilon_l) = \sum_{j=3,k<j,l<j}^{6} \epsilon_j\epsilon_k\epsilon_l. \qquad (7)$$

If the attacker has already determined the zero element of the support, i.e. $\gamma_{P_0} = \alpha_0 = 0$, he can enforce a simplified version of Equ. 7 by choosing, w.l.o.g, $\epsilon_6 = 0$:

$$\Omega_{2,(\epsilon_6=0)} = \sum_{j=3,k<j,l<j}^{5} \epsilon_j\epsilon_k\epsilon_l.$$

Control Flow Analysis of the Key Equation EEA In order to determine the control flow in the second invocation of the EEA for the case $\Omega_4 = 0 \wedge \Omega_2 = 0$, we first note that for $w = 6$ we have

$$\sigma_{w-1} = \sigma_5 = \delta^{-1} \sum_{j=1}^{6} \epsilon_j = \Omega_4 \tag{8}$$

$$\sigma_{w-3} = \sigma_3 = \delta^{-1} \sum_{j=3,k<j,l<k}^{6} \epsilon_j \epsilon_k \epsilon_l = \Omega_2 \tag{9}$$

$$\sigma_{w-5} = \sigma_1 = \delta^{-1} \pi_{i,j,k,l,p}^{(1,6),\emptyset} (\epsilon_i \epsilon_j \epsilon_k \epsilon_l \epsilon_p) = \Omega_0 \tag{10}$$

Thus the computations in Patterson's Algorithm proceed as follows:

$$S^{-1}(Y) = \frac{\sigma_6 Y^6 \oplus \sigma_4 Y^4 \oplus \sigma_2 Y^2 \oplus \Omega_0 Y \oplus \sigma_0}{\Omega_0}$$

$$\tau(Y) = \sqrt{S^{-1}(Y) + Y} = \tau_3 Y^3 \oplus \tau_2 Y^2 \oplus \tau_1 Y \oplus \tau_0$$

Accordingly, the Key Equation solving EEA performs zero iterations, in contrast to the case where either $\Omega_4 \neq 0$ or $\Omega_2 \neq 0$, since then also $\sigma_5 \neq 0$ respectively $\sigma_3 \neq 0$, which causes contributions to the square root from odd powers of Y which in turn results in $\deg(\tau(Y)) = t - 1$ in the general case and a non-zero number of iterations in the Key Equation solving EEA.

4 The Effects of "Iteration-Skipping"

In Cor. 1 we only derived an upper bound for the number of iterations M performed in the Syndrome Inversion EEA based on the degrees of $\Omega(Y)$ and $\sigma(Y)$. Experiments show that for the majority of the ciphertexts, this maximal number of iterations is performed. However, it can happen that in one iteration we have $\deg(q_i(Y)) > 1$, this implies a reduced number of iterations $M < M_{\max}$. The only concrete lower bound we can give is that $M \geqslant 2$ if $\deg(\Omega(Y)) \neq 0$, because in this case we have to pass at least one intermediate iteration with $r_i(Y) = \delta\Omega(Y)$ which cannot be the last iteration [4]. Thus, we we have to consider the effects of "iteration-skipping" for the $w = 4$ and $w = 6$ attacks.

But experimental results and closer look into the implementation of the EEA algorithm show that the effect of iteration skipping is actually minimal. Firstly, this is because the smaller number of iterations is compensated by a higher complexity of those iterations where $q_i(Y) > 1$. Secondly, and much more significantly, the timing-based distinction of whether $\deg(\Omega(Y))$ equals zero or not is not affected by iteration skipping. This is because of the following observation.

Assume the case $w = 4$ executes in the maximal number of iterations. This means that, independently of $\deg(\Omega(Y))$, we have $b_4(Y) = \delta\sigma(Y)$ and $r_4(Y) =$

[4] In case of $\deg(\Omega(Y)) = 0$, we can in fact have a single iteration, as it is permanently the case for $w = 1$ and for instance frequently for $w = 2$.

$\delta\Omega(Y)$. Because of $\deg(\sigma(Y)) = 4$, we have that $\deg(q_1(Y)) = \deg(q_2(Y)) = \deg(q_3(Y)) = \deg(q_4(Y)) = 1$, and thus $\deg(r_i(Y)) - \deg(r_{i+1}(Y)) = 1$ for $i \in \{-1, 0, 1, 2\}$. Accordingly, to fulfill $\deg(r_4(Y)) \leqslant 2$, we must have $\deg(r_3(Y)) - \deg(r_4(Y)) = \deg(q_5(Y)) = t - 4 - \deg(\Omega(Y))$. This is a large degree of $q_5(Y)$ for realistic choices of t such as for instance $t = 33$ we use in our example attacks in subsequent sections. As a consequence, the complexity of the 5th iteration will be by degrees larger than that of the other iterations, because the complexity of the polynomial multiplication $q_5(Y)b_4(Y)$ in Alg. 4 certainly is linear[5] in $\deg(q_5(Y))$. Now we see that for the case $\deg(\Omega(Y)) = 0$ this highly complex EEA iteration will not occur, since then there is no 5th iteration, and for $\deg(\Omega(Y)) \neq 0$ it will inevitably occur. Accordingly, our assumption that timing analysis unambigously allows for the distinguishing of both cases is reinforced.

When relaxing the initial assumption that the maximal number of iterations $M_{\max}$ are executed, we easily see that the skipping of iterations cannot change anything about the previous statement about the highly complex iteration. Furthermore, these observations can be generalized to higher even values w in a straightforward manner.

5 Limitations of linear Equations for the secret Support

In this section, we give some fundamental observations about a system of equations which describes the secret support as it arises from a $w = 4$ and $w = 6$ attacks.

Corollary 2. *Let there be a system of linear equations describing a permutation of the elements of $\mathbb{F}_{2^m}$, i.e. $\{\gamma_{P_i}\}$ with $\gamma_i \in \mathbb{F}_{2^m}$ and $\{P_i\}$, $i \in \{0, \ldots 2^m - 1\}$. The rank of this equation system is no larger than $2^m - m$.*

Proof. There must be m linearly independent elements γ_i to form $\mathbb{F}_{2^m}$. □

In the experimental analysis however, the rank is always $m - 1$. This restriction seems to be a result of the type of equations that are gained.

Corollary 3. *Given a system of linear equations describing a permutation of the elements of $\mathbb{F}_{2^m}$, there are at least*

$$b_m = \prod_{k=0}^{m-1} \left(2^m - 1 - \sum_{i=1}^{k} \binom{k}{i} \right) (m!)^{-1}$$

conflict-free solutions, i.e. yielding the 2^m different elements of $\mathbb{F}_{2^m}$.

Proof. With r being the rank of the equation system, for solving the system $d = n - r$ elements have to be guessed. Among these d elements, for the real solution (i.e. yielding the correct permutation P), there is a specific set Q of the

[5] Assuming a straigforward implementation of the polynomial multiplication.

indices of the m elements that have to be linearly independent. In the following, we only regard those guesses where Q is chosen correctly.

We now show that for any such choice of the basis elements, there exists a choice of the eventually remaining elements that have to be guessed until the system can be solved, such that the solution is a permutation of $\mathbb{F}_{2^m}$. For the real soluation, the values specific to which are written with a bar above them, in the equation system for each element of the permutation we have an equation

$$\bar{\gamma}_i = \sum_{i \in B^*} \bar{\beta}_i + \sum_{i \in D} \bar{\rho}_i = \sum_{i \in B^*} \bar{\beta}_i + \sum_{i \in D} \sum_{j \in R_i} \bar{\beta}_j. \tag{11}$$

B^* and D are read from the equation system: the former is the set of indices of the basis elements $\{\bar{\beta}_i\}$ from Q contributing to $\bar{\gamma}_i$, and D is the set of indices of the remaing to-be-guessed elements $\{\bar{\rho}_i\}$ contributing to this element. Thus the left hand side of Equ. (11) describes each element beyond the guessed elements as a linear combination of the guessed elements. In the right hand side of that equation, we expand each $\bar{\rho}_i$ in terms of its linear combination from the basis elements, thus R_i amounts to the set of the indices of basis elements $\{\bar{\beta}_j\}$ forming $\bar{\rho}_i$.

Now regard the equations we have for a guess with correctly chosen Q, but different values for the basis elements $\{\beta_i\}$:

$$\gamma_i = \sum_{i \in B^*} \beta_i + \sum_{i \in D} \sum_{j \in R_i} \beta_j. \tag{12}$$

From Equ. (12) we see that using the same set of indices of basis elements R_i for the each of the guessed $\{\rho_i\}$ as for the real $\{\bar{\rho}_i\}$ also yields a conflict-free solution, since if the $\{\bar{\gamma}_i\}$ according to Equ. (11) are all different from each other, then the same applies to the $\{\gamma_i\}$ according to Equ. (11).

Thus for each possible basis of $\mathbb{F}_{2^m}$, there exists at least one conflict-free solution to the equation system. The number b_m is just the number of possible basis for the field. $\square$

This implies a vast number of wrong solutions which could only be verified by performing a complete key recovery based on each solution for the permutation.

The integration of of the non-linear Equ. (7) introduces a means of detecting such wrong solutions to the linear equation system, since in contrast to the linear equations they are "aware" of the field's basis. However, they do not allow to acutally reduce the defect of the equation system beyond $d = m$: Since according to Equ. (8), $\delta\Omega_4 = \sum_{j=1}^{6} \epsilon_j = 0$ is given whenever a non-linear equation involving the very same $\{\epsilon_j\}$ is found, a reduction of the defect through the non-linear equation would imply a reduction through a linear equation, which is impossible according to Col. 2.

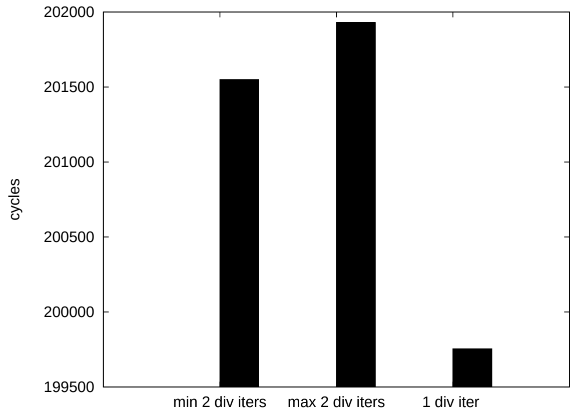

Fig. 1: Cycle counts taken on an ATMega1284P for the syndrome decoding with parameters $n = 512$, $t = 33$ and inputs of the n different error vectors of hamming weight $w = 1$. The first two bars depict the minimal and maximal values of the $n - 1$ runs where the division performs two iterations (corresponding to $\sigma_0 \neq 0$). The third bar gives the cycles taken when one iteration in the division occured (i.e. the single case where $\sigma_0 = \epsilon_1 = 0$).

6 Experimental Results

6.1 Attacking the Zero-Element of the Support

An experimental attack against the zero-element of the support was conducted by measuring the decrytion time of the $n = 512$ ciphertexts with $w = 1$. From the results depicted in Fig. 1 we find that the case $\sigma_0 = \epsilon_1 = 0$ can be identified by a running time which lies clearly below the range of timings found for ciphertexts which cause $\sigma_0 \neq 0$.

6.2 Linear Equations

The experimental results of the timing side channel vulnerability based on "$w =$ 4 error vectors" from Section 3.3 are given in Figure 2. It shows the minimal an maximal values for the case $\sigma_3 = 0$ with $M = 4$ iterations in the syndrome inversion, labeled A, and for comparison it shows timings for the case $\sigma_3 \neq 0$ with reduced iteration counts (due to "iteration skipping" as defined in Section 4), labeled B and C. Remember that N denotes the number of iterations in the key equation solving EEA. Note that even the case C, where the total number of EEA iterations is the same as in the case A, has timings clearly distinguishable from those of A; and that the difference in the timings between B and C, which differ in one iteration of the syndrome inversion, is comparatively small. Both of these observations are in line with the theoretic analysis given in Section 4.

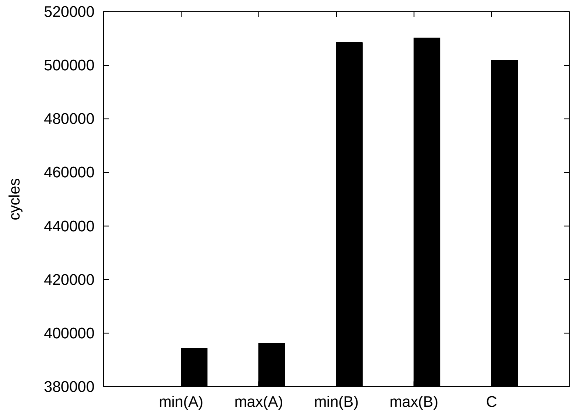

Fig. 2: Cycle counts taken on an ATMega1284P for the syndrome decoding with parameters $n = 512$, $t = 33$. The minimal and maximal cycle counts for the cases $A : M = 4,\ N = 0,\ \sigma_3 = 0$; $B : M = 4,\ N = 1,\ \sigma_3 \neq 0$; $C : M = 3,\ N = 1,\ \sigma_3 \neq 0$.

7 Conclusion

The results of this work show that timing attacks based on control flow vulnerabilities in the syndrome inversion are a threat to the confidentiality of the secret key. Though a practical attack based on the side channel information gained through the vulnerabilities described in this work with secure code parameters remains an open question, some additional side channel information about the secret support may solve this problem.

The question of countermeasure against this attack remains open in this work, but two possibilities seem to suggest themselves: the first would be in course of the countermeasures given [12], where "premature" abortion of the key equation solving EEA is prevented enforcing the "missing" iterations. This however is a delicate undertaking, as even the smallest timing differences have to be prohibited and thus the complexity of the individual iterations must be accounted for (consider for instance the "$w = 1$ attacks" from Section 3.2). The second option would be to alter the cryptosystem's parameter specification: during the encryption, only $t - 1$ parameters are added, and the prior to the standard decryption operation, another "bit flip error", the position of which is pseudorandomly (yet unpredicably for the attacker) derived from the ciphertext, is applied. The advantage of this latter suggestion is that it is much more reliable and also prevents potentially undetected subtle vulnerablities leading to similar equations as those from this work, as such equations in general will be rendered useless if one of the involved variables takes on random values, different for each ciphertext.

This work is another brick in the slowly but persistently growing "wall of timing side channel vulnerabilities" of code-based cryptosystems. Beyond timing attacks, the leakage-critical control flow ambiguities will pose a great challenge for hardware designers. Thus a generic countermeasure as suggested above might acutally be a feasible solution, even though it demands an increase of security parameters to compensate for the lower error weight.

References

1. R. J. McEliece: A public key cryptosystem based on algebraic coding theory. DSN progress report **42–44** (1978) 114–116
2. Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. In: Problems Control Inform. Theory. Volume Vol. 15, number 2. (1986) 159–166
3. Bernstein, D.J., Buchmann, J., Dahmen, E.: Post Quantum Cryptography. Springer Publishing Company, Incorporated (2008)
4. Biswas, B., Sendrier, N.: McEliece Cryptosystem Implementation: Theory and Practice. In: PQCrypto. (2008) 47–62
5. Heyse, S.: Low-Reiter: Niederreiter Encryption Scheme for Embedded Microcontrollers. In Sendrier, N., ed.: Post-Quantum Cryptography. Volume 6061 of Lecture Notes in Computer Science., Springer Berlin / Heidelberg (2010) 165–181
6. Eisenbarth, T., Güneysu, T., Heyse, S., Paar, C.: MicroEliece: McEliece for Embedded Devices. In: CHES '09: Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems, Berlin, Heidelberg, Springer-Verlag (2009) 49–64
7. Shoufan, A., Wink, T., Molter, G., Huss, S., Strenzke, F.: A Novel Processor Architecture for McEliece Cryptosystem and FPGA Platforms. In: ASAP '09: Proceedings of the 2009 20th IEEE International Conference on Application-specific Systems, Architectures and Processors, Washington, DC, USA, IEEE Computer Society (2009) 98–105
8. Strenzke, F.: A Smart Card Implementation of the McEliece PKC. In: Information Security Theory and Practices. Security and Privacy of Pervasive Systems and Smart Devices. Volume 6033 of Lecture Notes in Computer Science., Springer Berlin / Heidelberg (2010) 47–59
9. Molter, H.G., Stöttinger, M., Shoufan, A., Strenzke, F.: A Simple Power Analysis Attack on a McEliece Cryptoprocessor. Journal of Cryptographic Engineering (2011)
10. Strenzke, F., Tews, E., Molter, H., Overbeck, R., Shoufan, A.: Side Channels in the McEliece PKC. In Buchmann, J., Ding, J., eds.: Post-Quantum Cryptography. Volume 5299 of Lecture Notes in Computer Science., Springer Berlin / Heidelberg (2008) 216–229
11. Strenzke, F.: A Timing Attack against the secret Permutation in the McEliece PKC. In: The third international Workshop on Post-Quantum Cryptography PQCRYPTO 2010, Lecture Notes in Computer Science
12. Shoufan, A., Strenzke, F., Molter, H., Stöttinger, M.: A Timing Attack against Patterson Algorithm in the McEliece PKC. In Lee, D., Hong, S., eds.: Information, Security and Cryptology - ICISC 2009. Volume 5984 of Lecture Notes in Computer Science., Springer Berlin / Heidelberg (2009) 161–175
13. Heyse, S., Moradi, A., Paar, C.: Practical power analysis attacks on software implementations of mceliece. In Sendrier, N., ed.: PQCrypto. Volume 6061 of Lecture Notes in Computer Science., Springer (2010) 108–125
14. Kocher, P.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology (1996) 104–113

15. Kocher, P., Jaff, J., Jun, B.: Differential Power Analysis. Advances in Cryptology-CRYPTO'99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings **1666** (1999) 388–397
16. Engelbert, D., Overbeck, R., Schmidt, A.: A Summary of McEliece-Type Cryptosystems and their Security. Journal of Mathematical Cryptology **1**(2) (2006) 151–199
17. Goppa, V.D.: A new class of linear correcting codes. Problems of Information Transmission **6** (1970) 207–212
18. F. J. MacWilliams and N. J. A. Sloane: The theory of error correcting codes. North Holland (1997)
19. Bernstein, D.J., Lange, T., Peters, C.: Attacking and defending the McEliece cryptosystem. Post-Quantum Cryptography, LNCS **5299** (2008) 31–46
20. Kobara, K., Imai, H.: Semantically secure McEliece public-key cryptosystems - conversions for McEliece PKC. Practice and Theory in Public Key Cryptography - PKC '01 Proceedings (2001)
21. Pointcheval, D.: Chosen-chipertext security for any one-way cryptosystem. Proc. of PKC **1751** (2000) 129–146
22. Biswas, B., Sendrier, N.: McEliece Cryptosystem Implementation: Theory and Practice. Post-Quantum Cryptography, LNCS **5299** (2008) 47–62
23. Patterson, N.: Algebraic decoding of Goppa codes. IEEE Trans. Info.Theory **21** (1975) 203–207