Sloppy Alice attacks! Adaptive chosen ciphertext attacks on the McEliece Public-Key Cryptosystem

Eric R. Verheul, Jeroen M. Doumen and Henk C.A. van Tilborg
Department of Math. and Comp. Sc.,
P.O. Box 513, Eindhoven University of Technology,
5600 MB Eindhoven, The Netherlands.
Eric.Verheul@pobox.com, doumen@win.tue.nl, henkvt@win.tue.nl

Abstract

We introduce new adaptive chosen ciphertext attacks, called *Sloppy Alice Attacks*, in which a malicious sender or an adaptive eavesdropper Eve has an oracle which allows her to find out whether a sent encrypted message does, or does not, decrypt properly. From this information she can extract the plaintext that was encrypted. In this paper we show that the McEliece public–key cryptosystem is susceptible to Sloppy Alice attacks.

Keywords: Ciphertext Only Adaptive Cryptanalysis, Decryption Problems, Differential Cryptanalysis, Differential Fault Analysis, McEliece Public–Key Cryptosystem

1 Introduction

In the last decade, several forms of attacks have been published where some of the inputs of an encryption system with a secret fixed key are adaptively chosen. By letting each new input (either plaintext or ciphertext) depend on the previous outputs and by looking at certain aspects of the resulting output at each step, additional secret information of the cryptosystem (e.g. the fixed key) may be determined. Among the studied aspects of the output are:

- the differences in the output when the differences in the plain inputs are known, see [4], [15];
- some statistical or number–theoretic aspects of the output of the cryptosystem when errors are inflicted to the cryptosystem itself (e.g. by radiation), see for instance [5], [6];

- the execution time when the precise complexity of the underlying cryptographic process is known, see [13].

In this paper, we will look at a different setting. Here an attacker, called Eve, has access (for instance by interception) to one or more encrypted messages (called ciphertexts) for a receiver Bob.

We also suppose that Eve has access to an oracle that can tell her whether a ciphertext deciphers correctly or not. This oracle scenario was already studied in a more general form by Goldwasser, Micali and Tong [9]. Here we will present an efficient algorithm against the McEliece cryptosystem.

Note that in practice such an oracle might be easy to obtain: Eve may have access to Bob's decryption device or Eve might be an active eavesdropper. Another possibility is that Bob's decryption device is in fact automated, and will send a reply if the decryption somehow went wrong, thus asking for a retransmission. This reply can then be intercepted by Eve.

Now the general idea of this attack is based on the following components:

1. Eve alters the ciphertext slightly in such a way that there is a reasonable probability that the message still deciphers correctly and sends the altered message to her oracle.
2. Knowledge on whether the message still deciphers correctly or not reveals new information and opens interesting new possibilities for adapting the ciphertext.

Eve will continue to alter messages in this way (called a *round*), until she has retrieved enough secret information. It is very likely that Eve will have to send a considerable number of altered messages.

In the case that Eve uses Bob as her oracle (she sends the messages to him and checks for a retransmission request), this means that Bob (or the protocol he is using) has to be willing to ask the presumed sender Alice over and over again for retransmission. This is only likely if Bob accepts sloppy (or faulty) behavior of Alice (or her computer). That is why we refer to this type of attacks as *Sloppy Alice attacks* (SA–attacks).

In [19] we discuss Sloppy Alice attacks on key–recovery schemes. Here, we will concentrate on SA attacks on the McEliece [16] public–key cryptosystem, which is based on algebraic coding theory and more in particular on *Goppa codes.* So we assume that Eve has a validly encrypted McEliece message for Bob which she can alter and for which she is able to find out (e.g. by using her oracle) if it still is a validly encrypted McEliece message.

The outline of this paper is as follows: in Section 2 we will describe the McEliece public–key cryptosystem – in a general form – and we will determine the value of the *maximum error correcting capability* that is attained by two most important decoding (deciphering) methods for Goppa codes: the Berlekamp–Massey algorithm and Euclid's algorithm. The proof of this value is given in Appendix A. Section 3 is the main part of this paper; there we will describe an effective SA–attack on the McEliece public–key cryptosystem based on the maximum error correcting property of the above two decoding algorithms. In Section 4, we consider some countermeasures against the described SA–attacks. Finally, we summarize our findings in Section 5.

Related Work
The attack described here differs from the one in [2] where it is assumed that the (original) sender, instead of an eavesdropper, sends the same message more than once using different random error vectors.

Also, please note that since the initial submission of this paper in 1998, an independent description of a similar algorithm has appeared in [10].

2 The McEliece Public–Key Cryptosystem

We shall give some introductory remarks on error–correcting codes. Let $V_n(q)$ denote an n–dimensional vector space over the finite field $GF(q)$ (where q is a prime power). Let $\mathcal{C}$ be a k dimensional linear subspace of $V_n(q)$. It will be called a linear $[n, k]$ *code.* The elements in $\mathcal{C}$ are similarly called *codewords.* A *generator matrix* G for $\mathcal{C}$ is a $k \times n$ (q–ary) matrix whose rows span $\mathcal{C}$. This means that each codeword $\mathbf{c}$ can be written as $\mathbf{c} = \mathbf{m}G$ (in $\mathcal{C}$) with $\mathbf{m} \in V_k(q)$. One can describe this by saying that the information vector $\mathbf{m}$ is encoded in the codeword $\mathbf{c}$. The quantity $n - k$ is the *redundancy* of $\mathcal{C}$. It gives the number of excess symbols in $\mathbf{c}$ with regards to $\mathbf{m}$.

Now $\mathbf{c}$ is sent over an (unreliable) channel and certain errors may be inflicted on $\mathbf{c}$: the received vector is $\mathbf{r} = \mathbf{c} + \mathbf{e}$ where $\mathbf{e}$ is the so called *error vector.* Let the Hamming weight $w_H(\underline{x})$ of a vector $\mathbf{x}$ simply count the number of non–zero coordinates of $\mathbf{x}$. If the weight of $\mathbf{e}$ is not too large, the received vector $\mathbf{r}$ coincides on many coordinates with $\mathbf{c}$ and so $\mathbf{c}$ can be recovered.

To this end, the *(Hamming) distance* $d_H(\mathbf{x}, \mathbf{y})$ between vectors $\mathbf{x}$ and $\mathbf{y}$ is defined as the number of coordinates where $\mathbf{x}$ and $\mathbf{y}$ differ. Note that $d_H(\mathbf{x}, \mathbf{y}) = w_H(\mathbf{x} - \mathbf{y})$. The *minimum distance* d of $\mathcal{C}$ is defined as the minimum Hamming distance between different codewords in $\mathcal{C}$. Since $\mathcal{C}$ is linear, an equivalent definition would be that d is the minimum non–zero weight in $\mathcal{C}$. A code with minimum distace d will be denoted as a $[n, k, d]$ code. The number $t = \lfloor (d-1)/2 \rfloor$ is called the *error–correcting capability* of $\mathcal{C}$. It follows from the triangle inequality that for each element $\mathbf{r}$ in $V_n(q)$ there can be at most one element $\mathbf{c}$ in $\mathcal{C}$ at distance $\leq t$ to it. So, in principle, one can correct up to t errors inflicted to an element in $\mathcal{C}$ by finding the nearest point in $\mathcal{C}$. However, in practice the process of determining the nearest point (called *decoding*) is often very complex. To illustrate, the problem for general linear codes on deciding on whether there exists a point in $\mathcal{C}$ at a given distance of a given point $x \in V_n(q)$ is known to be in the class NP–complete, see [3]. Fortunately there are certain classes of linear codes where decoding can be done quite effectively, see below.

In 1978, McEliece [16] proposed a public–key cryptosystem based on the general difficulty of decoding. Consider a generator matrix G, generating a q–ary code $\mathcal{C}$ with parameters $[n, k, d]$, which is constructed by a user Bob. Let $0 \leq t \leq e = \lfloor (d-1)/2 \rfloor$ and let $\mathcal{A}_t$ be an effective decoding algorithm for $\mathcal{C}$ that can correct up to t errors.

Now, to use this in a cryptographic setting, Bob generates a random, invertible, q–ary, $k \times k$ matrix S and a random permutation matrix P of size $n \times n$. The public key of Bob is $G' = SGP$ together with the value of t. The matrices, S, G, P are kept secret. The idea is that G', although it generates a codespace $\mathcal{C}'$ which is equivalent to $\mathcal{C}$, behaves like a "random" generator matrix for which the decoding problem is hard.

Now suppose another user, say Alice, wants to encrypt a message $\mathbf{m} \in V_k(q)$ for Bob. Then, she generates a random error vector $\mathbf{e}$ of weight $\leq t$ and she forms:[1]

$$\mathbf{r} = \mathbf{m}G' + \mathbf{e} \quad (= \mathbf{m}SGP + \mathbf{e}) \tag{1}$$

On delivery, Bob calculates

$$\mathbf{r}P^{-1} = (\mathbf{m}S)G + \mathbf{e}P^{-1}.$$

As $\mathbf{e}P^{-1}$ has the same weight as $\mathbf{e}$, Bob can determine $\mathbf{m}S$ (and $\mathbf{e}P^{-1}$) from $\mathbf{r}P^{-1}$ by means of algorithm $\mathcal{A}_t$. Since S is a invertible matrix, Bob can easily determine $\mathbf{m}$, for instance by the method of Gaussian elimination.

More in particular, in his scheme McEliece proposed to use binary (i.e. $q = 2$), irreducible Goppa codes, with $n = 1024$, $k \approx 524$ and $t = 50$. There exist many (different) codes of these parameters, they are easy to generate (randomly) and efficient decoding algorithms for them are easy to find. McEliece's construction can be extended to larger classes of codes (for instance non–binary Goppa codes). We will not go into detail here as that is not necessary for our attack; it suffices to mention the following bounds on the security–related parameters of the system.

Assumption 2.1 (The security of McEliece cryptosystem) *The following observations can be made on the parameters of a McEliece public–key cryptosystem:*

Sec–1 $k \approx n/2 \geq 512$*: this makes "syndrome decoding" as well as an exhaustive search for finding the nearest codeword to the received word infeasible;*

Sec–2 $50 \leq t \leq 100$*: this makes all kinds of techniques that are based on guessing/finding k (almost) error free coordinates less time consuming than the methods in Sec–1, but still infeasible (see [1, 7, 16]).*

We shall now formulate a property of the decoding algorithm $\mathcal{A}_t$. This property will play a crucial role in the proposed SA Attack.

Property 2.1 (Maximum Error Property (MEP)) *On input of a vector* $\mathbf{r} \in V_n(q)$*, the decoding algorithm* $\mathcal{A}_t$ *will either return a codeword* $\mathbf{c}$ *in* $\mathcal{C}$ *at distance* $\leq t$ *to* $\mathbf{r}$ *(if such a codeword exists) or it will return an error–message.*

[1]In some variants of McEliece the weight of the error vector e is always exactly equal to t.

The MEP property states that the decoding algorithm $\mathcal{A}_t$, on input $\mathbf{r}$, never returns an element $\mathbf{c} \in C$ at distance more than t from $\mathbf{r}$. It is important to realize that if too many transmission errors have occurred, the received vector $\mathbf{r}$ may be at distance $\leq t$ form another codeword than the transmitted one. In this case $\mathcal{A}_t$ will not return an error message. The probability that this occurs, will be of importance in the analysis of our SA attack and will be discussed in Section 3.

We end this section with stating that two relevant decoding algorithms for Goppa codes have the MEP; the proof of this will be given in Appendix A.

Proposition 2.1 *Let $\mathcal{C}$ be a Goppa code with designed distance t, and let $\mathcal{A}_t$ be either Euclid's algorithm or the Berlekamp–Massey algorithm for decoding $\mathcal{C}$. Then $\mathcal{A}_t$ has the MEP.*

3 A SA–Attack on the McEliece Cryptosystem

We shall now describe the SA–attack on the McEliece cryptosystem. Let $\mathbf{g}'_i$ denote the i–th column of the (public) generator matrix G' $(1 \leq i \leq n)$.

Algorithm 3.1 (SA–attack on McEliece) *Assume that the decoding algorithm $\mathcal{A}_t$ used in a McEliece cryptosystem has the Maximal Error Property. Let $\mathbf{r}$ be the ciphertext sent by Alice and intercepted by Eve (it is of the form $\mathbf{r}_A = \mathbf{m}G' + \mathbf{e}_A$). Then Eve does the following:*

Step 1. Increase the number of errors made by Alice to exactly t.

In order to increase the number of errors to the maximum, Eve continues changing a random coordinate arbitrarily (though each coordinate is selected at most once) and sending the resulting codeword to Bob until an error message is returned, i.e. the message is not accepted as a valid McEliece ciphertext. Once this occurs, Eve knows that this message contains one error too many, and thus the previous message she sent to Bob has the maximum number of errors. She now goes on to step 2 with this message $\mathbf{r}'$.

Step 2. Determine enough error–free coordinates.

Once Eve knows she has a message with exactly t errors, she can start probing a random coordinate (different from all preceeding choices, including those made in step 1) by changing this arbitrarily in $\mathbf{r}$*', and sending the mutated message to Bob. If an error message is returned, this coordinate was error–free. Once enough error–free coordinates are determined, Eve can determine the plaintext in step 3.*

Step 3. Determine the plaintext.

Once Eve knows enough error–free coordinates, she can solve the matrix equation $r' = mG$ for the plaintext m by using Gaussian elimination on the columns corresponding with the (known) error–free coordinates.

Before formulating Theorem 3.1, we recall the following definition. Let A_w, $0 \leq w \leq n$ be the number of codewords in $\mathcal{C}$ of weight w. The weight distribution of an $[n,k,d]$ code $\mathcal{C}$ will be called *approximately binomial* if (in the context of the problem here) the weight distribution may be approximated as follows:

$$A_w \approx \frac{\binom{n}{w}(q-1)^w}{q^{n-k}}, \; d \leq w \leq n, \tag{2}$$

or, in other words, the cardinality of codewords with weight d is approximately $\binom{n}{w}(q-1)^w/q^{n-k}$.

Note that in this case,

$$\sum_{w=0}^{n} A_w \approx \sum_{w=0}^{n} \frac{\binom{n}{w}(q-1)^w}{q^{n-k}} = \frac{(1+(q-1))^n}{q^{n-k}} = q^k = |C|,$$

as it should be. Certainly the weight distribution of $V_n(q)$ itself is binomial (approximation (2) is actually an equality). We are not familiar with any result on how well the weight distribution of Goppa codes can be approximated by the binomial distribution, but based on [11, 12] it seems very reasonable to make that assumption here.

We will also assume that the minimum distance of the used code is odd, i.e. $d = 2t+1$.

Theorem 3.1 *With the notation of Section 2, let $\mathcal{C}$ be a McEliece–like cryptosystem based on a q–ary code with approximately binomial weight distribution and with a decoding algorithm that has the maximum error property, for instance a Goppa code. Then Algorithm 3.1 is a SA attack on $\mathcal{C}$ returning the plaintext* $\mathbf{m}$.

Let the number of times the loop in step 1 is executed be S_1. We have that $S_1 \leq 2t+1$ times. With large probability the loop in step 2 will be executed at most $k + \log_q(k) + X + 1$ times, where X is equal to $\min(t, 2t+1-S_1)$.

So, with large probability Algorithm 3.1 is a SA attack of at most $k + \log_q(k) + 2t + 2$ rounds.

We remark that if a binary code is used, then any error–coordinate is a one, so Algorithm 3.1 can be slightly improved to a SA attack of at most $k + 2t + 1$ rounds. We also remark, that if in step 2 of Algorithm 3.1 t rejections are encountered, then all errors introduced by Alice have been found and all remaining coordinates are error–free. In this case deciphering can be done much faster. Of course, the probability that this occurs is negligible.

Before proving Theorem 3.1, we need two lemmas. We first recall that the *binary entropy function* $h(x)$ is defined by: $h(0) = h(1) = 0$ and $h(x) = -x\log_2(x) - (1-x)\log_2(1-x)$ $(0 < x < 1)$.

Lemma 3.1 *In the notation of Section 2, let $\mathbf{e} \in V_n(q)$ be an 'error–vector' of weight $t+1$. Then the following holds.*

i) *There is at most one vector* $\mathbf{f} \in V_n(q)$ *of weight* $\leq t$ *such that* $\mathbf{e}+\mathbf{f} \in \mathcal{C}$. *Also, if such an* $\mathbf{f}$ *exists, then the weight of* $\mathbf{f}$ *is exactly equal to* t, *the supports (the sets of non–zero coordinates) of* $\mathbf{e}$ *and* $\mathbf{f}$ *are disjoint and* $d = 2t+1$.

ii) *If* $\mathbf{e}$ *is chosen uniformly random, then the probability* P *that a vector* $\mathbf{f}$ *of weight at most* t *exists such that* $\mathbf{e}+\mathbf{f} \in \mathcal{C}$ *is given by*

$$P = \frac{A_{2t+1}\binom{2t+1}{t+1}}{\binom{n}{t+1}(q-1)^{t+1}}. \tag{3}$$

iii) *If the weight distribution of* $\mathcal{C}$ *is approximately binomial, then*

$$P \approx \frac{\binom{n-(t+1)}{t}(q-1)^t}{q^{n-k}} \leq \frac{2^{(n-(t+1))h(\frac{t}{n-(t+1)})}(q-1)^t}{q^{n-k}}.$$

iv) *Assume that the weight distribution of the Goppa codes is indeed approximately binomial [11, 12]. If a binary Goppa code is used in a McEliece cryptosystem, the above probability* P *is negligible. To be more precies, when the parameters originally proposed by McEliece in [16] are used, we have* $P \leq 2^{-215}$. *When the improved parameters as mentioned in Assumptions 2.1 are used, we have that* $P \leq 2^{-54}$.

Proof: i) Suppose that two distinct candidates for $\mathbf{f}$ – as mentioned in the first part of the lemma – exist, say $\mathbf{f}$ and $\ddot{\mathbf{e}}$. Then

$$d_H(\mathbf{e}+\mathbf{f}, \mathbf{e}+\ddot{\mathbf{e}}) = d_H(\mathbf{f}, \ddot{\mathbf{e}}) = w_H(\mathbf{f}-\ddot{\mathbf{e}}) \leq w_H(\mathbf{f}) + w_H(\ddot{\mathbf{e}}) =\leq 2t < d.$$

As $\mathbf{e}+\mathbf{f}$ and $\mathbf{e}+\ddot{\mathbf{e}}$ are two distinct members of $\mathcal{C}$ at distance less than d, we arrive at a contradiction. A similar argument shows that the weight of $\mathbf{f}$ must be equal to t and that $d = 2t+1$.

ii) Let

$$B = \{(\mathbf{e}, \mathbf{f}) \in V_n(q) \times V_n(q) \mid \mathbf{e}+\mathbf{f} \in C,\ w_H(\mathbf{e}) = t+1,\ w_H(\mathbf{f}) = t\}.$$

Let $\mathbf{c}$ be a codeword in $\mathcal{C}$ of weight $2t+1$. Then each $(t+1)$–subset of the support of $\mathbf{c}$ gives rise to a unique pair $(\mathbf{e}, \mathbf{f}) \in B$ (change the remaining t non–zero coordinates of $\mathbf{c}$ into a zero, resp. the $t+1$ coordinates themselves). Conversely, each element $(\mathbf{e}, \mathbf{f}) \in B$ can be obtained this way. Hence it follows that $|B| = A_{2t+1}\binom{2t+1}{t+1}$.

The total number of 'error–vectors' of weight $t+1$ in $V_n(q)$ is $\binom{n}{t+1}(q-1)^{t+1}$. The probability P is the quotient of $|B|$ and this number.

iii) By assumption

$$A_{2t+1} \approx \frac{\binom{n}{2t+1}(q-1)^d}{q^{n-k}}.$$

It follows from ii) that

$$P \approx \frac{\binom{n}{2t+1}(q-1)^{2t+1}\binom{2t+1}{t+1}}{q^{n-k}\binom{n}{t+1}(q-1)^{t+1}} = \frac{\binom{n-(t+1)}{t}(q-1)^t}{q^{n-k}}.$$

To show the inequality in iii), we note that the binomial theorem implies for each $0 \le m \le n$ the inequality

$$n^n = (m+(n-m))^n = \sum_{i=0}^{n} \binom{n}{i} m^i (n-m)^{n-i} \ge \binom{n}{m} m^m (n-m)^{n-m}.$$

This can be rewritten as

$$\binom{n}{m} \le \frac{n^n}{m^m(n-m)^{n-m}} = \frac{2^{n\log_2 n}}{2^{m\log_2 m}2^{(n-m)\log_2(n-m)}} = 2^{-m\log_2(m/n)-(n-m)\log_2((n-m)/n)} = 2^{nH(m/n)}.$$

iv) With the assumption that the weight distributions of the Goppa codes are indeed approximately binomial, the probability P mentioned in ii) can be approximated using iii). If we take the parameters proposed by McEliece, i.e. $n = 1024$, $k = 524$, $t = 50$, we obtain

$$P \le \frac{2^{975*H(50/975)}}{2^{500}} \approx \frac{2^{285}}{2^{500}} = 2^{-215},$$

which is a negligible. Similarly, if we take the parameters as in Assumption 2.1 we obtain that $P \le 2^{-54}$. So the same holds for general McEliece cryptosystems, provided of course the weight distribution of code is approximately binomial.

$\square$

The following observation may be of interest to the reader. It is well known that for a perfect t–error correcting code $\binom{2t+1}{t}A_{2t+1} = \binom{n}{t+1}(q-1)^{t+1}$ (e.g. see [18, Problem 3.4.9]). Substitution of this relation into (3) gives $P = 1$, as it should be for a perfect code: each word at distance $t+1$ from one codeword lies at distance t from exactly one other codeword. Thus a Sloppy Alice attack on McEliece–like cryptosystem which uses a perfect code will not work, since each vector can be decoded.

Lemma 3.2 *The probability that a random $k \times (k + \log_q k)$ q–ary matrix has rank k is at least* $1 - \frac{1}{k}$.

Proof: Let $P(k,m)$, $m \ge k$, denote the probability that a random $k \times m$ binary matrix A has rank k. Looking at the rows of A we observe that the first row of A should be non–zero, the second row should be independent of the first, etc. This argument leads to

$$\begin{aligned} P(k,m) &= \frac{\prod_{i=0}^{k-1}(q^m - q^i)}{\prod_{i=0}^{k-1} q^m} = \prod_{i=m-k+1}^{m} (1 - \frac{1}{q^i}) \\ &\overset{(*)}{\geq} 1 - \sum_{i=m-k+1}^{m} \frac{1}{q^i} = 1 - \frac{1 - \frac{1}{q^k}}{(q-1)q^{m-k}} \geq 1 - \frac{1}{q^{m-k}}, \end{aligned}$$

where $(*)$ follows quite easily with an induction argument.

Now substituting $m = k + \log_q k$ in the above relation gives

$$P(k, k + \log_q k) \geq 1 - \frac{1}{k}.$$

□

Proof of Theorem 3.1

We start with a general observation. Consider any $\mathbf{r} = \mathbf{c} + \mathbf{e}$ where $\mathbf{c} \in C$ and $w_H(\mathbf{e}) = s \leq t$. If we change the ith coordinate of $\mathbf{r}$, which can be described by adding a vector $\mathbf{u}$ of weight 1 and with support $\{i\}$ to $\mathbf{r}$, then there are three possibilities (only two if $q = 2$) for the resulting $\mathbf{r}' = \mathbf{r} + \mathbf{u} = \mathbf{c} + \mathbf{e}'$.

1. $w_H(\mathbf{e}') = s - 1$ iff $e_i \neq 0$ and $u_i = -e_i$;
2. $w_H(\mathbf{e}') = s$ iff $e_i \neq 0$ and $u_i \neq -e_i$
   This is impossible if $q = 2$, because both are also non–zero in this case)
3. $w_H(\mathbf{e}') = s + 1$ iff $e_i = 0$.

Consider step 1 of Algorithm 3.1. For the range $0 \leq i \leq 2t+1$, let $\mathbf{e}^{(i)} = \mathbf{r}^{(i)} - \mathbf{c}$, that is, $\mathbf{r}^{(i)} = \mathbf{c} + \mathbf{e}^{(i)}$. Of course each $\mathbf{e}^{(i)}$ is unknown to Eve and $\mathbf{e}^{(0)} = \mathbf{e}_A$. As $w_H(\mathbf{e}^{(0)}) = w_H(\mathbf{e}_M) \leq t$ it follows that there exists a first $0 < i \leq 2t+1$ in step 1, such that $w_H(\mathbf{e}^{(i)}) = t$ and $w_H(\mathbf{e}^{(i+1)}) = t + 1$. So, for $0 \leq j \leq i$ execution of the decoding algorithm $\mathcal{A}_t$ applied by Bob to $\mathbf{r}^{(j)}$ does not result in an error–message.

Note that i can only reach the value $2t+1$ in the (extremely unlikely) case that the $2t+1$ errors introduced by Eve in this step of the SA algorithm include all the errors that Alice originally has added to $\mathbf{c}$. In this case, we can immediately proceed to step 3 of the algorithm since all other coordinates will be error–free, and contain enough independent columns of the generator matrix G.

Next, we claim that the $\mathcal{A}_t$ applied to $\mathbf{r}^{(i+1)} = \mathbf{c} + \mathbf{e}^{(i+1)}$ with $\mathbf{e}^{(i+1)}$ of weight $t+1$, will result in an error–message (and we go to step 2). Indeed, if $\mathcal{A}_t$ applied to $\mathbf{r}^{(i+1)}$ does not issue an error–message then $\mathbf{r}^{(i+1)}$ lies at distance at most t from $\mathcal{C}$ by MEP. This means that there exist a codeword $\mathbf{c}'$ in $\mathcal{C}$ and a vector $\mathbf{e}'$ in $V_n(q)$ of weight at most t such that $\mathbf{r}^{i+1} = \mathbf{c}' + \mathbf{e}'$. Hence, we are in the situation of Lemma 3.1, from which it follows that this situation has a negligible probability of occurring.

In step 2, write $\mathbf{r}' = \mathbf{c} + \mathbf{e}'$, with $w_H(\mathbf{e}') = t$. We follow the same reasoning as above. When the f–th coordinate in $\mathbf{r}$ is changed, the obtained vector $\hat{\mathbf{r}}$ will not be accepted by $\mathcal{A}_t$ (i.e. without error–messages) if and only if (with a negligible probability of failure) the f–th coordinate of $\mathbf{e}'$ is zero.

The number of loops in this part of the SA algorithm follows directly from Lemma 3.2. Since we only take coordinates in step 2 different from those selected in step 1, we have to add an extra tern $+X$ that reflects the (marginal) possibility that X times a change made in this step canceled out an error introduced by Alice.

Because $S_1 + X \leq 2t+1$ with large probability the algorithm takes at most $k + \log_q(k) + 2t + 1$ rounds.

$\square$

4 Countermeasures to SA attacks on the McEliece cryptosystem

Countermeasures to the SA attack on the McEliece cryptosystem, should – at least – aim to achieve that there is no correlation between deciphering problems and the number of errors applied to the plaintext.

First, in the case that Bob is Eve's oracle, Bob could come up with the idea of checking for repeated messages. This would detect a SA–attack as described above, but nothing prevents the attacker Eve from adding a random codeword from $\mathcal{C}$ to her probe each round. This preserves the error–vector e, and will allow Eve to conduct her attack as usual with little additional effort.

As a second idea one might consider to fix the weight of the error vector to *exactly* t, (or any $t' \leq t$). (cf. equation (1)) and to return an error–message when in the deciphering process an error vector is encountered of weight unequal to t (or the chosen t'), that is, irrespective of whether successful decoding is still possible.

Let us illustrate how also in this setting effective SA attacks are still possible. First of all, suppose that the used code is non–binary. If one 'probes' a coordinate in step 2, say the i–th coordinate, twice but with different values, then Bob will always return an error–message if that coordinate is error–free (since there are $t+1$ errors in both probes). However, if there's an error on that coordinate, at most one probe will return no error message (since Eve only alters the value of $\mathbf{e}_i$ and not the weight of the error–vector).

So we have the following situation:

- if both probes give an error–message, then $e_i = 0$;
- if only one probe gives an error–message, then $e_i \neq 0$ and e_i is in fact determined;
- if none of the probes gives an error–message, then $e_i \neq 0$.

It easily follows that the plaintext can be found with around $k+t$ probes, i.e. in around $2(k+t)$ rounds.

If the used code is binary, then one starts by changing any coordinate, say the i–th, of $\mathbf{r}'$. In the setting here, this will always lead to an error–message from the oracle. Now we distinguish two possibilities.

With probability $(n-t)/n$ one has that $e_i = 0$. If one changes an additional coordinate in all possible ways, then $n-t-1$ times another error will have been introduced, resulting in an error–message from the oracle. Further, t times an error (introduced originally by the original sender) will be eliminated and so in this case one is back at t errors, leading to a correct decryption. In this way, all errors introduced by Alice can be found.

If $e_i = 1$ (with probability t/n), then additionally introduced single errors will lead $n-t$ times to correct decryptions (and coordinates with $e_i = 0$) and $t-1$ times to an error message.

In the case that Bob is in fact serving as Eve's oracle, a better countermeasure to the SA attack technique may be to introduce further redundancy in the system to enable Bob to check if an active eavesdropper is altering a proper ciphertext.

For instance, let Alice choose her plaintext $\mathbf{m}$ from $V_{k-64}(q)$ instead of $V_k(q)$. As before, she choose a random error vector of weight $\leq t$. Bob has published as part of his public key a cryptographically secure hash–function $h(.)$ that maps elements in $V_n(q)$ of weight $\leq t$ to words in $V_{64}(q)$ (the hash function can also be a system parameter). To encrypt $\mathbf{m}$ Eve computes (cf. equality (1)):

$$\mathbf{r} = (\mathbf{m}||h(\mathbf{m}||\mathbf{e}))G' + \mathbf{e} \quad (= (\mathbf{m}||h(\mathbf{e}))SGP + \mathbf{e}), \tag{4}$$

where $||$ stands for concatenation. When Bob receives a vector $\mathbf{r}'$, he will attempt to decode it. If this works, he will will find an error vector $\mathbf{e}'$ and an element $\mathbf{m}' \in V_k(q)$ satisfying

$$\mathbf{r}' = \mathbf{m}'G' + \mathbf{e}'.$$

Finally, he checks if $h(\mathbf{e}')$ equals the 64 right most coordinates of $\mathbf{m}'$. If this verification fails, an error–message is issued. This error–message will not give any information to Eve about the original choice of $\mathbf{e}$ by Alice. It follows that the SA attack as described in Section 3 fails.

However, if $h(.)$ is additive, i.e. $h(e_1+e_2) = h(e_1) + h(e_2)$, then one can easily extend the described SA attack. So it follows that is $h(.)$ should be reasonable "non–linear".

Note that the choice of 64 bits in the above example is arbitrary: this is a security parameter of the system which indicates how many message bits are used for increased security. We refer to [8] for a more general description of this construction. Also observe that as a side effect of this variation to the McEliece cryptosystem it loses its inherent error–correcting capabilities. This seems to be inevitable.

5 Conclusion

We have introduced the Sloppy Alice attack, an adaptive chosen ciphertext attack, which is based on the idea that (ordinary) users see no problem in overtly revealing whether or not an encrypted message deciphers correctly. We have described such an attack on the McEliece Public–Key cryptosystem.

The general conclusion is that such error–messages can be used to efficiently break the McEliece Public–Key cryptosystem. Therefore, at the very least, error–messages should be as non–descriptive as possible and users should be alerted when many encrypted messages do not decrypt properly. We have also proposed a variant of the McEliece system which is immune to SA attacks.

A Proof of Proposition 2.1

Before proving Proposition 2.1, we recall the definition and some properties of Goppa codes (see [14]). Let $g(x)$ be a monic polynomial of degree u over the finite field $GF(p^m)$ Also, let $\gamma_1, \gamma_2, \ldots, \gamma_n$ be n distinct members of $GF(p^m)$ such that $g(\gamma_i) \neq 0$ for all $1 \leq i \leq n$. The *Goppa code* $\mathcal{C}$ generated by the Goppa polynomial $g(x)$ consists of all words $\mathbf{c} = (c_1, c_2, \ldots, c_n) \in V_n(q)$ satisfying

$$\sum_{i=1}^{n} \frac{c_i}{x-\gamma_i} = 0 \quad (\mathbf{m}odg(x)).$$

The code $\mathcal{C}$ is linear of dimension $\geq n - m \cdot u$. Its error–correcting capability t is greater than or equal to $\lfloor u/2 \rfloor$. Moreover, if $p = 2$ and $g(x)$ does not have multiple zeros, then $t \geq \lfloor u \rfloor$.

Actually, in his original proposal, McEliece let $g(x)$ be an irreducible polynomial of degree 50 over $GF(2^{10})$.

An important features of Goppa codes is the existence of an efficient decoding algorithm $A_{t'}$ for any t' less than or equal to the designed error–correcting capability t. In practice, one therefore only corrects up–to the designed error–correcting capability.

Decoding goes as follows: suppose that the vector $\mathbf{r} = (r_1, r_2, \ldots, r_n)$ is received. The *syndrome* polynomial $S_\mathbf{r}(x)$ of the $\mathbf{r}$ is defined by

$$S_\mathbf{r}(x) \equiv \sum_{i=1}^{n} \frac{r_i}{x-\gamma_i} \pmod{g(x)}.$$

Observe that the inverse of each polynomial $x - \gamma_i$ exists modulo $g(x)$, as $g(x)$ is non–zero on the γ_i's. It is given by the solution of $u(x)(x-\gamma_i) \equiv 1 \pmod{g(x)}$. Also, the syndrome polynomial $S_\mathbf{r}(x)$ is zero if and only if $\mathbf{r} \in C$. Now suppose that $\mathbf{r}$ is of the following form

$$\mathbf{r} = \mathbf{c} + \mathbf{e} \;\; ; \;\; \mathbf{c} \in C, \; w_H(\mathbf{e}) \leq t, \tag{5}$$

Let $\mathcal{E}$ be the set of non–zero coordinates of $\mathbf{e} = (e_1, e_2, \ldots, e_n)$. Then the *error–locator* polynomial $\sigma(x)$ and the *error evaluator* polynomial $\omega_{\mathbf{r}}(x)$ are defined by

$$\sigma(x) = \prod_{i \in \mathcal{E}} (x - \gamma_i) \quad ; \quad \omega(x) = \sum_{i \in \mathcal{E}} e_i \prod_{j \in \mathcal{E} \setminus \{i\}} (x - \gamma_j).$$

Then $\deg(\omega(x)) < \deg(\sigma(x)) \leq t$, $\gcd(\sigma(x), \omega(x)) = 1$, and the following, so–called *key equation* holds:

$$S_{\mathbf{r}}(x)\sigma(x) \equiv \omega(x) \pmod{g(x)}.$$

The next (general) result is crucial for the decoding process. The proof is a direct consequence of [17, Theorem 8.5].

Lemma A.1 *Let $g(x)$, $u = \deg g(x)$ be defined as above. Let $S(x), \Sigma(x)$, and $\Omega(x)$ be q–ary polynomials and let the latter two be monic. Then the following two properties are equivalent:*

P–1 $S(x)\Sigma(x) \equiv \Omega(x) \pmod{g(x)}$, $\deg(\Omega(x)) < \deg(\Sigma(x)) \leq \lfloor u/2 \rfloor$,
$\deg \Sigma(x)$ *is minimal among all such* $\Sigma(x), \Omega(x)$*;*

P–2 $S(x)\Sigma(x) \equiv \Omega(x) \pmod{g(x)}$, $\deg(\Omega(x)) < \deg(\Sigma(x)) \leq \lfloor u/2 \rfloor$,
$\gcd(\Sigma(x), \Omega(x)) = 1$.

Moreover, a pair $(\Sigma(x)\ ,\ \Omega(x))$ satisfying P1 (or P2) will be unique.

In the remainder of this discussion, we shall first focus on the non–binary case (so $t = \lfloor u/2 \rfloor$). If the number of errors in $\mathbf{r}$ is not more than t, then the functions $\sigma(x)$ and $\omega(x)$ defined above form precisely the pair mentioned in Lemma A.1 w.r.t. $S(x) = S_{\mathbf{r}}(x)$. Clearly, from $\sigma(x)$ and $\omega(x)$ one can easily determine $\mathbf{c}$ and $\mathbf{e}$. Indeed, the error locations set $\mathcal{E}$ is completely determined by the roots of $\sigma(x)$. Further, for each i in $\mathcal{E}$ one can compute the error value e_i (the i–th coordinate of $\mathbf{e}$) from the relation $e_i = \omega(\gamma_i)/\sigma(\gamma_i)$. Finally, the original codeword $\mathbf{c}$ can be computed from as $\mathbf{c} = \mathbf{r} - \mathbf{e}$.

Proposition 2.1 states a property of the two main algorithms to determine the pair $\sigma(x)$, $\omega(x)$ satisfying property P1 (or P2). The Berlekamp–Massey algorithm tries to find the pair $(\sigma(x), \omega(x))$ satisfying the first condition (w.r.t. $S(x) = S_{\mathbf{r}}(x)$), whereas Euclid's algorithm tries to find a pair satisfying the second condition. It is important to note that the Berlekamp–Massey algorithm is successful if and only if Euclid's algorithm is; they also lead to the same result. This is irrespective of whether $S(x)$ is a syndrome polynomial $S_{\mathbf{r}}(x)$. As is clear from Lemma A.1, the behavior of the Berlekamp–Massey algorithm and Euclid's algorithm with "bad" input, i.e. when inputting a syndrome polynomial of a vector $\mathbf{r}$ at distance more than t from the code, is the same.

Proof of Proposition 2.1 (non–binary case):
Suppose that $\mathbf{r} \in V_n(q)$ is the input to A_t and suppose that A_t outputs a vector $\mathbf{v}$ in $V_n(q)$

(so without error–message). As we do not assume that the implementation explicitly checks that the output is actually a codeword we do not know beforehand that $\mathbf{v} \in \mathcal{C}$. Certainly, if $\mathbf{r}$ is of the form (5) then A_t will output $\mathbf{c}$. However, to prove the proposition we have to show the converse, i.e. that equality (5) holds with $\mathbf{c} = \mathbf{v}$.

To this end, let's analyze the decoding process. First the decoding process tries – using either Euclid's or Berlekamp–Massey's algorithm – to find $\sigma_{\mathbf{r}}(x)$ and $\omega_{\mathbf{r}}(x)$ satisfying property P1 (P2). If this fails we assume that an error–message will be returned. After that step the decoding process determines the presumed set of error locations $\mathcal{E}' = \{1 \leq i \leq n \mid \sigma_{\mathbf{r}}(\gamma_i) = 0\}$. If $\mathcal{E}'$ has cardinality strictly less than $\deg(\sigma_{\mathbf{r}}(x))$ we assume that an error–message will be given. Next, the decoding process determines a vector $\mathbf{e}'$ of weight $\leq t$ by

$$e'_i = \begin{cases} \omega_{\mathbf{r}}(\gamma_i)/\sigma'_{\mathbf{r}}(\gamma_i) & \text{if } i \in \mathcal{B}, \\ 0 & \text{otherwise.} \end{cases}$$

Note that all e'_i with $i \in \mathcal{E}'$ are necessarily non–zero, as otherwise the $\sigma_{\mathbf{r}}$ and $\omega_{\mathbf{r}}$ would not be relatively prime, as dictated by Lemma A.1(2). Finally, the decoding process determines $\mathbf{v} = \mathbf{r} - \mathbf{e}'$, which is returned by A_t.

Now to see that $\underline{v}$ is a member of $\mathcal{C}$, we consider the polynomials $\sigma_{\mathbf{e}'}(x)$ and $\omega_{\mathbf{e}'}(x)$ (corresponding to $\mathbf{e}'$ written as $\mathbf{e}' = \mathbf{0} + \mathbf{e}'$), i.e. $S_{\mathbf{e}'}(x)\sigma_{\mathbf{e}'}(x) \equiv \omega_{\mathbf{e}'}(x) \pmod{g(x)}$. First observe that the pair $(\sigma_{\mathbf{e}'}(x), \omega_{\mathbf{e}'}(x))$ is equal to the pair $(\sigma_{\mathbf{r}}(x), \omega_{\mathbf{r}}(x))$. That $\sigma_{\mathbf{r}}(x) = \sigma_{\mathbf{e}'}(x)$ is trivially true, because the error vectors in $\mathbf{r} = \mathbf{v} + \mathbf{e}'$ and $\mathbf{e}' = \mathbf{0} + \mathbf{e}'$ are the same. That $\omega_{\mathbf{r}}(x) = \omega_{\mathbf{e}'}(x)$ follows from the fact that both are polynomials of degree $\leq \deg(\sigma_{\mathbf{r}}(x)) - 1$ that coincide on $\deg(\sigma_{\mathbf{r}}(x))$ distinct points.
So

$$S_{\mathbf{r}}(x)\sigma_{\mathbf{r}}(x) \equiv \omega_{\mathbf{r}}(x) = \omega_{\mathbf{e}'}(x) \equiv S_{\mathbf{e}'}(x)\sigma_{\mathbf{e}'}(x) \equiv S_{\mathbf{e}'}(x)\sigma_{\mathbf{r}}(x) \pmod{g(x)},$$

that is,

$$(S_{\mathbf{r}}(x) - S_{\mathbf{e}'}(x))\sigma_{\mathbf{r}}(x) \equiv 0 \pmod{g(x)}.$$

The polynomials $g(x)$ and $\sigma_{\mathbf{r}}(x)$ are relatively prime, because a common factor would also divide $\omega_{\mathbf{r}}(x)$ by the key equation and this contradicts property P2. Hence, it follows that $S_{\mathbf{v}}(x) = S_{\mathbf{r}}(x) - S_{\mathbf{e}'}(x) = 0$, i.e. $\mathbf{v} \in C$. This concludes the proof that $\mathbf{r}$ is of the form as described in equality (5). It also concludes the proof of the proposition.

Proof of Proposition 2.1 (binary case):
In this case t is equal to the degree of $g(x)$. Then the crucial equation $S_{\mathbf{r}}(x)\sigma_{\mathbf{r}}(x) \equiv \omega_{\mathbf{r}}(x) \quad (\mathbf{m}odg(x))$ reduces to

$$S_{\mathbf{r}}(x)\sigma_{br}(x) = \sigma_{\mathbf{r}}(x). \tag{6}$$

Separating the squares from the non–squares in the expansion of the unknown $\sigma_{\mathbf{r}}(x)$ one can write $\sigma_{\mathbf{r}}(x) = (\alpha_{\mathbf{r}}(x))^2 + x(\mathbf{e}ta_{\mathbf{r}}(x))^2)$, one can use either Euclid's algorithm or the Berlekamp–Massey algorithm to find $\alpha_{\mathbf{r}}(x)$ and $\mathbf{e}ta_{\mathbf{r}}(x)$ and thus a $\sigma_{\mathbf{r}}(x)$ satisfying equation (6) (see for instance [18, Section 4.5]). As before, both algorithms lead to the same result (a codeword and an error vector) or they both lead to an error–message. The proof that the output $\mathbf{v}$ of $\mathcal{A}_t$ is of the form as in equation (5), with $\mathbf{v} \in \mathcal{C}$ is similar to the non–binary case.

□

References

[1] A. Barg, E. Krouk, and H.C.A. van Tilborg, *On the Complexity of Minimum Distance Decoding of Long Linear Codes*, preprint.

[2] T.A. Berson, *Failure of the McEliece Public–Key Cryptosystem Under Message–Resend and Related–Message Attack*, Advances in Cryptology – Crypto '97 Proceedings, Springer–Verlag, 1997, pp. 213–220.

[3] E.R. Berlekamp, R.J. McEliece, H.C.A. van Tilborg, *On the inherent intractability of certain coding problems*, IEEE Transactions on Information Theory, 24 (1978), pp. 384–386.

[4] E. Biham, A. Shamir *Differential Cryptanalysis of the Data Encryption Standard*, Springer–Verlag, 1993.

[5] E. Biham, A. Shamir, *Differential Fault Analysis of Secret Key Cryptosystems*, Advances in Cryptology – CRYPTO '97 Proceedings, Springer–Verlag, 1997, pp. 513–525.

[6] D. Boneh, R. DeMillo, R. Lipton, *On the importance of checking cryptographic Protocols for faults*, Advances in Cryptology – EUROCRYPT '97 Proceedings, Springer–Verlag, 1997, pp. 37–51.

[7] I. Dumer, *Suboptimal Decoding of Linear Codes: partition Technique*, IEEE Trans. on Inform. Theory, IT–42, 1996, pp. 1971–1986.

[8] Fujisaki, Okamoto, *How to Enhance the Security of Public–Key Encryption at a Minimum Cost*, PKC 1999, LNCS 1560, pp.53-68, Springer-Verlag, 1999.

[9] S. Goldwasser, S. Micali and P. Tong, *Why and How to Establish a Private Code On a Public Network*, 23rd symposium on FOCS 1982, pp 134–144

[10] C. Hall, I. Goldberg, B. Schneier, *Reaction Attacks Against Several Public–Key Cryptosystems*, Proceedings of Information and Communication Security, ICICS'99, Springer-Verlag, 1999, pp. 2-12.

[11] T. Kasami, T. Fujiwara, and Shu Lin, *An Approximation to the Weight Distribution of Binary Linear Codes*, IEEE Trans. on Inform. Theory, Vol.IT–31, No. 6, 1985, pp.769–780.

[12] I. Krasikov and S. Litsyn, *On the Accuracy of the Binomial Approximation to the Distance Distance Distribution of Codes*, IEEE Trans. on Inform. Theory Vol.IT–41, 1995, pp. 1472–1475.

[13] P.C. Kocher, *Timing Attacks on Implementations of Diffie–Hellman, RSA, DSS, and Other Systems*, Advances in Cryptology – Crypto '96 Proceedings, Springer–Verlag, 1996, pp. 104–113.

[14] J.H. van Lint, *Introduction to Coding Theory*, Springer–Verlag, 1982.

[15] M. Matsui, *Linear Cryptanalysis Method for DES Cipher*, Advances in Cryptology – EUROCRYPT '93 Proceedings, Springer–Verlag, 1994, pp. 74–87.

[16] R.J. McEliece, *A public–key cryptosystem based on algebraic coding theory*, DSN Progress Report 42–44, Jet Propulsion Laboratory, Pasadena, 1978.

[17] R.J. McEliece, *The theory of information and coding,* Encyclopedia of Math. and its Applications, Vol. 3, Addison–Wesley Publishing Company, Reading, Mass., 1977.

[18] H.C.A van Tilborg, *Error–correcting Codes – a first introduction*, Chartwell Bratt Ltd, 1993.

[19] E.R. Verheul, *Sloppy Alice Attacks! Differential Fault Attacks on ElGamal, Chor–Rivest and Mandatory LEAF–reconstruction Recovery Systems*, in preparation.