Improvements of the Attacks on Cryptosystems Based on Error-correcting Codes

Anne CANTEAUT
Florent CHABAUD

LIENS - 95 - 21

Improvements of the Attacks on Cryptosystems Based on Error-correcting Codes

Anne CANTEAUT*
Florent CHABAUD

LIENS - 95 - 21

July 1995

Laboratoire d'Informatique de l'Ecole Normale Supérieure
45 rue d'Ulm 75230 PARIS Cedex 05

Tel : (33)(1) 44 32 00 00
Adresse électronique : ...@dmi.ens.fr

*INRIA Projet Codes
Domaine de Voluceau, 78153 Le Chesnay

Improvements of the attacks on cryptosystems based on error-correcting codes

Anne Canteaut [*]
Anne.Canteaut@inria.fr

INRIA Projet Codes
Domaine de Voluceau
F-78153 Le Chesnay Cedex

Florent Chabaud [**]
Florent.Chabaud@ens.fr

LIENS [***]
45, rue d'Ulm
F-75230 Paris Cedex 05

Abstract. Many public-key cryptosystems and identification schemes based on error-correcting codes have been proposed as an alternative to the common cryptographic algorithms based on number theory. They rely on the NP-hardness of finding a fixed-weight word in a coset of a linear binary code. We here improve the previous attacks on these systems; this notably enables us to reduce the work factor involved in breaking McEliece's cryptosystem since our algorithm requires $2^{64.2}$ operations that is 2^7 times less than Lee-Brickell's attack.

KEYWORDS: Error-correcting codes, Minimum weight codewords, Markov chains, McEliece's cryptosystem, Cryptanalysis.

1 Introduction

Since the concept of public-key cryptography appeared in 1977, searching for secure public-key cryptosystems and identification schemes has been one of the most active areas in the field of cryptology. Seventeen years after the fundamental paper of Diffie and Hellman, public-key cryptography has however become dangerously dependent on the difficulty of two problems: integer factoring and discrete logarithm. Studying other hard problems which could be applied to public-key cryptography seems therefore essential in order to anticipate an important progress in factoring methods. The class of public-key cryptosystems or identification schemes based on algebraic error-correcting codes is an alternative to the common algorithms based on number theory. It relies on the NP-hardness of finding a codeword of a given weight in a linear binary code. Each algorithm which is able to decode any linear code, or equivalently to find short codewords, is consequently an attack against these systems. Improving this type of algorithms is then interesting in order to see whether the existing systems like McEliece's

[*] Also with École Nationale Supérieure de Techniques Avancées, laboratoire LEI, 32 boulevard Victor, F-75015 Paris.
[**] On leave from Centre d'Électronique de l'Armement, F-35174 Bruz.
[***] Supported by the Centre National de la Recherche Scientifique URA 1327.

cryptosystem resist such attacks, and also in order to delimit the parameters which could be insecure.

We carry out in this purpose two kinds of improvements to the existing algorithms for finding short codewords in any linear code. On one hand a very precise analysis of their complexity shows that their performances strongly depend on some parameters; a more general approach and the optimization of these parameters enable us to sizeable reduce their running-time.

On the other hand, as we have noticed that all these algorithms make many independent Gaussian eliminations, we propose replacing this time-consuming procedure by a faster iterative one. Using Markov chains theory we give the average number of operations required by the modified algorithms; that leads us to a new optimization of the parameters which is in accordance with the simulations we made. Our successive improvements notably enable us to attack McEliece's cryptosystem with $2^{64.2}$ elementary operations, that is 2^7 times less than the previous best known attack [9]. Although this work factor is still too high, that makes it possible to determine the insecure parameters of the codes used for cryptographic purposes; for instance, our algorithm is able to decode any $[512, 256]$-random linear binary code up to half its minimum distance.

The first part of this paper presents some well-known cryptographic systems relying on error-correcting codes; the following one describes the classical algorithms to find short codewords in a linear code. Part III shows how they can be improved by optimizing their parameters. The general improvement of all the algorithms and the corresponding complexity are exposed in part IV. The last part then applies these new attacks to the cryptosystems defined in part I.

PartI

Some systems based on error-correcting codes

We here describe three well-known cryptographic systems based on error-correcting codes: McEliece's and Niederreiter's cryptosystems and Stern's identification scheme.

Notations As usual in coding theory we write the vectors in $GF(2)^n$ as words $x = x_1 \dots x_n$ over $GF(2)$. The transpose of matrix A (resp. vector x) is denoted by A^t (resp. x^t).

2 McEliece's public-key cryptosystem

This cryptosystem is one of the first that has used error-correcting codes. It employs an $[n, k]$-linear code over $GF(2)$, $\mathcal{C}$, with an error-correcting capability of t errors, for which an efficient decoding algorithm is known. The original description of McEliece [13] uses Goppa codes but the larger class of alternant codes is also convenient.

- **private key:** it consists of the three matrices G, S and P, where G is an (n, k) generator matrix of this code, S is an arbitrary (k, k) invertible matrix and P is an arbitrary (n, n) permutation matrix.
- **public key:** it is composed by the (n, k) matrix G' defined by $G' = SGP$ and the error-correcting capability, t. G' is then a generator matrix for an equivalent $[n, k]$-linear code for which no decoding algorithm is known.
- **encryption:** the cipher-text corresponding to a k-bit message vector m is $y = mG' + e$, where e is an n-bit error of weight at most t which is randomly chosen by the sender.
- **decryption:** the decryption procedure consists in computing $yP^{-1} = mSG + eP^{-1}$ and using a fast decoding algorithm for $\mathcal{C}$ to recover mS. The message is then found by $m = (mS)S^{-1}$.

The parameters originally suggested by McEliece are: $n = 1024$, $k = 524$ and $t = 50$.

3 Niederreiter's public-key cryptosystem

This is a knapsack-type cryptosystem which uses an $[n, k]$-linear code over $GF(q)$, $\mathcal{C}$, with an error-correcting capability of t errors and an efficient decoding algorithm [14].

- **private key:** it consists of the three matrices H, M and P, where H is an $(n-k, n)$ parity check matrix of $\mathcal{C}$, M is an arbitrary $(n-k, n-k)$ invertible matrix and P is an arbitrary (n, n) permutation matrix.
- **public key:** it is composed by the $(n-k, n)$ matrix H' defined by $H' = MHP$.
- **encryption:** the cipher-text corresponding to an n-bit message vector e of weight less than or equal to t is the syndrome $s = eH'^t$.
- **decryption:** the decryption procedure consists in computing $s(M^{-1})^t = eP^tH^t$ and using a fast decoding algorithm for $\mathcal{C}$ to recover eP^t. The message is then found by $e = (eP^t)(P^{-1})^t$.

This cryptosystem is said to be of knapsack-type because the encryption consists in picking t columns from H' and computing a weighted sum of these vectors.

It is shown in [11] that McEliece's and Niederreiter's cryptosystems are equivalent from the security point of view when set up for corresponding choices of parameters.

Niederreiter initially mentioned two example systems: one using the [104,24,32] binary concatenated code obtained by concatenation of the [8,4,4] binary extended Hamming code with a [13,6] punctured Reed-Solomon code over $GF(16)$ of minimum distance 8, and the other using a [30,12,19] Reed-Solomon code over $GF(31)$. Both of these examples have been proved insecure by Brickell and Odlyzko [3].

Note 1. McEliece's and Niederreiter's cryptosystems therefore rely on the following principle: the secret key is a code $\mathcal{C}$ which is easy to decode, and the public key is a generator or parity-check matrix of a permutation equivalent code. The invertible matrix S has no cryptographic function; it only assures for McEliece's system that the public matrix is not systematic otherwise most of the bits of the plain-text would be revealed.

Then an eligible class of codes for these systems has to verify the following properties:

1. for given length, dimension and minimal distance this class is large enough to avoid any enumeration.
2. an efficient decoding algorithm is known for this class.
3. a generator or parity-check matrix of a permutation equivalent code gives no information about the structure of the secret code, that means the decoding algorithm requires some parameters of the code besides a generator matrix G'.

For example generalized Reed-Solomon codes are not convenient because their structure can be recovered using Sidelnikov-Shestakov algorithm [17]; concatened codes which were initially suggested by Niederreiter are not appropriated either [16]. But the class of Goppa codes is well suited to such systems insofar as there actually exists no algorithm which is able to compute the characteristic parameters of a Goppa code from one of its generator matrix.

In the case the used class of codes verifies the above properties the equivalent code defined by the public key presents no particular structure; recovering the plain-text from a cipher-text then amounts to decoding any linear code.

4 Stern's public-key identification scheme

Stern has presented at Crypto'93 [19] a public-key identification scheme which relies on the difficulty of finding a small-weight codeword of given syndrome.

This scheme uses an $[n, k]$-random linear code over $GF(2)$. All users share a fixed parity check matrix H for this code and an integer w slightly below the expected value for the minimal distance of a random linear code.

Each user receives a secret key s which is an n-bit vector of weight w. His public key is then the syndrome sH^t. Any user can identify himself to another one by proving he knows s without revealing it thanks to an interactive zero-knowledge protocol.

Minimal parameters proposed by Stern are $n = 512$, $k = 256$ and $w = 56$.

PartII

Classical attack algorithms

5 Principle

These three cryptographic systems rely on the same problem: recovering the plain-text from the cipher-text (or finding a secret key from the public one for Stern's identification scheme) is equivalent to finding a fixed-weight word in a coset of a linear code $\mathcal{C}$, where the only information we have about $\mathcal{C}$ is one of its generator or parity-check matrices.

For McEliece's cryptosystem this clearly corresponds to the general problem of minimal distance decoding which is an NP-complete problem [2]: recovering the plain-text is equivalent to recovering the error-vector e which is the shortest word in the coset $\mathcal{C}(y)$ containing the cipher-text.

We come down to the same problem for both Niederreiter's and Stern's systems: it is sufficient to apply a Gaussian elimination on the public parity-check matrix in order to obtain a standard-form matrix $BH = (Z|I_{n-k})$; finding a word of weight w and given syndrome s then amounts to finding a word of weight w in the coset containing the n-bit vector $y = (0|sB^t)$ since the syndrome of y is s.

Of course, every algorithm that can find the shortest word of a linear code is also a potential general algorithm to decode, as the matrix

$$\Gamma = \left(\begin{array}{c} \hat{G} \\ \hline m\hat{G} + e \end{array} \right)$$

is a generator matrix of an $[n, k+1, w]$-linear code of shortest codeword e.

6 General minimal distance decoding algorithms

from now on we consider a linear code $\mathcal{C}$ over $GF(2)$ of length n and dimension k, a generator matrix G for this code and an n-bit vector y. We try then to recover a word e of weight w in the coset $\mathcal{C}(y)$.

6.1 McEliece's algorithm

This attack was evoked by McEliece himself in his original paper [13]. One iteration of the algorithm is as follows:

1. Randomly permute the columns of the generator matrix.
2. Apply a Gaussian elimination on the rows of the matrix to obtain the form $\tilde{G} = (\ I_k \quad | \quad A\ )$, where the corresponding permuted cipher text is denoted by $y = (\ c_1 + e_1 \quad | \quad c_2 + e_2\ )$.

3. Guess that the error e_1 is null by checking whether the weight of error e_2 is w. In this case $y + c_1\tilde{G}$ is in coset $\mathcal{C}(y)$ and its weight is w.

Note 2. C. Adams and H. Meijer have shown [1] that the best parameters for McEliece's cryptosystem to counter this attack are

$$n = 1024,\ k = 654,\ t = 37$$

6.2 Lee-Brickell's improvement

Lee and Brickell have pointed out that the most time-consuming procedure in the previous attack was the Gaussian elimination whereas the cost of the verification was negligible. Hence they have modified it in order to reduce the number of Gaussian elimination by raising the cost of step 3. In their attack, the last step consists in checking whether the weight of the partial error e_1 is less than or equal to a fixed parameter p. The original McEliece's algorithm corresponds to $p = 0$ and they have heuristically proposed the value $p = 2$ to minimize the running-time of their algorithm [9].

6.3 Leon's algorithm

The algorithm proposed by J.S. Leon [10] is a probabilistic algorithm that tries to find short codewords. It introduces two parameters σ and p. This algorithm uses a generator matrix and it searches the zero-bits of minimum weight words. More precisely each iteration runs as follows:

1. Choose a random selection S of $k + \sigma$ columns of the matrix, which are put by permutations to the right end of the matrix.
2. Apply a Gaussian elimination so that the resulting matrix has the form

$$\tilde{G} = \left(\begin{array}{c|c|c} B & Z & I_e \\ \hline D & 0 & 0 \end{array}\right),$$

   where B is an $(n - k - \sigma, e)$ matrix, Z is a $(k + \sigma - e, e)$ matrix and D is an $(n - k - \sigma, k - e)$ one for some $e \leq k + \sigma$.
3. Look for the linear combinations that lead to codewords m such that $w(m|_S) \leq p$. This can be achieved by considering the single matrix Z. In the case of $w(m|_S) \leq p$, compute the corresponding n-bit word and verify if its weight is less than the weight of the previously shortest obtained word. When $e \neq 0$, this test must also be performed for the codewords that include D-codewords.

The parameters generally used for this algorithm [10, 6] are $p = 2$, $\sigma = 2$.

6.4 Stern's algorithm

This third algorithm [18], obtained independently from the above one, attacks the code by using a parity-check matrix H. It introduces two parameters p and ℓ.

1. The first step of the algorithm is similar to Leon's algorithm: choose a random selection S of $n-k$ columns of the matrix, which are put by permutations to the right end of the matrix. The degenerative cases are eliminated in order to obtain a matrix of the form:
$$\tilde{H}_1 = (\, Q \quad | \quad I_{n-k}\,)\,.$$
2. Randomly split the columns of matrix Q in two subsets in order to obtain, after permutations, a matrix of the form:
$$\tilde{H}_2 = (\, X \quad | \quad Y \quad | \quad I_{n-k}\,)\,.$$
3. Randomly choose ℓ rows of the matrix and perform permutations on rows in order to obtain a matrix of the form:
$$\tilde{H}_3 = \left(\begin{array}{c|c|c} X_\ell & Y_\ell & \\ \hline X_{n-k-\ell} & Y_{n-k-\ell} & \end{array} J \right).$$
4. For each group P_X of p columns of X_ℓ, compute their sum $\sigma_\ell(P_X)$, and do the same for each P_Y. If $\sigma_\ell(P_X) = \sigma_\ell(P_Y)$, select these $2p$ columns $P_X \cup P_Y$ and compute the sum V of the $n-k-\ell$ other rows of these columns. If the weight of V is $w-2p$, then it is possible to build a codeword of weight w.

The parameters proposed in [6] for this algorithm are $p=2$ or $p=3$ according to the available memory, and $\ell = \ln k$.

7 Efficiency of these algorithms

Table 1 gives the work factor of each of these algorithms (*i.e.* the average number of elementary operations required to perform it (see section 8) concerning three different cryptosystems:

1. Stern's public key identification scheme, *i.e.* searching for a word of weight 56 in a $[512, 256]$ code. The work factors for both Leon's and Stern's algorithms are therefore given for a $[512, 257]$ code.
2. McEliece's cryptosystem with its original parameters: $n = 1024$, $k = 524$ and $w = 50$. The work factors for both Leon's and Stern's algorithms are therefore given for a $[1024, 525]$ code.
3. McEliece's cryptosystem with the Adams and Meijer's parameters: $n = 1024$, $k = 654$ and $w = 37$. The work factors for both Leon's and Stern's algorithms are therefore given for a $[1024, 655]$ code.

Cryptosystem	Attacks							
	McEliece	Lee		Leon		Stern		
Stern	$2^{84.9}$	$p=2$	$2^{74.9}$	$p=2$, $\sigma=2$	$2^{74.9}$	$p=2$, $\ell=8$	$2^{73.5}$	
McEliece	$2^{80.7}$	$p=2$	$2^{71.3}$	$p=2$, $\sigma=2$	$2^{70.7}$	$p=2$, $\ell=9$	$2^{69.9}$	
Adams	$2^{83.8}$	$p=2$	$2^{73.5}$	$p=2$, $\sigma=2$	$2^{73.1}$	$p=2$, $\ell=9$	$2^{71.9}$	

Table 1. Work factors of classical attacks

These results are obtained from the formulas of appendix A.

PartIII
Improvement of the attacks

8 Generalization

8.1 A more general algorithm

Description: This algorithm was first studied in [7] as a variant of Stern's algorithm. We here present this algorithm in its generator matrix form, and we try to find a short codeword. It introduces three parameters p, σ and s.

Algorithm *A*: At each iteration:

1. Randomly choose a set of k columns and perform a Gaussian elimination to obtain a matrix in standard form.
2. Randomly select σ other columns of the matrix to obtain
$$\tilde{G} = (B \quad | \quad Z \quad | \quad I_k),$$
where B is an $(n-k-\sigma, k)$ matrix and Z is a (σ, k) matrix.
3. Look for all linear combinations of at most p rows of Z that give words of weight less than or equal to s on the $k+\sigma$ selected columns. Then compute the corresponding whole codewords and check their weights.

Note 3. Lee-Brickell's algorithm with parameter p then corresponds to $\sigma = 0$ and $s = p$ and Leon's algorithm corresponds to $s = p$.

8.2 Work factor

We call *work factor* of an algorithm the average number of elementary operations required to perform it and $W_{Algo(params)}(n,k,w)$ will denote it for an algorithm *Algo* – depending on its parameters – when it is applied to a code of length n, dimension k and minimal weight w.

We will denote $\bar{N}_{Algo(params)}(n,k,w)$ the average number of iterations of the algorithm *Algo*.

All the previous algorithms apply a Gaussian elimination on a generator (or parity-check) matrix for the code in order to obtain a systematic matrix $(I|Z)$, and they try then to find short words in a few operations on the reduced matrix Z. From now on $\mathcal{G}(n,\kappa)$ denotes the average number of operations to perform a Gaussian elimination on the rows of an (n,κ)-matrix of rang κ, and $J_{Algo(*)}(n,k,w)$ denotes the average number of additional operations performed at each iteration by the algorithm. Hence we obviously have a relation of the form

$$W = \bar{N} \times (\mathcal{G} + J)\,. \tag{1}$$

8.3 Variants of these algorithms

Dual versions

Principle: For all those algorithms, it is possible to construct a version of same success probability, but working on the dual code.

Example: If we now consider algorithm A, it deals with a generator matrix of the code. We can easily obtain a corresponding parity-check matrix. Let us consider the algorithm where each iteration is as follows:

1. Randomly choose $(n-k)$ columns and performs a Gaussian elimination.
2. Randomly choose σ rows of the matrix

   $$\tilde{H} = \left(\frac{Z}{B} \middle| \, J \right) .$$

   where B is a $(n-k-\sigma,k)$ matrix, Z is a (σ,k) matrix and J is the resulting permutation of the identity I_{n-k}.
3. Consider all the linear combinations of at most p columns of Z, and when the corresponding weight is less than or equal to s, compute the whole codeword of $\tilde{H}$.

In the same way Stern's algorithm can be modified for working on a generator matrix.

The success probability of the dual version of each algorithm is obviously the same as for the original version. An optimization can therefore be done according to the expansion rate of the code, since $\mathcal{G}(n,k)$ is smaller than $\mathcal{G}(n,n-k)$ if the rate n/k is smaller than $1/2$.

Decoding or finding short codewords ? All these algorithms have been described from a single point of view. Leon's and Stern's algorithms were first designed to find short codewords. On the other hand Lee-Brickell's algorithm tried to decode general linear codes. But all these algorithms can be modified to perform the other point of view.

As said in section 5, decoding a word y for an $[n, k]$-code with an error-correcting capability w is equivalent to searching for a word of weight w in an $[n, k+1]$-code if we consider the new code defined by the generator matrix

$$\left(\frac{G}{y}\right)$$

For instance, Stern's algorithm can be used almost directly for syndrome decoding and algorithm A with $\sigma = 0$ and $s = p$ exactly corresponds to Lee-Brickell's algorithm for finding short codewords.

Work factor Combining these two variants – Generator or Parity-check matrix and Decoding or Short codewords – we obtain four different versions for each algorithm. The D-type algorithms aim at decoding an $[n, k]$-code with error-correcting capability w whereas the S-type versions aim at finding a word of weight w in an $[n, k]$-code. We then have the following corresponding forms of equation 1:

$$W_{Algo_{GS}(*)}(n,k,w) = \bar{N}_{Algo_S(*)}(n,k,w) \times \left(\mathcal{G}(n,k) + J_{Algo_{GS}(*)}(n,k)\right)$$
$$W_{Algo_{PS}(*)}(n,k,w) = \bar{N}_{Algo_S(*)}(n,k,w) \times \left(\mathcal{G}(n,n-k) + J_{Algo_{PS}(*)}(n,k)\right)$$
$$W_{Algo_{GD}(*)}(n,k,w) = \bar{N}_{Algo_D(*)}(n,k,w) \times \left(\mathcal{G}(n,k) + \mathcal{D}_G(n,k) + J_{Algo_{GD}(*)}(n,k)\right)$$
$$W_{Algo_{PD}(*)}(n,k,w) = \bar{N}_{Algo_D(*)}(n,k,w) \times \left(\mathcal{G}(n,n-k) + \mathcal{D}_P(n,k) + J_{Algo_{PD}(*)}(n,k)\right)$$

The $\mathcal{D}$ expressions are correcting terms which take in account the extra operations performed during the Gaussian elimination on the codeword (type G) or the syndrome (type P) — see below.

9 Optimization of the parameters

9.1 Principle

The above algorithms use some parameters. Their influence on the performance of the algorithms is very important. A first approach to determine the optimal parameters of some of these algorithms was given in [6]. Unfortunately, this kind of asymptotic analysis is not optimal for the useful dimensions. A more practical approach was given in [7] and consists in a precise estimation of the work factor of these algorithms.

Since all these algorithms have independent iterations, according to note 13, we have

$$\bar{N}_{Algo(*)}(n,k,w) = \frac{1}{\pi_{Algo(*)}(n,k,w)}. \tag{2}$$

In appendix A and B we give for each algorithm:

- its type,
- its success probability $\pi_{Algo(*)}(n, k, w)$,
- its average number of operations by iteration $J_{Algo(*)}(n, k)$,

so that the resulting work factor can be obtained by the formulas given in section 8.3 assuming that a Gaussian elimination on an (n, κ)-matrix performs about $\kappa \times \kappa/2$ additions of n-bit words, that means:

$$\mathcal{G}(n, \kappa) = n \times \kappa \times \kappa/2. \tag{3}$$

In the same way we also have:

$$\mathcal{D}_G(n, k) = n \times k/2,$$
$$\mathcal{D}_P(n, k) = (n - k) \times (n - k)/2.$$

9.2 Results

The explicit formulas (see appendix A and B) of the work factors can be used to find the optimal parameters of the attacks. For the cryptanalysis of the previously described cryptosystems, we have to compare the following work factors:

$W_{Algo_{GD}(*)}(n, k, w), W_{Algo_{PD}(*)}(n, k, w), W_{Algo_{GS}(*)}(n, k{+}1, w), W_{Algo_{PS}(*)}(n, k{+}1, w).$

This first improvement is shown in table 2.

PartIV
An iterative algorithm

As shown in part III each iteration of any of these algorithms first performs a Gaussian elimination on a generator (or parity-check) matrix for the code.

We now propose an iterative algorithm which avoids this time-consuming procedure.

10 Principle

from now on we will use the following notations:

Let $N = \{1, \cdots, n\}$. For any subset I of N, $G = (V, W)_I$ denotes the decomposition of a generator matrix G of a code of length n onto I, that means $V = (G_i)_{i \in I}$ and $W = (G_j)_{j \in N \setminus I}$, where G_i is the ith column of matrix G.

Definition 4. Let I be a k-element subset of N. I is an information window for G iff $M = (V, W)_I$ where V is invertible. The complementary set, $J = N \setminus I$, is called a redundancy window.

Cryptosystem	Attacks	
	Lee	Leon
Stern	$p = 2$ $2^{74.9}$	$p = 3$, $\sigma = 5$ $2^{73.9}$
McEliece	$p = 2$ $2^{71.3}$	$p = 3$, $\sigma = 7$ $2^{69.5}$
Adams	$p = 2$ $2^{73.5}$	$p = 3$, $\sigma = 7$ $2^{71.9}$

Cryptosystem	Attacks			
	Algo. A_{GD}	Algo. A_{PD}	Algo. A_{GS}	Algo. A_{PS}
Stern	$p = 3$, $\sigma = 17$, $s = 7$ $2^{73.5}$	$p = 3$, $\sigma = 17$, $s = 7$ $2^{73.5}$	$p = 3$, $\sigma = 17$, $s = 7$ $2^{73.5}$	$p = 3$, $\sigma = 17$, $s = 7$ $2^{73.5}$
McEliece	$p = 3$, $\sigma = 11$, $s = 4$ $2^{69.6}$	$p = 3$, $\sigma = 11$, $s = 4$ $2^{69.5}$	$p = 3$, $\sigma = 11$, $s = 4$ $2^{69.3}$	$p = 3$, $\sigma = 11$, $s = 4$ $2^{69.3}$
Adams	$p = 3$, $\sigma = 11$, $s = 4$ $2^{71.9}$	$p = 2$, $\sigma = 19$, $s = 6$ $2^{71.1}$	$p = 3$, $\sigma = 11$, $s = 4$ $2^{71.7}$	$p = 2$, $\sigma = 19$, $s = 6$ $2^{71.2}$

Cryptosystem	Attacks			
	$Stern_{GD}$	$Stern_{PD}$	$Stern_{GS}$	$Stern_{PS}$
Stern	$p = 2$, $\ell = 13$ $2^{71.6}$	$p = 2$, $\ell = 13$ $2^{71.6}$	$p = 2$, $\ell = 13$ $2^{71.8}$	$p = 2$, $\ell = 13$ $2^{71.8}$
McEliece	$p = 2$, $\ell = 17$ $2^{66.2}$	$p = 2$, $\ell = 17$ $2^{66.0}$	$p = 2$, $\ell = 16$ $2^{66.2}$	$p = 2$, $\ell = 16$ $2^{66.1}$
Adams	$p = 2$, $\ell = 17$ $2^{68.0}$	$p = 2$, $\ell = 18$ $2^{66.8}$	$p = 2$, $\ell = 17$ $2^{68.1}$	$p = 2$, $\ell = 18$ $2^{66.9}$

Table 2. Work factors of optimized classical attacks

All the previously described algorithms aim at exploring the set of information windows until the initial word will be found. They first randomly select an information window and then they try to find short words with a few operations on the redundancy window.

The commonly used method for exploring this set consists in randomly selecting at each step a new information window, I', which is completely independent from the previous ones. We here propose to choose I' by modifying only one element of the previous information window like the simplex method as proposed by [15]. This idea was first exposed in [20, 5] concerning respectively McEliece's and Lee-Brickell's algorithm.

Definition 5. Two information windows I and I' are close iff:

$$\exists q \in I,\ \exists p \in N \setminus I,\ \text{such that } I' = (I \setminus \{q\}) \cup \{p\}$$

As any two information windows can be joined by a sequence of close information windows, we will use this iterative method in order to find one of them which enables us to recover the initial word.

The following proposition shows how choosing q and p such that I' is still an information window.

Proposition 6. *Let I be an information window such that $M = (V, W)_I$*
Let be $q \in I$, $p \in J$ and $I' = (I \setminus \{q\}) \cup \{p\}$.
I' is an information window iff $z_{q,p} = 1$, where $Z = V^{-1}W = (z_{i,j})_{i \in I, j \in J}$

Proof. Z is defined by $VZ = W$, thus we have:

$$\forall j \in J,\ G_j = \sum_{i \in I} z_{i,j} G_i$$

This relation holds for index p: $G_p = z_{q,p} G_q + \sum_{i \in I \setminus \{q\}} z_{i,p} G_i$
As the columns indexed by I are linearly independent, we have:
G_p and $(G_i)_{i \in I \setminus \{q\}}$ are linearly independent iff $z_{q,p} = 1$.

From now on we index rows of Z with I using $\tilde{G} = (I_k \mid Z)_I$ and we denote by Z^i the ith row of Z.

As the studied algorithms deal with the so-called standard-form matrix $V^{-1}G = (I_k \mid Z)_I$ corresponding to I, we only need a procedure able to obtain the standard form generator matrix $V'^{-1}G = (I_k \mid Z')_{I'}$ corresponding to I' from the previous one, $V^{-1}G$. This systematic matrix can be updated in the following way.

Proposition 7. *Let I and I' be two close information windows such that*

$$I' = (I \setminus \{q\}) \cup \{p\}.$$

Let $V^{-1}G = (I_k \mid Z)_I$ and $V'^{-1}G = (I_k \mid Z')_{I'}$ be the corresponding standard-form matrices. Then Z' is obtained from Z by:

- $Z'^p = Z^q$
- $\forall i \in I' \setminus \{p\}, \forall j \in N \setminus (I' \cup \{q\}),\ \ z'_{i,j} = z_{i,j} + z_{i,p} z_{q,j}$ *and* $\forall i \in I' \setminus \{p\}\ \ z'_{i,q} = z_{i,p}$

Proof. As $I' = (I \setminus \{q\}) \cup \{p\}$, $V'^{-1}G$ is obtained by exchanging the qth and pth columns of $V^{-1}G$. This can be done by a simple pivoting operation in position (q, p), *i.e.* by adding the qth row of matrix $V^{-1}G$ to all others rows $(V^{-1}G)^i$ iff the corresponding element $z_{i,p}$ is not equal to 0 (see figure 1).

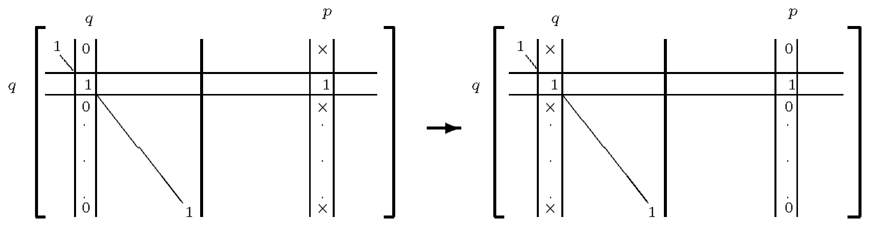

Fig. 1. Close information windows

11 Modified algorithms

We now describe the previous algorithms in which the Gaussian elimination has been replaced by this faster procedure. We only present for both algorithm A and Stern's algorithm the GS-type versions.

11.1 Algorithm A_{GSM}

Randomly choose an information window I and apply a Gaussian elimination in order to obtain a systematic generator matrix $\tilde{G} = (\, I_k \quad | \quad Z\,)$.

Then each iteration runs as follows:

1. Randomly choose a σ-element subset Σ of $N \setminus I$.
2. For all $1 \leq i \leq p$ and for each linear combination λ of i rows of matrix G do:
   - compute $w(\lambda_{|\Sigma})$.
   - if $w(\lambda_{|\Sigma}) \leq s - i$, then compute $w(\lambda_{|N\setminus(I\cup\Sigma)})$.
     if $w(\lambda_{|N\setminus(I\cup\Sigma)}) + w(\lambda_{|\Sigma}) \leq w - i$ then return λ.
3. Randomly choose $q \in I$ and $p \in N \setminus I$.
   Replace information window I with $(I \setminus \{q\}) \cup \{p\}$ by updating matrix Z according to the preceding proposition.

11.2 Stern's GSM algorithm

Randomly choose an information window I and apply a Gaussian elimination in order to obtain a systematic generator matrix $\tilde{G} = (\, I_k \quad | \quad Z\,)$.

Then each iteration runs as follows:

1. Randomly split information window I in two subsets I_1 and I_2 where $|I_1| = \lfloor k/2 \rfloor$ and $|I_2| = \lceil k/2 \rceil$. We now obtain
$$\tilde{G}_1 = \left(J \left| \frac{Z_1}{Z_2} \right. \right)$$
2. Randomly choose an ℓ-element subset L of $N \setminus I$.
3. - For each linear combination λ_1 of p rows of matrix Z_1, compute $w(\lambda_{1|L})$.
   - For each linear combination λ_2 of p rows of matrix Z_2, compute $w(\lambda_{2|L})$.

   - If $\lambda_{1|L} = \lambda_{2|L}$, then compute $w((\lambda_1+\lambda_2)_{|N\setminus(I\cup L)})$.
     If this weight is less than or equal to $w-2p$ then return$(\lambda_1+\lambda_2)$.

4. Randomly choose $q \in I$ and $p \in N \setminus I$.
   Replace information window I with $(I\setminus\{q\})\cup\{p\}$ by updating matrix Z according to the preceding proposition.

12 Theoretical running-time

12.1 Where does the improvement come from?

When the new information window was chosen independently from the previous ones, the work factor involved in putting the matrix in standard form was given by equation 3. As this new information window is now close to the previous one, the same operation can be done by adding the row Z^q to rows Z^i whose element $z_{i,p}$ is non-zero. Assuming that the average weight of Z_p is $\frac{\kappa}{2}$, the work factor involved in this procedure is

$$g(n,\kappa) = \kappa \times (n-\kappa) \times \frac{1}{2}.$$

In the same way, for the D-type algorithms, the correction term becomes for both types P and G:

$$d(n,k) = (n-k)/2.$$

While the number of operations required by each iteration slightly decreases, the number of iterations for these new algorithms is now higher because the successive information windows are not independent any more. Hence, $\bar{N}_{Algo_M(*)}$ is not equal to $\bar{N}_{Algo(*)}$ since the estimation 2 resulted from this independence.

Therefore the formulas of section 8.3 must be changed accordingly.

$$W_{Algo_{GSM}(*)}(n,k,w) = \bar{N}_{Algo_{SM}(*)}(n,k,w) \times \big(g(n,k) + J_{Algo_{GS}(*)}(n,k)\big)$$
$$W_{Algo_{PSM}(*)}(n,k,w) = \bar{N}_{Algo_{SM}(*)}(n,k,w) \times \big(g(n,n-k) + J_{Algo_{PS}(*)}(n,k)\big)$$
$$W_{Algo_{GDM}(*)}(n,k,w) = \bar{N}_{Algo_{DM}(*)}(n,k,w) \times \big(g(n,k) + d(n,k) + J_{Algo_{GD}(*)}(n,k)\big)$$
$$W_{Algo_{PDM}(*)}(n,k,w) = \bar{N}_{Algo_{DM}(*)}(n,k,w) \times \big(g(n,n-k) + d(n,k) + J_{Algo_{PD}(*)}(n,k)\big)$$

12.2 Average number of iterations

The previous modified algorithm can be associated with a discrete-time stochastic process $\{X_i\}_{i\in\mathbb{N}}$ where the random variable X_i represents the outcome of the ith iteration of the algorithm. For all considered algorithms, X_i then corresponds to the distribution of positions of $supp(e)$ among the windows splitting N at step i. For example, if we consider Lee-Brickell's algorithm, X_i is the number of positions of $supp(e)$ contained by the information window at step i.

We note that the state space $\mathcal{E}$ is finite

The ith iteration of the algorithm is then described by the probability distribution π_i:

$$\forall u \in \mathcal{E},\ \pi_i(u) = Pr[X_i = u],$$

and we have $\sum_{u\in\mathcal{E}} \pi_i(u) = 1$.

Definition 8. The stochastic process $\{X_i\}_{i\in I\!N}$ is a Markov chain iff for all i and for all $(u_0, u_1, \cdots, u_i) \in \mathcal{E}$,

$$Pr[X_i = u_i/X_{i-1} = u_{i-1},\ X_{i-2} = u_{i-2}, \cdots, X_0 = u_0] = Pr[X_i = u_i/X_{i-1} = u_{i-1}].$$

Definition 9. Every matrix $(P_{u,v})_{(u,v)\in\mathcal{E}^2}$ of real elements such that

$$P_{u,v} \geq 0$$
$$\sum_{v\in\mathcal{E}} P_{u,v} = 1 \text{ for all } u \in \mathcal{E}$$

is called Markovian.

Definition 10. A Markov chain $\{X_i\}_{i\in I\!N}$ is said to be stationary (or homogeneous in time) iff there exists a Markovian matrix P such that for all i and $\forall(u, v) \in \mathcal{E}$,

$$Pr[X_i = u/X_{i-1} = v] = P_{v,u}.$$

A stationary Markov chain $\{X_i\}_{i\in I\!N}$ is then completely determined by its starting distribution π_0 and by its transition matrix P.

The state space can be split in two subsets $\mathcal{S}$ and $\mathcal{F}$ where $\mathcal{S}$ contains the recurrent states and $\mathcal{F}$ the transient states. We now assume that $\{X_i\}_{i\in I\!N}$ is a transient chain, that means that each recurrent state is a success state or equivalently that it is an absorbing state; then $\mathcal{F}$ corresponds to the failure states.

Proposition 11 [8]. *If $\{X_i\}_{i\in I\!N}$ is a transient chain with transition matrix P, and Q is the sub-stochastic matrix corresponding to transitions among the transient states, i.e. $Q = (P_{u,v})_{\substack{u\,\in\,\mathcal{F}\\ v\,\in\,\mathcal{F}}}$ then $(Id - Q)$ has an inverse R called the fundamental matrix of the chain and*

$$R = \sum_{m=0}^{\infty} Q^m = (Id - Q)^{-1}.$$

Proposition 12. *The expectation $\bar{N}$ of the number of iterations required until X_n reaches a success state is given by:*

$$\bar{N} = \sum_{u\in\mathcal{F}} \pi_0(u) \sum_{v\in\mathcal{F}} R_{u,v}$$

where R is the corresponding fundamental matrix.

Proof.

$$\begin{aligned}\bar{N} &= \sum_{n\geq 0} nPr[N = n]\\ &= \sum_{n\geq 1} Pr[N \geq n]\\ &= \sum_{n\geq 0} Pr[X_n \in \mathcal{F}].\end{aligned}$$

As the initial probability distribution is vector π_0, we have:

$$\bar{N} = \sum_{n \geq 0} \sum_{u \in \mathcal{F}} Pr[X_n \in \mathcal{F} / X_0 = u] \pi_0(u)$$

Let $Q = (P_{u,v})_{\substack{u \in \mathcal{F} \\ v \in \mathcal{F}}}$ then we have

$$Pr[X_i \in \mathcal{F} / X_0 = u] = \sum_{v \in \mathcal{F}} (Q^i)_{u,v}.$$

Thanks to the preceding proposition we finally obtain

$$\bar{N} = \sum_{u \in \mathcal{F}} \pi_0(u) \sum_{v \in \mathcal{F}} R_{u,v}$$

Note 13. If $\{X_i\}_{i \in \mathbb{N}}$ corresponds to an algorithm whose iterations are independent, with a fixed success probability π for each iteration, then the average number of iteration is

$$\begin{aligned} \bar{N} &= \sum_{n \geq 1} Pr[N \geq n] \\ &= \sum_{n \geq 0} (1 - \pi)^n \\ &= \frac{1}{\pi} \end{aligned}$$

12.3 Results

This general result can easily be adapted to any of our algorithms as shown in appendix C. Using the implicit formulas of the work factors, we can find the optimal parameters of the attacks. This final improvement is shown in table 3.

PartV
Experimental results

In order to check the correctness of our optimizations, we have made a great number of simulations for a small binary problem: decoding a [256, 128] random linear code whose minimal distance is obtained by Gilbert-Varshamov's bound, *i.e.* recovering an error vector of weight 14. Table 4 gives the experimental results obtained with Leon's algorithm. For each set of parameters, 1000 computations have been made. The computer was a Sun Sparc 10 with two SuperSPARC 50 MHz running under Solaris 2.3.

Cryptosystem	Attacks	
	Lee	Leon
Stern	$p=1$ $2^{74.3}$	$p=2$ $\sigma=6$ $2^{72.4}$
McEliece	$p=1$ $2^{70.5}$	$p=2$ $\sigma=8$ $2^{68.1}$
Adams	$p=1$ $2^{72.5}$	$p=2$ $\sigma=8$ $2^{70.5}$

Cryptosystem	Attacks			
	Algo. A_{GD}	Algo. A_{PD}	Algo. A_{GS}	Algo. A_{PS}
Stern	$p=1$ $\sigma=17$ $s=5$ $2^{72.6}$	$p=2$ $\sigma=6$ $s=2$ $2^{72.7}$	$p=2$ $\sigma=6$ $s=2$ $2^{72.4}$	$p=2$ $\sigma=6$ $s=2$ $2^{72.5}$
McEliece	$p=1$ $\sigma=11$ $s=2$ $2^{68.4}$	$p=2$ $\sigma=8$ $s=2$ $2^{68.7}$	$p=2$ $\sigma=8$ $s=2$ $2^{68.1}$	$p=2$ $\sigma=8$ $s=2$ $2^{68.3}$
Adams	$p=1$ $\sigma=11$ $s=2$ $2^{70.4}$	$p=2$ $\sigma=8$ $s=2$ $2^{71.0}$	$p=1$ $\sigma=14$ $s=3$ $2^{70.5}$	$p=2$ $\sigma=8$ $s=2$ $2^{70.6}$

Cryptosystem	Attacks				**best algorithm**
	$Stern_{GD}$	$Stern_{PD}$	$Stern_{GS}$	$Stern_{PS}$	
Stern	$p=2$ $\ell=15$ $2^{69.8}$	$p=2$ $\ell=15$ $2^{69.8}$	$p=2$ $\ell=15$ $2^{69.9}$	$p=2$ $\ell=15$ $2^{69.9}$	$Stern_{GD}$: $2^{69.8}$
McEliece	$p=2$ $\ell=18$ $2^{64.3}$	$p=2$ $\ell=18$ $2^{64.3}$	$p=2$ $\ell=18$ $2^{64.2}$	$p=2$ $\ell=18$ $2^{64.2}$	$Stern_{GS}$: $2^{64.2}$
Adams	$p=2$ $\ell=18$ $2^{66.0}$	$p=2$ $\ell=18$ $2^{66.0}$	$p=2$ $\ell=18$ $2^{66.0}$	$p=2$ $\ell=18$ $2^{66.0}$	$Stern_{GS}$: $2^{66.0}$

Table 3. Work factors of optimized modified attacks with Markov chain improvement

PartVI

Accessible weights

The computer speed dramatically increases and every interpretation of the above theoretical work factors must be taken with care. The experiments show that a theoretical work factor of 2^{30} is currently done in about 15 seconds on a Sun 10 with two SuperSPARC 50 MHz running under Solaris 2.3. Hence, we can estimate that a work factor of 2^{50} can be done in about three months of computation.

On the other hand, the number $\mathcal{N}(n,k,w)$ of codewords of weight w in an

Type	Parameters	Theoretical work factor $\log_2(W)$	Theoretical $\bar{N}$	Average $\bar{N}$	Deviation	Average CPU (s)	Corrected CPU (s)
	$p = 2,\ \sigma = 2$	29.3	244	256	+4.9%	7.10	6.8
	$p = 2,\ \sigma = 3$	29.3	266	275	+3.5%	6.31	6.1
	$p = 2,\ \sigma = 4$	29.3	291	292	+0.2%	6.12	6.1
	$p = 3,\ \sigma = 5$	29.4	61	60	-1.6%	7.42	7.5
	$p = 3,\ \sigma = 6$	29.4	66	68	+2.8%	6.61	6.4
	$p = 3,\ \sigma = 7$	29.5	71	72	+1.7%	6.12	6.0
	$p = 3,\ \sigma = 8$	29.7	77	76	-0.9%	5.91	6.0
	$p = 1,\ \sigma = 2$	28.2	23970	23776	-0.8%	4.85	4.9
Spare	$p = 1,\ \sigma = 3$	28.1	26247	26753	+1.9%	4.64	4.6
	$p = 1,\ \sigma = 4$	28.1	28770	29751	+3.4%	4.65	4.5
	$p = 1,\ \sigma = 5$	28.2	31565	33959	+7.6%	5.06	4.7
Markov	$p = 2,\ \sigma = 5$	29.5	4940	4921	-0.4%	10.78	10.8
chain	$p = 2,\ \sigma = 6$	29.5	5329	5494	+3.1%	10.35	10.0
	$p = 2,\ \sigma = 7$	29.6	5755	5980	+3.9%	10.84	10.4
	$p = 2,\ \sigma = 8$	29.8	6220	6474	+4.1%	10.46	10.0
	$p = 2,\ \sigma = 9$	30.0	6728	6869	+2.1%	10.71	10.5

Table 4. Experiments on $[256, 129, 14]$ code using Leon's algorithm

$[n, k]$-random linear code can be estimated by the following formula:

$$\mathcal{N}(n, k, w) \sim \frac{\binom{n}{w}}{2^{n-k}}. \tag{4}$$

For sufficient large values of w, this number grows enough to allow the algorithms to find at least one of the codewords. In fact the work factor can be divided by $\mathcal{N}(n, k, w)$. Table 5 shows an estimation of the weights that can be found using the described algorithms. We note that the use of any code of length 512 in McEliece's scheme will fail as the upper bound is greater than the error-correcting ability of such codes.

13 Conclusions

We here improve all the previously known algorithms for finding short words in a coset of any linear code, since all our results can easily be extended codes over $GF(q)$ (see appendix D).

New attacks on the cryptosystems based on error-correcting codes then follow. We notably obtain an attack on McEliece's cryptosystem which requires $2^{64.2}$ elementary operations. Such an attack is of course still infeasible, but it enables us to define the parameters which could be insecure.

We show for instance that the whole message in McEliece's cryptosystem can be recovered in 2^{50} operations if only 120 bits of the plain-text are known

Binary code	[512,256]	[1024,654]	[1024,524]
w	56 (Stern)	37 (Adams)	50 (McEliece)
Upper bound	37	25	35
Parameters	$p = 2$ $2^{49.1}$ $\ell = 15$	$p = 2$ $2^{49.9}$ $\ell = 18$	$p = 2$ $2^{49.9}$ $\ell = 18$
Estimated minimum weight	58	72	111
Lower bound	72	117	171
Parameters	$p = 2$ $2^{49.5}$ $\ell = 13$	$p = 2$ $2^{49.6}$ $\ell = 15$	$p = 2$ $2^{50.2}$ $\ell = 15$

Table 5. Accessible weights using Stern's $_{GSM}$ algorithm

- this corresponds to the work factor required for finding a word of weight 50 in a [1024,524-125] code using Stern's modified algorithm -. This could occur if several relatively close plain-texts are sent.

We otherwise notice that the parameters for McEliece's cryptosystem which maximize the work factor of these new attacks are $n = 1024$, $k = 614$ and $w = 41$; the corresponding work factor using Stern's $_{GSM}$ algorithm is 2^{66}.

References

1. C. Adams and H. Meijer. Security-related comments regarding McEliece's public-key cryptosystem. In C. Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, number 293 in Lecture Notes in Computer Science, pages 221–228. Springer-Verlag, 1988.
2. E.R. Berlekamp, R.J. McEliece and H.C.A. Van Tilborg. On the inherent intractability of certain coding problems *IEEE Trans. Inform. Theory,* IT-24(3):384–386, May 1978.
3. E.F. Brickell and A.M. Odlyzko. Cryptanalysis: A survey of recent results. In *Proceedings of IEEE*, volume 76, pages 578–593, May 1988.
4. A. Canteaut and H. Chabanne. A further improvement of the work factor in an attempt at breaking McEliece's cryptosystem. In P. Charpin, editor, *EUROCODE 94 – Livre des résumés*, pages 163–167. INRIA, oct 1994.
5. H. Chabanne and B. Courteau. Application de la méthode de décodage itérative d'Omura à la cryptanalyse du système de McEliece. Rapport 122, Université de Sherbrooke, Canada, 1993.
6. F. Chabaud. Asymptotic analysis of probabilistic algorithms for finding short codewords. In P. Camion, P. Charpin and S.Harari, editors, *EUROCODE 92*, number 339 in CISM Courses and Lectures, pages 175–183. Springer-Verlag, 1993.
7. F. Chabaud. On the security of some cryptosystems based on error-correcting codes. In *pre-proceedings of EUROCRYPT '94*. Springer-Verlag, 1994.
8. J.G. Kemeny and J.L. Snell. *Finite Markov chains.* D.Van Nostrand, Princeton, New Jersey, 1960.

9. P.J. Lee and E.F. Brickell. An observation on the security of McEliece's public-key cryptosystem. In C.G. Günther, editor, *Advances in Cryptology – EUROCRYPT '88*, number 330 in Lecture Notes in Computer Science, pages 275–280. Springer-Verlag, 1989.
10. J.S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Trans. Inform. Theory*, IT-34(5):1354–1359, September 1988.
11. Y.X. Li, R.H. Deng, and X.M. Wang. On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. *IEEE Trans. Inform. Theory*, IT-40(1):271–273, January 1994.
12. R. Lidl and H. Niederreiter. Finite fields. In Gian-Carlo Rota, editor, *Encyclopedia of Mathematics and its applications*. Addison-Wesley Publishing Company, 1983.
13. R.J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN progress report 42-44*, pages 114–116, 1978.
14. H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):159–166, 1986.
15. J.K. Omura. Iterative decoding of linear codes by a modulo-2 linear program. *Discrete Math*, 3:193–208, 1972.
16. N. Sendrier. On the structure of a randomly permuted concatenated code. In *EUROCODE 94 – Livre des résumés*, pages 169–173. INRIA, oct 1994.
17. V.M. Sidelnikov and S.O. Shestakov. On cryptosystems based on generalized Reed-Solomon codes. *Diskretnaya Math*, 4:57–63, 1992. in Russian.
18. J. Stern. A method for finding codewords of small weight. In G. Cohen and J. Wolfmann, editors, *Coding Theory and Applications*, number 388 in Lecture Notes in Computer Science, pages 106–113. Springer, 1989.
19. J. Stern. A new identification scheme based on syndrome decoding. In D.R. Stinson, editor, *Advances in Cryptology – CRYPTO '93*, number 773 in Lecture Notes in Computer Science, Springer-Verlag, 1994.
20. J. van Tilburg. On the McEliece public-key cryptosystem. In S. Goldwasser, editor, *Advances in Cryptology – CRYPTO '88*, number 403 in Lecture Notes in Computer Science, pages 119–131. Springer-Verlag, 1990.

A Work factor of the previously known attacks

A.1 McEliece's algorithm

This algorithm corresponds to Lee-Brickell's algorithm with parameter $p = 0$.

A.2 Lee-Brickell's algorithm

This algorithm is a GD-type algorithm.

For this algorithm the success probability depends on the value of parameter p.

$$\pi_{Lee(p)}(n,k,w) = \sum_{i=0}^{p} \frac{\binom{n-w}{k-i}\binom{w}{i}}{\binom{n}{k}}.$$

At each iteration, we perform about $\sum_{i=0}^{p} i \times \binom{k}{i}$ additions of $(n-k)$-bit words and a weight checking:

$$J_{Lee(p)}(n,k) = (n-k)\left[\sum_{i=0}^{p}(i+1) \times \binom{k}{i}\right].$$

A.3 Leon's algorithm

This algorithm is of GS-type. The success probability is

$$\pi_{L(p,\sigma)}(n,k,w) = \sum_{i=0}^{p} \frac{\binom{n-w}{k+\sigma-i}\binom{w}{i}}{\binom{n}{k+\sigma}}.$$

According to [12, page 455], the number of $(k, k+\sigma)$ matrices over $GF(2)$ of rank $1 \le e \le k$ is

$$2^{\frac{e(e-1)}{2}} \prod_{i=0}^{e-1} \frac{(2^{k+\sigma-i}-1)(2^{k-i}-1)}{(2^{i+1}-1)}.$$

Hence, the probability for the $(k, k+\sigma)$ extracted matrix to be of rank e is

$$\rho(\sigma,k,e) = \frac{2^{\frac{e(e-1)}{2}}}{2^{(k+\sigma)k}} \prod_{i=0}^{e-1} \frac{(2^{k+\sigma-i}-1)(2^{k-i}-1)}{(2^{i+1}-1)}.$$

In this case, the average number of additions and weight-checking on $(s-e)$-bit words is given by $\sum_{i=1}^{p} i\binom{e}{i}$, and the probability that the complete computation on the remaining $(n-k-\sigma) \times 2^{k-e}$ bits would be necessary is for each $1 \le i \le p$

$$\sum_{j=0}^{p-i} \frac{\binom{k+\sigma-e}{j}}{2^{k+\sigma-e}}.$$

Finally, we get the average number of operations by iteration of Leon's algorithm:

$$J_{L(p,\sigma)}(n,k) = \sum_{e=1}^{k} \rho(\sigma,k,e) \times \left[\sum_{i=1}^{p} i\binom{e}{i}\left[(k+\sigma-e)+(n-k-\sigma)\times 2^{k-e} \times \sum_{j=0}^{p-i}\frac{\binom{k+\sigma-e}{j}}{2^{k+\sigma-e}}\right]\right].$$

Note 14. As pointed out by Pascal Veron, the degenerative cases for the extracted matrix ($e \ne 0$) greatly increase the number of operations. Hence the only selections of columns that will be considered are the non-degenerative ones. This leads to minor change for the success probability:

$$\pi_{L(p,s)}(n,k,w) = \sum_{i=1}^{p} \frac{\binom{n-w}{s-i}\binom{w}{i}}{\binom{n}{s}}.$$

On the other hand, the average number of operations by iteration becomes:

$$J_{L(p,\sigma)}(n,k) = \sum_{i=1}^{p} i\binom{k}{i}\left[\sigma + (n-k-\sigma)\frac{\sum_{j=0}^{p-i}\binom{\sigma}{j}}{2^{\sigma}}\right].$$

A.4 Stern's algorithm

***PS* type (initial version)** The initial version of this algorithm is of PS-type. The success probability is according to [18]:

$$\pi_{S(p,\ell)}(n,k,w) = \frac{\binom{n-w}{k-2p}\binom{w}{2p}}{\binom{n}{k}} \frac{\binom{2p}{p}}{4^p} \frac{\binom{n-w-k+2p}{\ell}}{\binom{n-k}{\ell}}$$

and the average number of operations by iteration is:

1. $2p\binom{k/2}{p}$ operations on ℓ-bit words.
2. The average number of collisions is $\frac{\binom{k/2}{p}^2}{2^\ell}$ and for each collision we must perform $2p-1$ additions and a weight checking of $(n-k-\ell)$ bit codewords.
3. We need $K(p\binom{k/2}{p}+2^\ell)$ more operations to perform the dynamic memory allocation where K is the size of a computer word (usually 32).
4. As the efficient data structure to perform a Gaussian elimination is the row-structure, and on the other hand column structure is necessary for the rest of the iteration, you must add a conversion which needs $(n-k)k$ operations.

Hence, we obtain

$$J_{S_{PS}(p,\ell)}(n,k) = (n-k)\times k + 2p\ell\binom{k/2}{p}$$
$$+2p(n-k-\ell)\frac{\binom{k/2}{p}^2}{2^\ell} + K \times \left(p\binom{k/2}{p} + 2^\ell\right).$$

***GS* type** The single modification is that in this case the conversion between the row-structure and the column-structure is not necessary.

Then we have:

$$J_{S_{GS}(p,\ell)}(n,k) = 2p\ell\binom{k/2}{p}$$
$$+2p(n-k-\ell)\frac{\binom{k/2}{p}^2}{2^\ell} + K \times \left(p\binom{k/2}{p} + 2^\ell\right).$$

***GD* type** The success probability is unchanged but the number of operations by iteration slightly increases since we need at each step one more addition to compare the word to decode with the codewords obtained by the linear combinations.

$$J_{S_{GD}(p,\ell)}(n,k) = (2p+1)\ell\binom{k/2}{p}$$
$$+(2p+1)(n-k-\ell)\frac{\binom{k/2}{p}^2}{2^\ell} + K \times \left(p\binom{k/2}{p} + 2^\ell\right).$$

***PD* type** Same remark.

$$J_{S_{PD}(p,\ell)}(n,k) = (n-k)\times k + (2p+1)\ell\binom{k/2}{p}$$
$$+(2p+1)(n-k-\ell)\frac{\binom{k/2}{p}^2}{2^\ell} + K \times \left(p\binom{k/2}{p} + 2^\ell\right).$$

B Algorithm *A*

B.1 *GS* type

This version is described in section 8.1. The success probability of the algorithm is

$$\pi_{A_{GS}(p,\sigma,s)}(n,k,w) = \sum_{i=1}^{p}\left[\frac{\binom{w}{i}\binom{n-w}{k-i}}{\binom{n}{k}}\sum_{j=\max(0,\sigma-(n-k-w+i))}^{\min(\sigma,s-i)}\frac{\binom{w-i}{j}\binom{n-k-w+i}{\sigma-j}}{\binom{n-k}{\sigma}}\right]$$

For each iteration, an estimation of the number of operations is:

1. $\sum_{i=1}^{p}\binom{k}{i}(i-1)$ additions of σ-bit words and a weight computation.
2. The average number of cases such that the whole computation is needed is

$$\sum_{i=1}^{p}\binom{k}{i}\frac{\sum_{j=0}^{s-i}\binom{\sigma}{j}}{2^\sigma}.$$

We consequently have the following estimation:

$$J_{A_{GS}(p,\sigma,s)}(n,k) = \sum_{i=1}^{p} i\binom{k}{i}\left[\sigma + (n-k-\sigma)\frac{\sum_{j=0}^{s-i}\binom{\sigma}{j}}{2^\sigma}\right].$$

B.2 *GD* type

In this case, as the word to decode is kept outside of the matrix, it is possible to find and detect an information window which contains no error position.

$$\pi_{A_{GD}(p,\sigma,s)}(n,k,w) = \sum_{i=0}^{p}\left[\frac{\binom{w}{i}\binom{n-w}{k-i}}{\binom{n}{k}}\sum_{j=\max(0,\sigma-(n-k-w+i))}^{\min(\sigma,s-i)}\frac{\binom{w-i}{j}\binom{n-k-w+i}{\sigma-j}}{\binom{n-k}{\sigma}}\right]$$

Besides, we need at each step one more addition to compare the word to decode with the codewords obtained by the linear combinations:

$$J_{A_{GD}(p,\sigma,s)}(n,k) = \sum_{i=0}^{p}(i+1)\binom{k}{i}\left[\sigma + (n-k-\sigma)\frac{\sum_{j=0}^{s-i}\binom{\sigma}{j}}{2^\sigma}\right].$$

B.3 *PS* type

This version is described in section 8.3. In binary case we must take care of the implantation constraints. The Gaussian elimination must be performed on the rows and the other operations on the columns of the matrix. A small conversion is therefore needed, and it requires $k \times (n-k)$ operations. Otherwise, the operations are similar.

$$J_{A_{PS}(p,\sigma,s)}(n,k) = k \times (n-k) + \sum_{i=1}^{p} i \binom{k}{i} \left[\sigma + (n-k-\sigma) \frac{\sum_{j=0}^{s-i} \binom{\sigma}{j}}{2^{\sigma}} \right].$$

B.4 *PD* type

Same remark.

$$J_{A_{PD}(p,\sigma,s)}(n,k) = k \times (n-k) + \sum_{i=0}^{p} (i+1) \binom{k}{i} \left[\sigma + (n-k-\sigma) \frac{\sum_{j=0}^{s-i} \binom{\sigma}{j}}{2^{\sigma}} \right].$$

C Average number of iterations for the algorithms with Markov chain improvement

C.1 Expected number of iterations for Lee-Brickell's *DM* algorithm

The first description of this modified algorithm was given in [4].

Let us suppose we look for a codeword e of weight w. Then the algorithm can be described by a random variable X_i that represents the number of positions of $supp(e)$ contained in the information window I at step i. Considering Lee-Brickell's algorithm with parameter p means that the algorithm stops as soon as the information window contains p positions of $supp(e)$.

Proposition 15. *The stochastic process associated with the modified Lee-Brickell's algorithm with parameter p is a stationary Markov chain whose state space is $\mathcal{E} = \{0, \cdots, w\}$ and success space is $\mathcal{S} = \{0, \cdots, p\}$.*

Proof. We just have to find the transition matrix of the algorithm, that is to say for all $(u, v) \in \mathcal{F}^2$, the probability

$$P_{u,v} = Pr[X_{i+1} = v / X_i = u].$$

Since we only change one column of the information window I, we have

$$\begin{aligned}
P_{u,u} &= \frac{k-u}{k} \times \frac{n-k-(w-u)}{n-k} + \frac{u}{k} \times \frac{w-u}{n-k} \\
P_{u,u-1} &= \frac{u}{k} \times \frac{n-k-(w-u)}{n-k} \\
P_{u,u+1} &= \frac{k-u}{k} \times \frac{w-u}{n-k} \\
P_{u,v} &= 0 \text{ for all } v \notin \{u-1, u, u+1\}
\end{aligned}$$

Hence by applying proposition 12, we get the average number of iterations for Lee-Brickell's algorithm.

Note 16. The initial probability vector is

$$\pi_0 = \left(\frac{\binom{w}{u}\binom{n-w}{k-u}}{\binom{n}{k}} \right)_{0 \le u \le w}.$$

It is easy to see that π_0 is an invariant probability distribution for this chain:

$$\pi_i = \pi_0 \text{ for all } i \ge 0.$$

C.2 Expected number of iterations for Stern's M-type algorithm

The modification of Stern's algorithm consists in replacing the Gaussian elimination in the first step by the procedure detailed in section 11.

Proposition 17. *The stochastic process associated with the modified Stern's algorithm is a stationary Markov chain whose state space is* $\mathcal{E} = \{0, \dots, 2p-1\} \cup \{2p+1, \dots, w\} \cup \{(2p)_S, (2p)_F\}$ *and whose single success state is* $(2p)_S$*. The state* $(2p)_F$ *corresponds to a correct number of supp(e)-positions in the information window I but to an incorrect distribution of these positions in L and among* I_1 *and* I_2 *(see section 11.2).*

Proof. As $(2p)_S$ corresponds to a correct distribution of the error positions among all windows, we have

$$\begin{aligned} \beta &= Pr[X_i = (2p)_S / X_i = 2p] \\ &= \frac{\binom{2p}{p}\binom{k-2p}{\lfloor k/2 \rfloor - p}}{\binom{k}{\lfloor k/2 \rfloor}} \frac{\binom{n-k-w+2p}{\ell}}{\binom{n-k}{\ell}} \end{aligned}$$

Hence the transition probabilities corresponding to the transient states are

$$\begin{aligned} P_{u,u} &= \frac{k-u}{k} \times \frac{n-k-(w-u)}{n-k} + \frac{u}{k} \times \frac{w-u}{n-k} \text{ for all } u \ne (2p)_F \\ P_{u,u-1} &= \frac{u}{k} \times \frac{n-k-(w-u)}{n-k} \text{ for all } u \ne 2p+1 \\ P_{u,u+1} &= \frac{k-u}{k} \times \frac{w-u}{n-k} \text{ for all } u \ne 2p-1 \\ P_{u,v} &= 0 \text{ for all } v \notin \{u-1, u, u+1\} \\ P_{(2p)_F,(2p)_F} &= (1-\beta) \left[\frac{k-2p}{k} \times \frac{n-k-(w-2p)}{n-k} + \frac{2p}{k} \times \frac{w-2p}{n-k} \right] \\ P_{2p+1,(2p)_F} &= (1-\beta) \left[\frac{2p+1}{k} \times \frac{n-k-(w-(2p+1))}{n-k} \right] \\ P_{2p-1,(2p)_F} &= (1-\beta) \left[\frac{k-(2p-1)}{k} \times \frac{w-(2p-1)}{n-k} \right] \end{aligned}$$

And we can apply proposition 12 to get the average number of iterations for modified Stern's algorithm.

C.3 Expected number of iterations for algorithm A_{DM}

Proposition 18. *The stochastic process associated with algorithm A_{DM} is a stationary Markov chain whose success space is $\mathcal{E} = \{(0)_S, \ldots, (p)_S\}$ and whose failure space is $\mathcal{F} = \{(0)_F, \ldots, (p)_F\} \cup \{p+1, \ldots, w\}$. The states $(u)_F$, $0 \le u \le p$, correspond to a correct number of errors in the information window I but a too large number of these errors in Σ (see section 11.1).*

Proof. As $(u)_S$ corresponds to a correct distribution of the error positions among all windows, we have

$$\begin{aligned}\beta_u &= Pr[X_i = (u)_S / X_i = u] \\ &= \sum_{j=0}^{\min(\sigma, s-u)} \frac{\binom{w-u}{j}\binom{n-k-w+u}{\sigma-j}}{\binom{n-k}{\sigma}}\end{aligned}$$

Hence the transition probabilities corresponding to the transient states are

$$\begin{aligned}
P_{u,u} &= \frac{k-u}{k} \times \frac{n-k-(w-u)}{n-k} + \frac{u}{k} \times \frac{w-u}{n-k} \text{ for all } u > p \\
P_{u,u-1} &= \frac{u}{k} \times \frac{n-k-(w-u)}{n-k} \text{ for all } u > p+1 \\
P_{u,u+1} &= \frac{k-u}{k} \times \frac{w-u}{n-k} \text{ for all } u > p-1 \\
P_{u,j} &= 0 \text{ for all } j \notin \{u-1, u, u+1\} \\
P_{(u)_F,(u)_F} &= (1-\beta_u)\left[\frac{k-u}{k} \times \frac{n-k-(w-u)}{n-k} + \frac{u}{k} \times \frac{w-u}{n-k}\right] \text{ for all } 0 \le u \le p \\
P_{(u)_F,(u-1)_F} &= (1-\beta_{u-1})\left[\frac{u}{k} \times \frac{n-k-(w-u)}{n-k}\right] \text{ for all } 1 \le u \le p+1 \\
P_{(u)_F,(u+1)_F} &= (1-\beta_{u+1})\left[\frac{k-u}{k} \times \frac{w-u}{n-k}\right] \text{ for all } 0 \le u \le p
\end{aligned}$$

And we can apply proposition 12 to get the average number of iterations for this algorithm.

D General case over $GF(q)$

D.1 Generalities

The general equations of sections 8.3 and 12.1 remain true. But, whereas the average number of iterations of the algorithms is unchanged, the cost of the operations increases.

In the following $\mathcal{M}$ will denote the cost of a multiplication, $\mathcal{A}$ the cost of an addition and $\mathcal{I}$ the cost of an inversion over $GF(q)$.

Then the previously defined costs become:

$$\begin{aligned}
\mathcal{G}(n,\kappa) &= \kappa \times \left(\mathcal{I} + \kappa \times \frac{q-1}{q} \times n(\mathcal{M}+\mathcal{A})\right), \\
\mathcal{D}_G(n,k) &= \frac{q-1}{q} k \times n(\mathcal{M}+\mathcal{A}), \\
\mathcal{D}_P(n,k) &= \frac{q-1}{q}(n-k) \times (n-k)(\mathcal{M}+\mathcal{A}), \\
g(n,\kappa) &= \mathcal{I} + \kappa \times \frac{q-1}{q} \times (n-\kappa)(\mathcal{M}+\mathcal{A}), \\
d(n,k) &= \frac{q-1}{q} \times (n-k)(\mathcal{M}+\mathcal{A}).
\end{aligned}$$

Formula 4 becomes

$$\mathcal{N}_q(n,k,w) \sim q^k \frac{\binom{n}{w}(q-1)^w}{q^n}.$$

Note 19. In $GF(2)$ we had $\mathcal{I} = \mathcal{M} = 0$ and $\mathcal{A} = 1$. For $m < 32$ we may estimate the computation costs in $GF(2^m)$ as follows:

$$\begin{aligned}
\mathcal{A} &= 1 \\
\mathcal{M} &= m^2 \\
\mathcal{I} &= m^2 \log_2 m.
\end{aligned}$$

D.2 Algorithm *A*

***S* type** For each iteration, an estimation of the number of operations is:

1. $\sum_{i=1}^{p} \binom{k}{i}(q-1)^i(i-1)$ additions of σ-elements words and a weight computation.
2. The average number of cases such that the whole computation is needed is

$$\sum_{i=1}^{p} \binom{k}{i}(q-1)^i \frac{\sum_{j=0}^{s-i}\binom{\sigma}{j}(q-1)^j}{q^\sigma}.$$

Consequently we have the following estimation:

$$J_{A_{S(p,\sigma,s)}}(n,k) = \sum_{i=1}^{p}((i-1)\times\mathcal{A}+i\times\mathcal{M}+\mathcal{A})\binom{k}{i}(q-1)^i\left[\sigma + (n-k-\sigma)\frac{\sum_{j=0}^{s-i}(q-1)^j\binom{\sigma}{j}}{q^\sigma}\right].$$

***D* type** We have one more addition to perform to compare the given word with the codewords

$$J_{A_{D(p,\sigma,s)}}(n,k) = \sum_{i=0}^{p}(i\times\mathcal{A}+i\times\mathcal{M}+\mathcal{A})\binom{k}{i}(q-1)^i\left[\sigma + (n-k-\sigma)\frac{\sum_{j=0}^{s-i}(q-1)^j\binom{\sigma}{j}}{q^\sigma}\right].$$

This article was processed using the LaTeX macro package with LLNCS style