A further improvement of the work factor in an attempt at breaking McEliece's cryptosystem

Anne Canteaut, Hervé Chabanne

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A Further Improvement of the Work Factor in an Attempt at Breaking McEliece's Cryptosystem

Anne CANTEAUT - Hervé CHABANNE

N° 2227
Mars 1994

PROGRAMME 2

Calcul symbolique,
programmation
et génie logiciel

A further improvement of the work factor in an attempt at breaking McEliece's cryptosystem

Anne Canteaut[†*] and Hervé Chabanne[†]

March 1994

Abstract

The public-key cryptography has become dangerously dependent on the difficulty of a single problem: factoring. We need therefore to study other difficult problems which could be applied to public-key cryptography in order to anticipate an important progress in factoring methods. McEliece's cryptosystem is an alternative to RSA although the size of the required public keys is too large for pratical uses. It relies on the NP-completeness of the general decoding problem for linear codes. The known best general decoding algorithm for $[n, k, d]$-linear codes proposed by Lee and Brickell consists in repeatedly selecting at random a set of k bits, called an information window, from the n-bit corrupted vector until the number of errors in the window is small enough to recover the correct message. In this paper we propose an iterative algorithm which modifies only one bit of the information window at each iteration instead of choosing a completely new set so that the number of operations in each iteration is reduced by a factor k. A theorical calculation of the average work factor and a great number of simulations show that the use of linear code of length 512 for cryptographic purposes is not secure. Then we can notably conclude that it is not possible to reduce the key-size in McEliece's system by using codes of length 512.

*Ecole Nationale Supérieure de Techniques Avancées, Laboratoire LEI, 32 boulevard Victor, 75015 Paris, FRANCE; email: Anne.Canteaut@inria.fr

†Institut National de Recherche en Informatique et Automatique, projet Codes, Domaine de Voluceau, 78153 Le Chesnay Cedex, FRANCE

Une amélioration supplémentaire du facteur de travail dans l'attaque du cryptosystème de McEliece

Résumé

La cryptographie à clef publique est devenue de nos jours dangeureusement dépendante de la difficulté d'un unique problème, celui de la factorisation. L'étude d'autres problèmes difficiles applicables à la cryptographie à clef publique est donc indispensable afin de se prémunir contre d'importants progrès dans le domaine de la factorisation. Le cryptosystème de McEliece constitue une alternative à RSA bien que la taille très élevée des clefs publiques qu'il requiert soit dissuasive pour son utilisation. Il repose sur la NP-complétude du problème général de décodage des codes linéaires. Le meilleur algorithme connu pour décoder un code linéaire $[n, k, d]$ quelconque est celui de Lee et Brickell ; il consiste à sélectionner successivement au hasard parmi les n bits du vecteur corrompu des ensembles de k bits, appelés fenêtres d'information, jusqu'à ce que la fenêtre choisie contienne un nombre d'erreurs suffisamment faible pour retrouver le message correct. Dans cet article, nous proposons un algorithme itératif qui ne modifie qu'un seul bit de la fenêtre d'information à chaque itération au lieu de choisir un ensemble complètement nouveau, ce qui permet de diviser le nombre d'opérations lors de chaque itération par un facteur k. Un calcul théorique du facteur de travail moyen et un grand nombre de simulations montrent que l'utilisation de codes linéaires de longueur 512 dans des applications cryptographiques n'est pas sûre. Nous avons ainsi prouvé qu'il est impossible de réduire la taille des clefs dans le système de McEliece en utilisant des codes de longueur 512.

1 Introduction

We propose a method for decoding linear binary codes. Finding improvements of the running-time for decoding may lead to a cryptanalysis of McEliece cryptosystem [9].

Let $\mathcal{C}$ be a $[n, k, d]$-linear binary code and G its generator matrix, i-e a set of k vectors which generates $\mathcal{C}$.

An information window is a subset $\{i_1, \cdots, i_k\} \subset N = \{1, \cdots, n\}$ such that the columns $G_{i_1}, \cdots, G_{i_k}$ are lineary independent. The complementary set of an information window in N is a redundancy window. It is well-known that when the support of an error is included in a redundancy window we may recover the transmitted bits [6].

Following Lee and Brickell [5] we want to explore the set of all information windows of a code replacing at each iteration an information column by a redundancy one. This method was first investigated by Omura [10].

An advantage of such a method is to perform each iteration in roughly k^2 operations; another one is to have to deal only with information windows and to be able to reach all information windows.

Using Markov chains theory we give the average number of iterations. We obtain even better practical results. For example, it appears to be insecure to use linear binary codes of length 512 for cryptographic purposes. On the other hand our procedure is to our knowledge the best way for decoding without using structure of codes.

2 The decoding problem

Consider the vector space $GF(2)^n$ with the Hamming metric, and $\mathcal{C}$ a k-dimensional subspace of $GF(2)^n$.

The weight of a vector x in $GF(2)^n$ is the number of non-zero entries of x; it is denoted by $w(x)$. Let d denotes the smallest of the weights of the non-zero elements of $\mathcal{C}$.

$\mathcal{C}$ is a $[n, k, d]$-linear binary code where n is the length of the code, k the dimension and d the minimal distance.

Let c be a codeword, e a n-dimensional binary error vector, and $y = c + e$ the received vector. The decoding problem is to find from y the codeword c or equivalently the error e.

Put $t = [\frac{d-1}{2}]$. As the coset $\{e + c,\ c \in \mathcal{C}\}$ contains at most one element of weight less than or equal to t, it is possible to solve the decoding problem without ambiguity when $w(e) \leq t$.

An obvious decoding algorithm is to pick at random k bits of vector y in hope that none of this bits are in error:

Algorithm 1: (as described in [9] [1])

Let G be a generator matrix of the code

1. Select k positions from n
   - project $y = c + e$ onto these positions
   - take the $k \times k$-submatrix of G, V, composed of the k corresponding columns
2. If the choosen positions do not contain any error positions, and if V is inversible, inverse V to obtain the codeword c.

Adams and Meijer [1] show the inefficiency of Algorithm 1 against the McEliece cryptosystem. In the next section we introduce several improvements.

3 An iterative decoding algorithm

Our algorithm is based as Algorithm 1, but it chooses the submatrices V such that they always are inversible and that V^{-1} can be fast computed.

Let G be a fixed generator matrix of the code $\mathcal{C}$.

Put $N = \{1, \cdots, n\}$.

Let I be a subset of N with k elements, and $J = N \setminus I$ the complementary set of I in N.

We denote $G = (W, V)_I$ the decomposition of G onto I, that means $W = (G_j)_{j \in J}$ and $V = (G_i)_{i \in I}$ where $(G_1, \cdots, G_n)$ is the columns' decomposition of G.

Definition 1 *An* **information window** *is a subset I of N with k elements such that $G = (W, V)_I$ where V is inversible.*

The complementary set $J = N \setminus I$ is called **redundancy window**.

We can immediatly affirm that such information windows exist because of the following result from coding theory:

Proposition 1 *Every set of $n - d + 1$ coordinate positions contains an information window.*

proof: Let X be a set of s coordinate positions, and $G = (A, B)_X$. Suppose X does not contain any information window. Then the column rank of B, and hence the row rank of B is less than k. Hence there exists a linear combination of rows of B which egals 0, and a codeword c which is 0 on these s positions. As we have $w(c) \geq d$, $n - s \geq d$. Then $s \leq n - d$.

Definition 2 *Two information windows I and I' are* **close** *if*

$$|I \setminus I'| = |J \setminus J'| = 2$$

In other terms we obtain I' by exchanging an element of I with an element of J.

Remark 1: The information windows of a linear binary code form the bases of a binary matroid [4].

The following result motivates the use of close information windows:

Proposition 2 *Let I and I' be two information windows. There exists a sequence of close information windows from I to I'.*

proof: It is a classical result of matroid theory: consider the matroid introduced in Remark 1 and the graph whose vertices are the basis of this matroid and where two vertices are bound if you can pass from one to the other by the exchange property. Then this graph is connected [7] [8].

The decoding algorithm consists therefore of covering a set of close information windows and searching I which contains no error position.

It is easy to find close windows thanks to the following proposition:

Proposition 3 (Condition of closeness) *Let I and I' be two information windows and $G = (W,V)_I = (W',V')_{I'}$.*

I and I' are close iff:

- $\exists l \in I$, $\exists p \in J$ *such that* $I' = I \setminus \{l\} \cup \{p\}$.
- *if* $Z = V^{-1}W = (z_{i,j})_{i \in I, j \in J}$, *we have* $z_{l,p} = 1$.

The proof is in [10].

We can check if an information window contains no error position with the method proposed in [5]:

Proposition 4 *Let I be an information window, $G = (W,V)_I$ and $y = c + e = (y_J, y_I)_I$.*

Put $x = y + y_I V^{-1} G$, then I contains no error position iff $w(x) \leq t$.

In this case we have $e = x$.

proof: Let y be $mG + e$. Let $\mathcal{C}(e) = \{c + e, c \in \mathcal{C}\}$.

As $y_I V^{-1} G$ is a codeword, x is an element of the coset $\mathcal{C}(e)$ and $\mathcal{C}(e)$ contains a unique word whose weight is less than or egal to t.

We index rows of Z with I using $V^{-1}G = (Z, Id)_I$ and we denote by Z^i the ith row of Z.

The following proposition shows how to update the matrix Z and the vector x between two close information windows.

Proposition 5 *Let I and I' be two close information windows such that $I' = I \setminus \{l\} \cup \{p\}$.*

Consider $Z = V^{-1}W$ and $Z' = V'^{-1}W'$ where $G = (W,V)_I = (W',V')_{I'}$, and consider $x = y + y_I V^{-1} G$ and $x' = y + y_{I'} V'^{-1} G$.

Then we have

.1. *Z' is obtained from Z by:*

- $Z'^p = Z^l$
- *if* $z_{i,p} = 0$ *then* $Z'^i = Z^i$.
- *if* $z_{i,p} = 1$ *then* $Z'^i = Z^i + Z^l$, $i \in I \setminus \{l\}$

2. $x' = x + Z^l x_p$.

proof:

1. As $G = (W, V)_I$ is a generator matrix of the code and as V is inversible,
   $\tilde{G} = V^{-1}G = (Z, Id)_I$ is a generator matrix in standard form of the code.
   We get $\tilde{G}' = (Z', Id)_{I'}$ by exchanging the lth and pth columns of $\tilde{G}$.
   It is a simple pivoting operations in (l, p) position.
   Thus we replace every row of $\tilde{G}$ by pivoting and we obtain:

$$\forall i \in I \setminus \{p\},\ \tilde{G}'^i = \tilde{G}^i + z_{i,p}\tilde{G}^l \text{ and } \tilde{G}'^p = \tilde{G}^l$$

2. We get the second point by making the same pivoting operation on vector x.

Therefore we only need to store matrix Z and vector x at each iteration.

To improve this decoding algorithm, we want to allow one error position in the information window as suggested by Lee and Brickell [5]. Hence we will test at each iteration the weight of all the vectors x' obtained from x by exchanging 2 columns in the generator matrix, i-e by adding one row of $\tilde{G}$.

We may easily check if $w(x') \leq t$:

Proposition 6 *Let $x' = x + \tilde{G}^l x_p$.*
A condition for $w(x') \leq t$ is $w(x) + w(Z^l) - 2w(x_J.Z^l) + 1 \leq t$
where $x = (x_J, x_I)_I$ and $a.b$ denotes the inner product in $GF(2)^n$.

proof:

- If $x_p = 0$, $w(x') = w(x)$.
- If $x_p = 1$, $x' = x + Z^l$.
  Thus we have $w(x') = w(x) + w(\tilde{G}^l) - 2w(x.\tilde{G}^l)$.
  But $\tilde{G}^l = (Z^l, (\delta_{l,j})_{j \in I})_I$ and $x_I = 0$ where $\delta_{l,j}$ denotes the Kronecker's symbol.
  Hence we have $w(\tilde{G}^l) = w(Z^l) + 1$ and $x.\tilde{G}^l = x_J.Z^l$.
  We obtain $w(x') = w(x) + w(Z^l) - 2w(x_J.Z^l) + 1$.

Therefore we propose the following algorithm:

Algorithm 2:
Choose an information window I, compute $Z = V^{-1}W$ and $x = y + y_I Z + y_I$. While $w(x) > t$

1. For each l in I, compute $\alpha_l = w(x) + w(Z^l) - 2w(x_J.Z^l) + 1$.
   - if $\alpha_l \leq t$ search p such that $z_{l,p} = 1$ and $x_p = 1$.
   - else select at random $l \in I$ and $p \in J$ such that $z_{l,p} = 1$
2. $I \leftarrow (I \setminus \{l\}) \cup \{p\}$
3. update matrix Z (proposition 5 (1))
4. update vector x (proposition 5 (2))

Remark 2: Selecting at random l and p may encounter cycles. In practice, for large n, such problems do not appear.

Remark 3: We may have the dual approach to Algorithm 2 by working with parity check matrices of the code instead of generator matrices (this is Omura's approach). A parity check matrix of $\mathcal{C}$ is a $(n-k) \times k$ binary matrix H such that $H^t G = 0$.

We call information window a set I such that $H = (B, R)_I$ where B is inversible and syndrome the vector $s = y^t H = e^t H$.

If $G = (Z, Id)_I$ is a generator matrix of the code, then $H = (Id, Y)_I$ is a parity check matrix with $Y = {}^t Z$.

Thus we can deduce similar results for the dual algorithm:

- $I' = I \setminus \{l\} \cup \{p\}$ is an information window iff $y_{p,l} = 1$.
- We can check whether I contains some error positions by the classical way used in permutation decoding [6]:
  Let $H = (B, R)_I$ and $x = (s^t(B^{-1}), 0)_I$.
  $supp(e) \subset J$ iff $w(x) \leq t$. In this case $e = x$.
- We pass from I to I' by the same pivoting operation as in Algorithm 2.

Remark 4: We may try different heuristics in order to speed up computations. For example Omura in [10] uses a "simplex" method that means he wants to decrease the weight of x at each iteration. We do not notice any improvement with this heuristic and we give it up.

4 Theorical running time for algorithm 2

We here obtain an explicit and computable expression for the expected running time of Algorithm 2.

4.1 Work factor involved in each iteration

This work factor is the average number of operations required for each iteration.

As we only make binary additions we can store each k-dimensional binary vector as a $\frac{k}{\alpha}$-dimensional vector of integers where $\alpha = 32$ or $\alpha = 64$ according to the internal representation of integers (we will use $\alpha = 32$).

- We assume that the cost for computing the weight of a k-dimensional binary vector is $\frac{k}{\alpha}$ since we have stored the weight of all α-dimensional vectors in a table. Therefore the work factor involved in step 1 is $2\frac{k^2}{\alpha}$.
- The work factor involved in step 3 is $w(Z_p)\frac{k}{\alpha}$ because the row Z^i does not change if $z_{i,p} = 0$.
- The work factor involved in step 4 is $\epsilon\frac{k}{\alpha}$ where $\epsilon = x_p = 0$ or 1.

Assuming that the average weight of Z^p is $\frac{k}{2}$ we have for each iteration:

$$w_f \leq \frac{k}{\alpha}\left(\frac{5}{2}k + 1\right)$$

Remark 5: If $x_p = 0$ the vector x does not change. Therefore it is not necessary to compute α_l at the next iteration if Z^l has not changed, i-e if $z_{l,p} = 0$.

4.2 Expected number of executions

Let the random variable X_i represent the number of error positions in the redundancy window J at step i, $X_i \in \{1, \cdots, t\}$.

The next redundancy window $J' = J \setminus \{p\} \cup \{l\}$ is obtained by replacing a redundancy position p by an information position l. Therefore we may remove $r = 0$ or 1 error position from J according to whether p is in $supp(e)$ or not, and we may add $r' = 0$ or 1 error position in J' according to whether l is in $supp(e)$ or not. Then we have for $s < t - 1$:

$$Pr[X_{i+1} = s' \,/ X_i = s] = \sum_{\substack{r, r' \in \{0,1\} \\ s' = s - r + r'}} \frac{\binom{s}{r}\binom{n-k-s}{1-r}}{n-k} \frac{\binom{t-s}{r'}\binom{k-t+s}{1-r'}}{k}$$

Thanks to our test (step 1) we have $Pr[X_{i+1} = t \,/ X_i = t-1] = 1$, and $Pr[X_{i+1} = t \,/ X_i = t] = 1$.

Since these probabilities do not depend from the previous iterations and from index i, the sequence of random variables (X_n) is an homogeneous Markov chain whose transition matrix P is defined by:

$$\forall s \leq t,\ \forall s' \leq t,\ \ P_{s,s'} = Pr[X_{i+1} = s' \,/ X_i = s]$$

We wish to evaluate the expectation of the number n of executions we have to make to reach $X_n = t$. As $X_n = t$ iff $X_{n-1} = t-1$, we have to estimate:

$$\begin{aligned}\bar{N}_i &= \sum_{n=1}^{\infty} n\, Pr[X_n = t-1, X_{n-1} < t-1 \,/ X_0 = i] \\ &= \sum_{n=1}^{\infty} \sum_{m=0}^{n-1} Pr[X_n = t-1, X_{n-1} < t-1 \,/ X_0 = i]\end{aligned}$$

Fubini's theorem applies since all terms are non-negative, and we obtain:

$$\begin{aligned}\bar{N}_i &= \sum_{m=0}^{\infty} \sum_{n=m+1}^{\infty} Pr[X_n = t-1, X_{n-1} < t-1 \,/ X_0 = i] \\ &= \sum_{m=0}^{\infty} Pr[\,\exists j > m,\ X_j = t-1, X_m < t-1 \,/ X_0 = i] \\ &= \sum_{m=0}^{\infty} Pr[X_m < t-1 \,/ X_0 = i] \\ &= \sum_{m=0}^{\infty} \sum_{j=0}^{t-2} Pr[X_m = j \,/ X_0 = i]\end{aligned}$$

Let Q denote the submatrix $(P_{i,j})_{\substack{0 \le i \le t-2 \\ 0 \le j \le t-2}}$.

It is easy to see that $Pr[X_m = j \,/ X_0 = i] = (Q^m)_{i,j}$.

Therefore the quantity $\sum_{m=0}^{\infty} Pr[X_m = j \,/ X_0 = i]$ is the (i,j)-entry of matrix R where

$$R = Id + Q + Q^2 + \ldots = (Id - Q)^{-1}$$

Then we have $\bar{N}_i = \sum_{j=0}^{t-2} R_{i,j}$.

The probability that there are exactly i errors among the initial redundancy window is:

$$p_i = \frac{\binom{t}{i}\binom{n-t}{n-k-i}}{\binom{n}{n-k}}$$

Since $\bar{N}_{t-1} = 1$ and $\bar{N}_t = 0$, the expected number of executions $\bar{N}$ is given by:

$$\bar{N} = \sum_{i=0}^{t-2} \bar{N}_i p_i + p_{t-1}$$

Remark 6: We observe that the numerical value of $\bar{N}_i$ scarcely changes as i varies. It shows that the choice of initial window doesn't play any role in the algorithm.

Figure 1 shows the evolution of the overall work factor $W_f = \bar{N}\, w_f$ of Algorithm 2 (we use for w_f the upper bound obtained at 4.1) for random $[n, k, d]$-linear codes where $\frac{k}{n} = \frac{1}{2}$ and d is obtained with the Gilbert-Varshamov's bound:

$$\frac{k}{n} = 1 - H_2(\frac{d}{n}) \text{ with } H_2(x) = -x\log_2(x) - (1-x)\log_2(1-x)$$

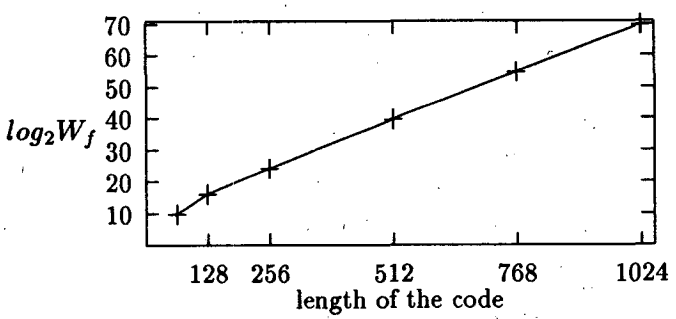

Figure 1: evolution of the work factor for Algorithm 2

4.3 Comparison with other decoding algorithms

Let us compare for different parameters this work factor with those found for Algorithm 1 in [1] and for Algorithm 1bis which is the attack proposed in [5]; this is a variant of Algorithm 1 which uses Proposition 4 for checking whether I contains some errors and allows 2 errors in I.

- **n=512, k=256, t=28 :** it corresponds to a [512,256]-random linear code whose minimal distance is obtained by the Gilbert-Varshamov's bound.
- **n=1024, k=524, t=50 :** these are the original parameters in McEliece's cryptosystem which uses Goppa codes.
- **n=1024, k=654, t=37 :** optimal value of t for McEliece's system which maximizes the work factor for Algorithm 1 [1].

	Algorithm 1	*Algorithm 1bis*	*Algorithm 2*
$n = 512,\ k = 256, t = 28$	$2^{53.1}$	$2^{44.8}$	$2^{39.8}$
$n = 1024, k = 524, t = 50$	$2^{80.7}$	$2^{70.6}$	$2^{65.5}$
$n = 1024, k = 654, t = 37$	$2^{84.1}$	$2^{73.4}$	$2^{68.3}$

Remark 7: If we allow 2 errors in the information window in Algorithm 2 (that means that the test at Step 1 involves 2 rows of matrix Z) then the overall work factor increases (for $n = 512, k = 256, t = 28$, we find $2^{42.7}$ instead of $2^{39.8}$).

5 Simulation results

We simulated Algorithm 2 on a Sparc station 2 which performs 28.5 Mips.

We made a great number of simulations for small lengths in order to get some statistical results on the number of required iterations:

	theorical average	experimental average	min	max	standart deviation	median
[128,64,13]	156	141	0	1480	153	93
[256,128,25]	6031	5518	1	40049	5635	3759

We can notice that the variance of the number of iterations is very high because of the random selection of the l and p columns at each iteration and that the median number of iterations is quite smaller than the average as it appears in Figure 2.

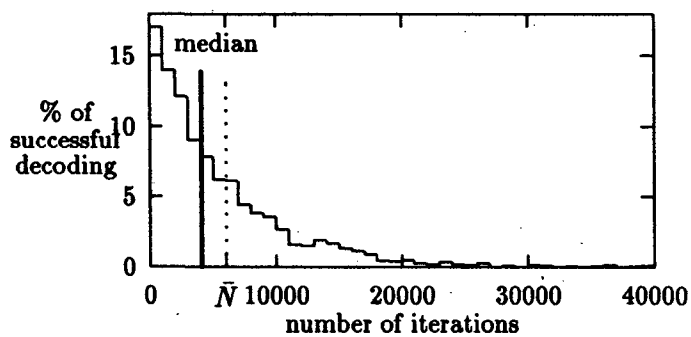

Figure 2: Distribution of the number of iterations for [256,128]-codes with $t = 12$

For $n = 512$, $k = 256$, $t = 28$ we made simulations for 5 different error patterns (with different generator matrices) and we succeeded in decoding 3 of them in less than 10 days (one in 2 days, one in 3 days and one in 10 days).

We observe that the running-time follows the previsions given in Figure 1 and confirms the validity of the previous theory.

Although the theorical average number of iterations is very high (191 millions) it is not necessary to perform so many executions in order to decode an important percentage of all messages. For example we have determined the average number of iterations required for decoding at least 1% of them: if you eliminate 64 randomly choosen columns of G you have more than one chance in 50 that you have eliminated no error positions. Then you can apply Algorithm 2 with the restricted [448,192,57]-linear code which requires 5.6 millions iterations in average; it represents 12 hours

on our system. Assuming that the median number of iterations is less than the average number we may suppose that 1% of all messages will be decoded with less than 5.6 millions iterations. With the corresponding simulation we have decoded 3 messages among 100 after 12 hours.

6 Conclusion

We have defined a method for exploring the set of all information windows without using the matrix inversion. Then we have suggested an improved decoding algorithm which is more efficient than the previously known best attack and we have shown that the use of [512,256]-random linear codes is not secure for a cryptosystem because we are able to decode in 12 hours more than 1% of all messages corrupted with a random error of weight 28.

Although that time depends exponentially on the size of the code our algorithm seems to be the best way for decoding any linear code without using the structure of the code. The work factor is still too high to attack McEliece's cryptosystem but since our method can easily be parallelized (because it only uses independent additions) and since the variance of the number of iterations is very high we could expect to speed up the execution by using a parallel computer.

Acknowledgements:

The second author wishes to thank Bernand Courteau and the University of Sherbrooke for many fruitful discussions during the elaboration of this algorithm.

References

[1] Adams(C.M.) and Meijer(H.), "Security-Related Comments Regarding McEliece's Public-Key Cryptosystem", *Proceeding of CRYPTO'87*, Lecture Notes in Computer Science 293, 224-230, 1987.

[2] Chabanne(H.) and Courteau(B.), "Application de la méthode de décodage itérative d'Omura à la cryptanalyse du système de McEliece", Rapport de l'Université de Sherbrooke 122, Canada, 1993.

[3] Chabaud(F.), "Asymptotic analysis of probabilistic algorithms for finding short codewords", *Proceeding of EUROCODE'92*, CISM Courses and Lectures 339, 175-183, 1992.

[4] Ferrand(D.), "Transformé de Möbius des codes", IRMAR de Rennes, France, 1993.

[5] Lee(P.J.) and Brickell(E.F.), "An Observation on the Security of McEliece's Public-Key Cryptosystem", *Proceeding of EUROCRYPT'88*, Lecture Notes in Computer Science 330, 275-280, 1988.

[6] Mac Williams(F.J.) and Sloane(N.J.A.), "The theory of error-correcting codes", North-Holland, Amsterdam-New-York-Oxford, 1978.

[7] Maurer(S.B.), "Matroid basis graph I", *J. Combin. Theory*, Series B 14, 216-240, 1973.

[8] Maurer(S.B.), "Matroid basis graph II", *J. Combin. Theory*, Series B 15, 121-145, 1973.

[9] McEliece(R.J.), "A Public-Key Cryptosystem Based on Algebraic Coding Theory", DSN Progress Report 42-44, Jet Propulsion Laboratory, 114-116, 1978.

[10] Omura(J.K.), "Iterative decoding of linear codes by a modulo-2 linear program", *Discrete Math.*, 3 (1972), 193-208, 1972.

Unité de Recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

Unité de Recherche INRIA Lorraine Technopôle de Nancy-Brabois - Campus Scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 VILLERS LES NANCY Cedex (France)
Unité de Recherche INRIA Rennes IRISA, Campus Universitaire de Beaulieu 35042 RENNES Cedex (France)
Unité de Recherche INRIA Rhône-Alpes 46, avenue Félix Viallet - 38031 GRENOBLE Cedex (France)
Unité de Recherche INRIA Sophia Antipolis 2004, route des Lucioles - B.P. 93 - 06902 SOPHIA ANTIPOLIS Cedex (France)

EDITEUR
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

ISSN 0249 - 6399