followed by addition which is completed in one clock unit time. The DFT is computed using a systolic cell described in [15]. For IBA-1 and IBA-2 the computation of the common connection polynomial is done by processing rows or columns serially. The computations of the connection polynomials for individual rows or columns in the case of Blahut's 2-D burst error correction are done concurrently. Except for the MSTD, the B-M algorithm works in the spectral domain. For the MSTD, the time-domain B-M algorithm is employed and in this case all computations of the B-M algorithm for rows or columns are done concurrently. To simplify the matters, conjugacy constraints and fast computation algorithms for DFT are not taken into account. Table I shows the complexity comparison.

Both IBA-1 and IBA-2 show improvements in computational savings and decoding delay over BA-1 and BA-2, respectively. Relative improvement in the case of IBA-1 is more apparent compared to the improvement in the case of IBA-2. The IBA-1 has the smallest delay of all and the least hardware requirement. Since the IBA-1 requires simple control circuitry, it is preferable for small values of t and n. The IBA-2 has the smallest number of computations of all and marginally higher delay than the IBA-1. Hardware complexity of the IBA-2 is greater compared to the IBA-1. Therefore, the IBA-2 is preferable for large values of t and n. The MSTD has a larger number of computations compared to the IBA-2 and the longest delay of all. The MSTD may be used for moderate values of t and n, if the decoding delay is not stringent. It is to be noted that as the MSTD has to do 1-D DFT computation of either all the rows or columns, the time-domain implementation of the B-M algorithm for 2-D BCH decoding is not as advantageous as the time-domain implementation in the 1-D case.

X. Conclusions

We have presented deconvolution viewpoint for studying DFT domain decoding algorithms and applied it to obtain an alternative exposition of decoding algorithms for 2-D BCH codes. Some modifications to efficiently implement Blahut's decoding algorithm for 2-D BCH codes are suggested. It is shown that the modified algorithm requires at most half the number of passes compared to Blahut's original decoding algorithm. Improved versions of Blahut's decoding algorithms are given for correction of random and burst errors. Error-correcting capability of the class of 2-D BCH codes is determined and it is shown that Blahut's decoding algorithms correct up to their error-correcting capability. We have also given mixed spectral and time-domain implementation of 2-D BCH decoding algorithms and compared various decoding algorithms for the class of 2-D BCH codes with respect to computation and implementation complexities.

Some specialized class of 2-D BCH codes along with their decoding algorithms based on the idea of this correspondence, are given in [8]. These results will be communicated separately in a forthcoming paper.

References

[1] R. T. Chien and W. Ng, "Dual product codes for correction of multiple low-density burst errors," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 672–677, Sept. 1973.

[2] R. E. Blahut, "Algebraic codes in the frequency domain," in *Algebraic Coding Theory and Applications*, G. Longo, Ed. Wien/New York: Springer-Verlag, 1979, pp. 448–494.

[3] ——, "Transform techniques for error control codes," *IBM. J. Res. Develop.*, vol. 23, no. 3, May 1979.

[4] ——, *Theory and Practice of Error Control Codes.* Reading, MA: Addison-Wesley, 1983.

[5] S. Sakata, "Synthesis of two-dimensional linear feedback shift registers," in *Applied Algebra, Algebraic algorithms and Error-Correcting Codes: Proc. AAECC-5* (Menorca 1987), L. Huguet and A. Poli, Eds. Berlin, Germany: Springer-Verlag, 1987, pp. 399–407.

[6] J. L. Massey and T. Schaub, "Linear complexity in coding theory," in *Coding Theory and Applications* (Lecture Notes in Computer Series, vol. 311), G. Cohen and P. G. Godelwski, Eds. Berlin, Germany: Springer-Verlag, 1988, pp. 19–32.

[7] R. Lidl and H. Niedrreiter, "Finite fields," in *Encyclopedia of Mathematics and its Applications*, vol. 20. New York: Cambridge Univ. Press, 1984.

[8] H. S. Madhusudhana, "New decoding algorithms for Reed–Muller and cyclic codes based on spectral domain deconvolution," Ph.D. dissertation, Dept. Elec. Eng., Indian Inst. Technol., Kanpur, Oct. 1992.

[9] H. Imai, "Two-dimensional burst-correcting codes," *Trans. Inst. Electron. Comm. Eng. Japan*, vol. 55-A, pp. 9–18, Aug. 1992.

[10] K. A. S. Abdel-Ghaffar, R. J. McEliece, and H. C. A Van Tilborg, "Two-dimensional burst identification codes and their use in burst correction," *IEEE Trans. Inform. Theory*, vol. 34, pp. 494–504, May 1988.

[11] I. M. Boyarinov, "Method of decoding direct sums of products of codes and its applications," *Probl. Pered. Inform.*, vol. 17, no. 2, pp. 39–51, Apr./June 1981.

[12] V. A. Zinov'ev and V. V. Zyalov, "Correction of error bursts and independent errors using generalized cascade codes," *Probl. Pered. Inform.*, vol. 15, no. 2, pp. 58–70, Apr./June 1979.

[13] R. L. Miller, "Minimal codes in abelian group algebra," *J. Comb. Theory, Ser. A26*, pp. 166–178, 1979.

[14] H. S. Madhusudhana and M. U. Siddiqi, "On Blahut's decoding algorithm for two-dimensional BCH codes," in *Proc. 1991 IEEE Int. Symp. on Information Theory* (Budapest, Hungary, June 24–28, 1991), p. 94.

[15] H. M. Shao and I. S. Reed, "On the VLSI design of a pipeline R-S decoder using systolic arrays," *IEEE Trans. Comput.*, vol. 37, pp. 1273–1280, Oct. 1988.

A New Algorithm for Finding Minimum-Weight Words in a Linear Code: Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511

Anne Canteaut and Florent Chabaud

***Abstract*—An algorithm for finding minimum-weight words in large linear codes is developed. It improves all previous attacks on the public-key cryptosystems based on codes and it notably points out some weaknesses in McEliece's cipher. We also determine with it the minimum distance of some BCH codes of length 511.**

***Index Terms*—BCH codes, decoding algorithm, error-correcting codes, McEliece's cryptosystem, minimum weight.**

I. Introduction

Although determining the weights of any linear binary code is an NP-complete problem [1], the general problem of finding a codeword of weight bounded by a prescribed integer in a linear code is not

Manuscript received August 28, 1995; revised October 29, 1996. The material in this correspondence was presented in part at the 5th IMA Conference on Coding and Cryptography, Cirencester, U.K., September 1995 and at the Mediterranean Workshop on Coding and Information Integrity, Palma, Spain, 1996.

A. Canteaut is with INRIA–Projet Codes, Domaine de Voluceau, 78153 Le Chesnay Cedex, France.

F. Chabaud is with LIENS, 75230 Paris Cedex 05, France, on leave from Centre d'Électronique de l'Armement.

Publisher Item Identifier S 0018-9448(98)00018-2.

proved to be NP-hard. It seems most unlikely that a polynomial-time algorithm for solving it exists. However, it does not imply that no general algorithm which runs faster than the exhaustive search can be discovered. Finding an efficient algorithm for computing a minimum-weight word in a linear code has in fact some important consequences both in cryptography and in coding theory. Such an algorithm actually constitutes an attack against a whole class of cryptosystems which rely on the hardness of decoding or on finding a minimum-weight codeword in a large linear code with no visible structure. The most famous are McEliece and Niederreiter public-key systems [2], [3]—which are equivalent from the security point of view—and the identification schemes proposed by Stern [4] and by Véron [5]. This class of public-key systems is at the moment the only alternative to the common algorithms based on number theory which resists the cryptanalysis. Twenty years after the fundamental paper of Diffie and Hellman, public-key cryptography has in fact become dangerously dependent on only two problems: integer factoring and discrete logarithm. Studying the security of these systems based on algebraic error-correcting codes therefore seems essential in order to anticipate a possible important progress in factoring methods, for example. Any improvement of the algorithms for finding a minimum-weight word in a general linear code consequently conditions the security of these cryptosystems and it delimits the parameters which could make them insecure. The motivations for searching for such an algorithm in coding theory are slightly different. This mainly aims at verifying or at establishing some conjectures on the minimum distance of some particular linear codes. For instance, there are several well-known families of codes for which only a lower bound on the minimum distance is known. It is then of great interest to determine whether such a bound is tight and to detect a specific behavior of some of these codes regarding their weight distribution. Among such families of codes we especially focus here on the primitive narrow-sense BCH codes.

In this correspondence we present a new probabilistic algorithm for finding minimum-weight words in any linear code. It associates a heuristic proposed by Stern and an iterative procedure stemming from linear programming. We give a very precise analysis of the complexity of this algorithm which enables us to optimize the parameters it depends upon. Hence our algorithm is to our knowledge the best procedure for decoding without using the structure of the code. Section II describes our algorithm for binary codes but it could be generalized to linear codes over GF(q) (see [6]). Using Markov chain theory we show in Section III how to compute the number of elementary operations it requires; this enables us to determine the parameters which minimize this theoretical running time. We then give in Section IV some experimental results for decoding $[256, 128]$-binary linear codes which validate the earlier theoretical approach. Section V is dedicated to the specific problems of decoding and finding a minimum-weight codeword in a random $[n, k]$-binary code. Two applications of our algorithm are finally discussed: we point out its consequences for the security of some cryptosystems based on error-correcting codes like McEliece public-key system, and we give new results for the true minimum distance of some narrow-sense BCH codes of length 511.

II. Description of the Algorithm

As usual $\mathrm{wt}(x)$ will denote the Hamming weight of the binary word x.

Let $\mathcal{C}$ be a linear binary code of length n, dimension k, and minimum distance d about which nothing is known but a generator matrix. We now develop an algorithm for finding a word of weight w in $\mathcal{C}$ where w is close to d.

This algorithm can also be used for decoding up to the correction capability $t = \lfloor \frac{d-1}{2} \rfloor$. If a message x is composed of a codeword corrupted by an error vector e of weight $w \leq t$, e can be recovered with this algorithm since it is the only minimum-weight word in the linear code $\mathcal{C} \oplus x$. Decoding an $[n, k]$-linear code then comes down to finding the minimum-weight codeword in an $[n, k+1]$-code.

Since the main motivation of such a research is to attack McEliece's cryptosystem, most of the previous works are described as decoding algorithms. But they can easily be transformed in order to solve the problem of finding a codeword of low weight. The first algorithm given by McEliece himself in his first security analysis of his cryptosystem [2] is equivalent to the information set decoding procedure. It was later improved by Lee and Brickell [7]. Independently, Leon [8] and Stern [9] also gave some algorithms for solving this problem.

A. The Probabilistic Method

Enumerating randomly selected codewords in hope that one of weight w will be found is obviously not a suitable algorithm because the probability that the weight of a random codeword will be w is very small. It is then necessary to bias this random selection by only examining codewords verifying a given property so that their weight will be *a priori* small, for example, codewords which vanish on a randomly chosen coordinate subset. The problem is therefore to find a tradeoff between the number of operations required for searching such particular codewords and the success probability, i.e., the probability that the weight of such a codeword will be w.

All algorithms for finding short codewords use therefore the same method: they are in fact generalizations of the well-known information set decoding method.

Let $N = \{1, \cdots, n\}$ be the set of all coordinates. For any subset I of N, $G = (V, W)_I$ denotes the decomposition of matrix G 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 1: Let I be a k-element subset of N. I is an information set for the code $\mathcal{C}$ if and only if $G = (Id_k, Z)_I$ is a systematic generator matrix for $\mathcal{C}$. The complementary set, $J = N \setminus I$, is called a redundancy set.

From now on we index the rows of Z with I since $G = (Id_k, Z)_I$ is a generator matrix for the code and we denote by Z^i the ith row of matrix Z.

The basic idea proposed by Lee and Brickell [7] consists in randomly selecting at each iteration an information set I and in examining all codewords having at most p nonzero bits in I, where the parameter p usually equals 1 or 2. These codewords actually correspond to the linear combinations of at most p rows of the corresponding systematic generator matrix $(Id_k, Z)_I$.

Instead of computing the weight of all these linear combinations, which is a time-consuming procedure, Leon [8] suggested that this algorithm should be first applied to a punctured code $\mathcal{C}'$ composed of the information set I and a selection L of σ redundant positions, where σ is small—no more than 20 positions for codes of length around 500 or 1000. When a low-weight codeword in this punctured code is found the total weight of the corresponding word of the initial code $\mathcal{C}$ is then computed.

The probabilistic algorithm proposed by Stern [9] is slightly different but it was shown to give the best results [10]. It also consists in randomly choosing at each iteration an information set I which is split into two parts I_1 and I_2 of same size, and a subset L of σ redundant positions. We only examine codewords c verifying the following property:

$$\mathrm{wt}(c_{|I_1}) = \mathrm{wt}(c_{|I_2}) = p \text{ and } \mathrm{wt}(c_{|L}) = 0 \qquad (1)$$

until we find such a particular codeword whose restriction on $J\backslash L$ has weight $w - 2p$.

All codewords which satisfy condition (1) can easily be constructed from the systematic generator matrix $G = (Id_k, Z)_I$:

- randomly split the rows of Z into two subsets Z_1 and Z_2 corresponding to I_1 and I_2;
- randomly select a σ-element subset L of the redundant set J;
- for each linear combination Λ_1 of p rows of matrix Z_1, compute $\Lambda_{1|L}$;
- for each linear combination Λ_2 of p rows of matrix Z_2, compute $\Lambda_{2|L}$;
- if $\Lambda_{1|L} = \Lambda_{2|L}$, check whether $\mathrm{wt}\,((\Lambda_1 + \Lambda_2)_{|J\backslash L}) = w - 2p$.

Both p and σ are parameters of the algorithm.

B. The Iterative Procedure

All the previous algorithms therefore explore a set of randomly selected information sets by performing at each iteration a Gaussian elimination on an $(n \times k)$-generator matrix. But computing the complexity of these algorithms points out that their most expensive step is the Gaussian elimination. In order to avoid this time-consuming procedure, we here propose to choose at each step the new information set by modifying only one element of the previous one. This method is analogous to the one used in the simplex method as suggested in [11]–[13].

Definition 2: Two information windows I and I' are close if and only if:

$$\exists \lambda \in I,\ \exists \mu \in N\backslash I,\ \text{such that } I' = (I\backslash\{\lambda\}) \cup \{\mu\}.$$

As any two information sets can be joined by a sequence of close information sets, we use this iterative method in order to find one which enables us to exhibit a codeword of weight w. The following proposition shows how to choose λ and μ such that I' is still an information set.

Proposition 1: Let I be an information set such that $G = (Id_k, Z)_I$ is a generator matrix for $\mathcal{C}$. Let be $\lambda \in I$, $\mu \in J$, and $I' = (I\backslash\{\lambda\}) \cup \{\mu\}$.
I' is an information set if and only if $z_{\lambda,\mu} = 1$, where $Z = (z_{i,j})_{i\in I, j\in J}$.

Proof: Since $G = (Id_k, Z)_I$, we have:

$$G_\mu = z_{\lambda,\mu} G_\lambda + \sum_{i\in I\backslash\{\lambda\}} z_{i,\mu} G_i.$$

As the columns indexed by I are linearly independent, G_μ and $(G_i)_{i\in I\backslash\{\lambda\}}$ are linearly independent if and only if $z_{\lambda,\mu} = 1$. □

As the probabilistic method only deals with the redundant part of the systematic generator matrix, we only need a procedure to be able to obtain the redundant matrix Z' corresponding to I' from Z.

Proposition 2: Let I and I' be two close information windows such that $I' = (I\backslash\{\lambda\}) \cup \{\mu\}$. Let $(Id_k, Z)_I$ and $(Id_k, Z')_{I'}$ be the corresponding systematic generator matrices. Then Z' is obtained from Z by

- $\forall j \in J' \qquad z'_{\mu,j} = z_{\lambda,j}$
- $\forall i \in I'\backslash\{\mu\}$,
  - $\forall j \in J'\backslash\{\lambda\} \qquad z'_{i,j} = z_{i,j} + z_{i,\mu} z_{\lambda,j}$
  - $z'_{i,\lambda} = z_{i,\mu}$.

Proof: As $I' = (I\backslash\{\lambda\}) \cup \{\mu\}$, $(Id_k, Z')_{I'}$ is obtained by exchanging the λth and μth columns of $(Id_k, Z)_I$. This can be done by a simple pivoting operation in position (λ, μ), i.e., by adding the λth row of matrix Z to all other rows Z^i when the corresponding element $z_{i,\mu}$ is not zero. □

C. Description of the Iterative Algorithm

Using this iterative procedure then leads to the following algorithm:

Initialization:
Randomly choose an information set I and apply a Gaussian elimination in order to obtain a systematic generator matrix $(Id_k, Z)_I$.
Until a codeword of weight w will be found:

- randomly split I in two subsets I_1 and I_2 where $|I_1| = \lfloor k/2 \rfloor$ and $|I_2| = \lceil k/2 \rceil$. The rows of Z are then split in two parts Z_1 and Z_2;
- randomly select a σ-element subset L of J;
- for each linear combination Λ_1 of p rows of matrix Z_1, compute $\Lambda_{1|L}$ and store all these values in a hash table with 2^σ entries;
- for each linear combination Λ_2 of p rows of matrix Z_2, compute $\Lambda_{2|L}$ and store all these values in a hash table with 2^σ entries;
- using the hash table consider all pairs of linear combinations (Λ_1, Λ_2) such that $\Lambda_{1|L} = \Lambda_{2|L}$ and check whether $\mathrm{wt}\,((\Lambda_1 + \Lambda_2)_{|J\backslash L}) = w - 2p$;
- randomly choose $\lambda \in I$ and $\mu \in J$. Replace I with $(I\backslash\{\lambda\}) \cup \{\mu\}$ by updating matrix Z according to the preceding proposition.

III. Theoretical Running Time

We give here an explicit and computable expression for the work factor of this algorithm, i.e., the average number of elementary operations it requires. This analysis is essential for finding the values of parameters p and σ which minimize the running time of the algorithm.

A. Modeling of the Algorithm by a Markov Chain

The average number of iterations performed by the algorithm is not the same as the one performed by the initial Stern's algorithm since the successive information sets are not independent anymore. Hence the algorithm must be modeled by a discrete-time stochastic process.

Let c be the codeword of weight w to recover and $\mathrm{supp}\,(c)$ its support. Let I be the information set and I_1, I_2, and L the other selections corresponding to the ith iteration. The ith iteration can then be represented by a random variable X_i which corresponds to the number of nonzero bits of c in I. This random variable then takes its values in the set $\{1, \cdots, w\}$. But if this number equals $2p$ we have to distinguish two cases depending of whether condition (1) is satisfied or not. The state space of the stochastic process $\{X_i\}_{i\in \boldsymbol{N}}$ is therefore

$$\mathcal{E} = \{1, \cdots, 2p-1\} \cup \{(2p)_S, (2p)_F\} \cup \{2p+1, \cdots, w\}$$

where

$X_i = u$ iff $|I \cap \mathrm{supp}\,(c)| = u, \quad \forall u \in \{1, \cdots, 2p-1\} \cup \{2p+1, \cdots, w\}$
$X_i = (2p)_F$ iff $|I \cap \mathrm{supp}\,(c)| = 2p$ and $(|I_1 \cap \mathrm{supp}\,(c)| \neq p$ or $|L \cap \mathrm{supp}\,(c)| \neq 0)$
$X_i = (2p)_S$ iff $|I_1 \cap \mathrm{supp}\,(c)| = |I_2 \cap \mathrm{supp}(c)| = p$ and $|L \cap \mathrm{supp}\,(c)| = 0$.

The success space is then $\mathcal{S} = \{(2p)_S\}$ and the failure space is $\mathcal{F} = \{1, \cdots, (2p)_F, \cdots, w\}$.

Definition 3: A stochastic process $\{X_i\}_{i\in N}$ is a *Markov chain* if the probability that it enters a certain state only depends on the last state it occupied.

A Markov chain $\{X_i\}_{i\in N}$ is *homogeneous* if for all states u and v, the conditional probability $\Pr\,[X_i = v / X_{i-1} = u]$ does not depend on i. This probability is then denoted by $P_{u,v}$ and, if the state space $\mathcal{E}$ is finite, the matrix $P = (P_{u,v})_{u,v\in\mathcal{E}}$ is called the *transition matrix* of the chain.

Proposition 3: The stochastic process $\{X_i\}_{i\in N}$ associated with the algorithm is an homogeneous Markov chain.

Proof: The selections I, I_1, I_2, and L corresponding to the ith iteration only depend on the previous information window since I_1, I_2, and L are randomly chosen. We then have for all i and for all $(u_0, u_1, \cdots, u_i) \in \mathcal{E}$

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

Furthermore, this probability does not depend on the iteration. Hence there exists a matrix P such that

$$\forall i \in \mathbb{N},\ \forall (u,v) \in \mathcal{E}^2, \qquad \Pr[X_i = v/X_{i-1} = u] = P_{u,v}. \quad \square$$

The Markov chain $\{X_i\}_{i\in N}$ is therefore completely determined by its initial probability vector $\pi_0 = (\Pr[X_0 = u])_{u\in\mathcal{E}}$ and its transition matrix P. Both of these quantities can be easily determined as two successive information sets differ from only one element.

Proposition 4: The transition matrix P of the homogeneous Markov chain associated with the algorithm is given by

$$\begin{aligned}
P_{u,u} &= \frac{k-u}{k} \times \frac{n-k-(w-u)}{n-k} + \frac{u}{k} \times \frac{w-u}{n-k}, \\
&\qquad \text{for all } u \in \mathcal{E}\backslash\{(2p)_S, (2p)_F\} \\
P_{u,u-1} &= \frac{u}{k} \times \frac{n-k-(w-u)}{n-k}, \qquad \text{for all } u \neq 2p+1 \\
P_{u,u+1} &= \frac{k-u}{k} \times \frac{w-u}{n-k}, \qquad \text{for all } u \neq 2p-1 \\
P_{u,v} &= 0, \qquad \text{for all } v \notin \{u-1, u, u+1\}
\end{aligned}$$

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

where

$$\beta = \Pr[X_i = (2p)_S/|I \cap \mathrm{supp}\,(e)| = 2p] = \frac{\binom{2p}{p}\binom{k-2p}{k/2-p}}{\binom{k}{k/2}} \frac{\binom{n-k-w+2p}{\sigma}}{\binom{n-k}{\sigma}}.$$

The initial probability vector π_0 is

$$\begin{aligned}
\pi_0(u) &= \frac{\binom{w}{u}\binom{n-w}{k-u}}{\binom{n}{k}}, \qquad \text{if } u \notin \{(2p)_F, (2p)_S\} \\
\pi_0((2p)_F) &= \frac{(1-\beta)\binom{w}{2p}\binom{n-w}{k-2p}}{\binom{n}{k}} \\
\pi_0((2p)_S) &= \frac{\beta\binom{w}{2p}\binom{n-w}{k-2p}}{\binom{n}{k}}.
\end{aligned}$$

The only persistent space of this Markov chain, i.e., a maximal state subset which cannot be left once it is entered, exactly corresponds to the success space $\mathcal{S}$. Since this subset contains only one state which is an absorbing state, i.e., a state which once entered is never left, this chain is by definition an absorbing chain.

A basic property of absorbing Markov chains with a finite state space is that, no matter where the process starts, the probability that the process is in an absorbing state after n steps tends to 1 as n tends to infinity. We then deduce that our algorithm converges.

1) Expected Number of Iterations: The absorbing chain property also enables us to compute the average number of iterations performed by the algorithm. For any finite absorbing chain we can define its fundamental matrix introduced by Kemeny and Snell [14].

Proposition 5 [14]: If $\{X_i\}_{i\in\mathbb{N}}$ is a finite absorbing Markov chain with transition matrix P, and Q is the substochastic matrix corresponding to transitions among the transient states—the nonpersistent states—i.e., $Q = (P_{u,v})_{u,v\in\mathcal{F}}$ then $(Id - Q)$ has an inverse R called the fundamental matrix of the chain and

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

The average number of iterations performed by the algorithm can then be deduced from the fundamental matrix.

Theorem 1: The expectation of the number of iterations N required until $\{X_i\}_{i\in\mathbb{N}}$ reaches the success state $(2p)_S$ is given by

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

where R is the corresponding fundamental matrix.

Proof:

$$\begin{aligned}
E(N) &= \sum_{n=0}^{\infty} n \Pr[X_n \in \mathcal{S} \text{ and } X_{n-1} \in \mathcal{F}] \\
&= \sum_{n=0}^{\infty} \sum_{m=0}^{n-1} \Pr[X_n \in \mathcal{S} \text{ and } X_{n-1} \in \mathcal{F}].
\end{aligned}$$

Applying Fubini's theorem, we get

$$\begin{aligned}
E(N) &= \sum_{m=0}^{\infty} \sum_{n=m+1}^{\infty} \Pr[X_n \in \mathcal{S} \text{ and } X_{n-1} \in \mathcal{F}] \\
&= \sum_{m=0}^{\infty} \Pr[X_m \in \mathcal{F}] \\
&= \sum_{m=0}^{\infty} \sum_{u\in\mathcal{F}} \sum_{v\in\mathcal{F}} \Pr[X_m = v/X_0 = u] \\
&= \sum_{u\in\mathcal{F}} \pi_0(u) \sum_{v\in\mathcal{F}} \sum_{m=0}^{\infty} (Q^m)_{u,v} \\
&= \sum_{u\in\mathcal{F}} \pi_0(u) \sum_{v\in\mathcal{F}} R_{u,v}. \quad \square
\end{aligned}$$

2) Variance of the Number of Iterations: The fundamental matrix also gives the variance of the number of iterations, which estimates the deviation from the average work factor of the effective computational time required by the algorithm.

Theorem 2: The variance of the number of iterations N required until $\{X_i\}_{i\in\mathbb{N}}$ reaches a success state is given by:

$$V(N) = \sum_{u\in\mathcal{F}} \pi_0(u) \sum_{v\in\mathcal{F}} (2R_{u,v} - \delta_{u,v}) E_v(N) - \left(\sum_{u\in\mathcal{F}} \pi_0(u) E_u(N)\right)^2$$

where $\delta_{i,j}$ is the Kronecker symbol and $E_u(N)$ is the average number of iterations performed by the process when it starts in state u, i.e.,

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

Proof: Let $E_u(N^2)$ denote the expectation of the random variable N^2 when the process starts in state u. We then have

$$\begin{aligned} E_u(N^2) &= \sum_{n=1}^{\infty} n^2 \Pr[X_n \in \mathcal{S} \text{ and } X_{n-1} \in \mathcal{F}/X_0 = u] \\ &= \sum_{v \in \mathcal{S}} P_{u,v} + \sum_{v \in \mathcal{F}} \sum_{n=1}^{\infty} P_{u,v}(n+1)^2 \\ &\quad \times \Pr[X_n \in \mathcal{S} \text{ and } X_{n-1} \in \mathcal{F}/X_0 = v] \\ &= \sum_{v \in \mathcal{S}} P_{u,v} + \sum_{v \in \mathcal{F}} \sum_{n=1}^{\infty} P_{u,v} E_v((N+1)^2) \\ &= \sum_{v \in \mathcal{S}} P_{u,v} + \sum_{v \in \mathcal{F}} \sum_{n=1}^{\infty} P_{u,v}(E_v(N^2) + 2E_v(N) + 1) \\ &= 1 + \sum_{v \in \mathcal{F}} \sum_{n=1}^{\infty} P_{u,v}(E_v(N^2) + 2E_v(N)). \end{aligned}$$

Let $\bar{x}$ and $\bar{n}$, respectively, denote the vectors $(E_u(N^2))_{u \in \mathcal{F}}$ and $(E_u(N))_{u \in \mathcal{F}}$. The matrix form of the last equation is then

$$\bar{x} = \bar{1} + Q\bar{x} + 2Q\bar{n}$$

where Q is the restriction of the transition matrix P to the failure states. Using the definition of the fundamental matrix, we therefore obtain

$$\begin{aligned} \bar{x} &= (Id - Q)^{-1}(2Q\bar{n} + \bar{1}) \\ &= 2RQ\bar{n} + \bar{n}. \end{aligned}$$

Since $R = \sum_{n=0}^{\infty} Q^n$, we have

$$RQ = \sum_{n=1}^{\infty} Q^n = R - Id$$

and $\bar{x} = (2R - Id)\bar{n}$. The final result is then deduced from the equation

$$E(N^2) = \pi_0[(2R - Id)\bar{n}]. \qquad \square$$

3) Distribution of the Iteration Number: Besides the average iteration number we often want to estimate the probability that the algorithm will succeed after a fixed number of iterations. But the approximation given by the Tchebychev's inequality is usually very rough. A much more precise evaluation is obtained by raising the transition matrix of the Markov chain to the corresponding power. We actually have:

Proposition 6: Let P be the transition matrix of the Markov chain associated with the algorithm. If $P = L^{-1}\Lambda L$, where Λ is a diagonal matrix, then the probability that the algorithm will succeed after N iterations is given by

$$\sum_{u \in \mathcal{E}} \pi_0(u)(L^{-1}\Lambda^N L)_{u,(2p)_S}.$$

B. Average Number of Operations by Iteration

We now give an explicit expression of the average number of operations performed at each iteration.

1) There are exactly $\binom{k/2}{p}$ linear combinations of p rows of matrix Z_1 (respectively Z_2); computing each of them on a σ-bit selection and putting it in the hash table requires $p\sigma$ binary additions.
2) The average number of collisions, i.e., the average number of pairs (Λ_1, Λ_2) such that $(\Lambda_1 + \Lambda_2)_{|L} = 0$ is equal to $\frac{\binom{k/2}{p}^2}{2^\sigma}$. For each collision we perform $2p - 1$ additions of $(n - k - \sigma)$-bit words for computing $(\Lambda_1 + \Lambda_2)_{|J \setminus L}$ and a weight checking.
3) We need $K(p\binom{k/2}{p} + 2^\sigma)$ more operations to performthe dynamic memory allocation where K is the size of a computer word ($K = 32$ or 64).
4) For updating matrix Z according to Proposition 2 we have to add row Z^λ to all other rows Z^i when $z_{i,\mu} = 1$. Assuming that the average weight of column Z_μ is $k/2$, the work factor involved in this procedure is $\frac{1}{2}k(n-k)$.

Hence the average number of elementary operations performed at each iteration is

$$\begin{aligned} \Omega_{p,\sigma} &= 2p\sigma\binom{k/2}{p} + 2p(n-k-\sigma)\frac{\binom{k/2}{p}^2}{2^\sigma} \\ &\quad + K\left(p\binom{k/2}{p} + 2^\sigma\right) + \frac{k(n-k)}{2}. \end{aligned} \qquad (2)$$

Proposition 7: Suppose that the number of codewords of weight w is $\mathcal{A}_w$. The overall work factor required by the algorithm is

$$W_{p,\sigma} = \frac{\Omega_{p,\sigma} E(N)}{\mathcal{A}_w} \qquad (3)$$

where $E(N)$ is given by Theorem 1 and $\Omega_{p,\sigma}$ by (2).

Since each term in the previous expression can be explicitly computed, we are now able to determine the parameters p and σ which minimize the work factor required by the algorithm when the size of the code and the weight w of the searched codeword are given. Such a theoretical expression of the work factor is commonly used to assess the efficiency of an algorithm and to decide whether a given problem is computationally feasible. It is also applied to the automatic optimization of the parameters. But computer architectures, implementations, and compiler optimizations introduce some variations in the effective number of elementary operations performed at each iteration that cannot be evaluated in a precise way. The sharpest optimization can then only be performed by replacing in (3) the theoretical value of $\Omega_{p,\sigma}$ by the effective average CPU time of an iteration.

IV. Experimental Results for Decoding Random $[256, 128]$-Binary Codes

In order to check the correctness of our modeling we have performaed a great number of simulations for a small problem: decoding a random $[256,\ 128,\ 29]$-binary code, i.e., recovering an error vector of weight 14. For each set of parameters 1000 computations have been made on a DEC alpha 3000/900 workstation. The results given in Table I confirm the validity of the previous theory: the experimental average number of iterations is very close to the result obtained by modeling with a Markov chain. Furthermore, the parameters which minimize the theoretical work factor are optimal in practice as well. Decoding a random $[256,\ 128,\ 29]$-code then requires around 2 s. In the same way, decoding a $[512,\ 256,\ 57]$-code requires around 9 h on our computer.

TABLE I
EXPERIMENTAL RESULTS FOR DECODING RANDOM [256, 128, 29]-BINARY CODES

parameters	theoretical work factor $\log_2(W)$	theoretical average iteration number	experimental average iteration number	deviation %	average CPU time (s)
$p = 1, \sigma = 5$	27.37	3961	4072	+2.80	2.76
$p = 1, \sigma = 6$	26.80	4045	3985	-1.48	2.11
$p = 1, \sigma = 7$	26.51	4139	4190	+1.23	2.07
$p = 1, \sigma = 8$	26.56	4244	4338	+2.21	2.13
$p = 1, \sigma = 9$	26.95	4362	4417	+1.26	2.90
$p = 2, \sigma = 10$	29.80	432	433	+0.35	13.84
$p = 2, \sigma = 11$	29.04	442	470	+6.51	13.00
$p = 2, \sigma = 12$	28.51	454	446	-1.76	12.13
$p = 2, \sigma = 13$	28.37	466	487	+4.51	17.70
$p = 2, \sigma = 14$	28.68	480	508	+5.83	29.40

TABLE II
OPTIMAL PARAMETERS FOR DECODING RANDOM $[n, n/2]$-BINARY CODES

code	[64,32,7]	[128,64,15]	[256,128,29]	[512,256,57]	[768,384,85]	[1024,512,113]	[1536,768,170]	[2048,1024,226]
t	3	7	14	28	42	56	84	112
optimal parameters	$p = 1$ $\sigma = 4$	$p = 1$ $\sigma = 6$	$p = 1$ $\sigma = 7$	$p = 1$ $\sigma = 9$	$p = 2$ $\sigma = 17$	$p = 2$ $\sigma = 18$	$p = 2$ $\sigma = 19$	$p = 2$ $\sigma = 20$
work factor	$2^{15.39}$	$2^{19.36}$	$2^{26.51}$	$2^{40.48}$	$2^{54.55}$	$2^{68.51}$	$2^{96.87}$	$2^{125.50}$

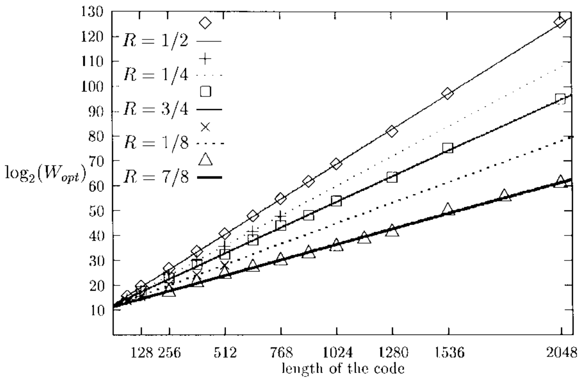

Fig. 1. Evolution of the theoretical work factor with optimized parameters for decoding random $[n, nR]$-binary codes.

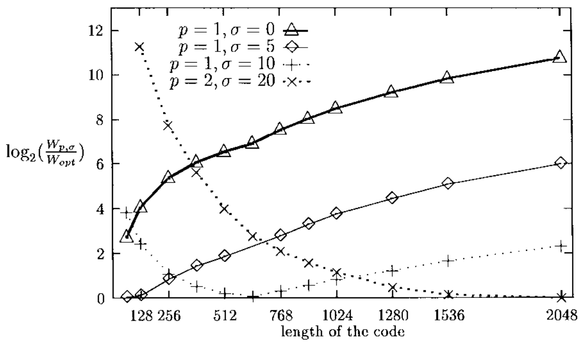

Fig. 2. Influence of parameters p and σ on the work factor for decoding random $[n, n/2]$-binary codes.

V. EXPLICIT APPROXIMATION OF THE WORK FACTOR FOR RANDOM BINARY CODES

Proposition 7 precisely estimates the computational cost for decoding and for finding a minimum-weight codeword once the parameters of the algorithm are optimized. But computing the average number of iterations with the transition matrix of the Markov chain is not very practical especially since this computation has to be performed for many different parameters. It would then be desirable to have an explicit formula which gives the work factor of the optimized algorithm as a function of the size of the code. Hence we establish here an approximation of this work factor for two classical problems: decoding a random linear binary code up to its correction capability and searching for a minimum-weight word in such a code. These results notably allow to immediately determine whether trying to decode or to find a minimum-weight word in a given code is realistic or not.

A. Decoding Random Linear Codes

We first consider the problem of decoding a random linear binary code of length n and dimension k up to its error-correcting capability, which is obtained by the Gilbert–Varshamov bound. Note that the code we consider for computing the work factor is an $[n, k + 1]$-code. Table II gives the optimal parameters of the algorithm and the corresponding work factors required for decoding some codes with expansion rate $R = k/n = 0.5$.

We see in Fig. 1 that, for a fixed expansion rate R, $\log_2(W)$ linearly depends on n when parameters p and σ are optimized and that the work factor can be written in the form $W_{\text{opt}} \simeq 2^{na(R)+b}$.

Fig. 2 shows how parameters p and σ act on the work factor involved in decoding a random $[n, n/2]$ code: if they are not optimized, $\log_2(W)$ does not linearly depend on n any more.

Furthermore, we see in Fig. 3 that $a(R)$ is close to the entropy function $H_2(R)$ multiplied by a fixed coefficient, where

$$H_2(x) = -x \log_2(x) - (1 - x) \log_2(1 - x).$$

Approximation 1: The theoretical work factor required for decoding a random $[n, nR]$-binary code can be approximated by the following formula:

$$W_{\text{opt}} \simeq 2^{naH_2(R)+b}, \qquad \text{where } a = 5.511\ 10^{-2} \text{ and } b = 12.$$

B. Finding Minimum-Weight Codewords

We now give an approximation of the theoretical complexity of the algorithm for recovering a word of weight d in a random $[n, k]$-binary code, where d equals the Gilbert–Varshamov bound. We assume that all these codes contain exactly one minimum-weight word. Table

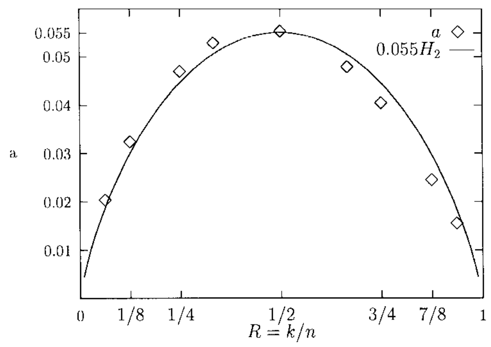

Fig. 3. Evolution of coefficient a versus $R = k/n$.

TABLE III
OPTIMAL PARAMETERS FOR FINDING A MINIMUM-WEIGHT WORD IN RANDOM $[n, n/2]$-BINARY CODES

code	[64,32,7]	[128,64,15]	[256,128,29]	[384,192,43]	[512,256,57]	[640,320,71]
optimal parameters	$p=1$ $\sigma=4$	$p=1$ $\sigma=5$	$p=1$ $\sigma=7$	$p=2$ $\sigma=14$	$p=2$ $\sigma=15$	$p=2$ $\sigma=16$
work factor	$2^{17.93}$	$2^{25.55}$	$2^{40.65}$	$2^{55.77}$	$2^{70.72}$	$2^{85.78}$

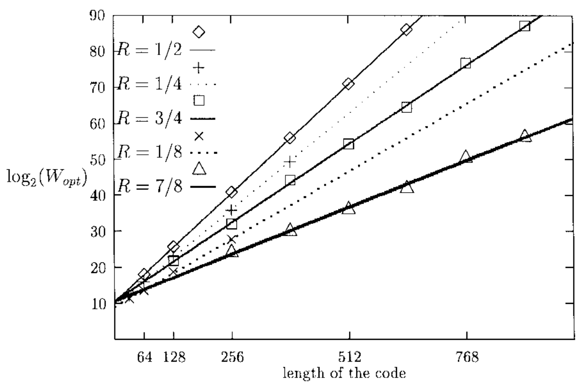

Fig. 4. Evolution of the theoretical work factor with optimized parameters for finding a minimum-weight word in random $[n, nR]$-binary codes.

III now gives the optimal parameters of the algorithm for finding a minimum-weight word in some random $[n, n/2]$-linear binary codes.

As for the decoding problem $\log_2(W_{\rm opt})$ linearly depends on n for a fixed expansion rate $R = k/n$ (see Fig. 4). If this work factor is written in the form $W_{\rm opt} \simeq 2^{nc(R)+d}$, Fig. 5 shows that $c(R)$ is closest to the translated entropy function $cH_2(R + R_0)$.

Approximation 2: The theoretical work factor required for finding a minimum-weight word in a random $[n, nR]$-binary code can be approximated by the following formula:

$$W_{\rm opt} \simeq 2^{ncH_2(R+R_0)+d}$$

where $c = 0.12$, $d = 10$, and $R_0 = 3.125\ 10^{-2}$.

VI. Application to the Cryptosystems Based on Error-Correcting Codes

Our algorithm constitutes the most efficient known attack against the class of public-key cryptosystems based on the hardness of decoding or on finding a minimum-weight word in a large code.

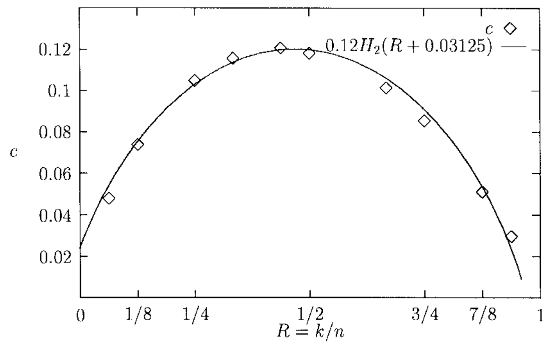

Fig. 5. Evolution of coefficient c versus $R = k/n$.

This class contains both McEliece's and Niederreiter's ciphers and some zero-knowledge identification schemes like the one proposed by Stern.

A. McEliece's and Niederreiter's Cryptosystems

McEliece's cryptosystem uses as a secret key a linear binary code chosen in a family Γ of $[n, k]$-linear codes with error-correcting capability t for which an efficient decoding algorithm is known. In his original paper [2], McEliece proposed to choose this secret code among the irreducible binary Goppa codes of length 1024, dimension 524, and minimum distance 101.

- **Private key:** it is composed of an $[n, k]$-linear binary code $\mathcal{C}$ chosen in the family Γ, a random $k \times k$ binary invertible matrix S, and a random $n \times n$ permutation matrix P.
- **Public key:** it consists of the $k \times n$ matrix G' defined by $G' = SGP$ where G is a generator matrix of the secret code $\mathcal{C}$. Matrix G' is then a generator matrix for another $[n, k]$-linear code $\mathcal{C}'$ which is equivalent to $\mathcal{C}$.
- **Encryption:** the ciphertext corresponding to the k-bit message m is $x = mG' + e$, where e is a random n-bit error vector of weight t.
- **Decryption:** the decryption procedure consists in computing $xP^{-1} = mSG + eP^{-1}$ and using a fast decoding algorithm for $\mathcal{C}$ to recover mS. The message is then given by $m = (mS)S^{-1}$.

A ciphertext in McEliece's cryptosystem then corresponds to a word of the public code $\mathcal{C}'$ with t corrupted positions.

Niederreiter proposed a dual version of this system [3] where the public key is a parity-check matrix H' of a code $\mathcal{C}'$ equivalent to the secret code. A plaintext m is here an n-bit vector of weight t and the associated ciphertext x corresponds to the syndrome of m relatively to the public code, $x = mH'^t$.

Niederreiter initially proposed to use as a secret code either the [104, 24, 32]-binary code obtained by concatenation of the [8, 4, 4]-binary extended Hamming code with a [13, 6, 8]-punctured Reed–Solomon code over GF(16), or a [30, 12, 19] Reed–Solomon code over GF(31). The parameters of both of these codes are obviously too small for the cryptosystem to be secure as shown by Brickell and Odlyzko [15].

1) Comparisons Between McEliece's and Niederreiter's Systems: McEliece's and Niederreiter's cryptosystems are actually equivalent from the security point of view when set up for corresponding choices of parameters [16]. But for given parameters Niederreiter's cipher presents many advantages.

- It allows a public key in systematic form at no cost in security, whereas this would reveal a part of the plaintext in McEliece's system. The public key in Niederreiter's system is then $(n-k)/n$ times smaller than in McEliece's version.
- The systematic form of the public matrix H' and the low weight of vector m significantly reduce the computational cost

TABLE IV
COMPARISON OF THE PERFORMANCE OF MCELIECE'S, NIEDERREITER'S, AND RSA PUBLIC-KEY CIPHERS

	McEliece [1024,524,101] binary code	Niederreiter [1024,524,101] binary code	RSA 1024-bit modulus public exponent = 17
public-key size	67,072 bytes	32,750 bytes	256 bytes
number of information bits transmitted per encryption	512	276	1024
transmission rate	51.17 %	56.81 %	100 %
number of binary operations performed by the encryption per information bit	514	50	2,402
number of binary operations performed by the decryption per information bit	5,140	7,863	738,112

involved in the multiplication of m by the public matrix. The encryption then requires around ten times less binary operations in Niederreiter's system than in McEliece's one (see Table IV).

- Its transmission rate, i.e., the number of information symbols divided by the number of transmitted symbols, equals

$$\frac{\log_2(\binom{n}{t})}{n-k}$$

whereas it equals k/n for McEliece's system. For $[1024, 524, 101]$-binary codes its value is then 56.8% instead of 51.2%.

- Another disadvantage of McEliece's cryptosystem is that it is easy to recover the plaintext if it has been encrypted twice with the same key (see [17]). Niederreiter's cipher is, on the contrary, deterministic since encrypting a given plaintext always leads to the same ciphertext.

A precise evaluation of the complexity of the encryption and decryption procedures for both of these systems is given in [17, ch. 2]. Table IV sums up their characteristics when they both use $[1024, 524, 101]$-binary codes. We give for information the values corresponding to the RSA system with a 1024-bit modulus $n = pq$ when the public exponent is 17—we here suppose that RSA encryption and decryption uses Karatsuba's method for large integer multiplication. These results finally show that it is preferable to use the version proposed by Niederreiter. They also point out that this public-key system runs much faster than the RSA. Its main disadvantages are the size of the public key and the lack of related signature scheme.

2) Cryptanalysis Methods: There are mainly two guidelines to cryptanalyze McEliece's cryptosystem:

- recover the original structure of the secret code from a generator (or parity-check) matrix of an equivalent code;
- decode the public code which has no visible structure.

The first class of attacks imposes some conditions on the family of secret codes Γ. It must in fact satisfy the following properties.

1) For given length, dimension, and minimal distance, the family Γ is large enough to avoid any enumeration.
2) An efficient decoding algorithm is known for this family.
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 fast decoding algorithm requires some parameters of the secret code besides a generator matrix G'.

The first condition on the size of Γ aims at protecting the system from the attack which consists in enumerating all the elements of Γ until a code equivalent to the public code is found. This can be performed with an algorithm due to Sendrier [18] that is able to determine from two generator matrices whether they correspond to equivalent codes and then to recover the permutation. This algorithm can be applied when the automorphism group of the code is trivial. Finding the permutation between two $[1000, 500]$-equivalent binary codes then requires around 2 s on a workstation DEC 500/266 and noting that the codes are not equivalent is even faster.

The third condition is the most restrictive: it actually dismisses many families of codes. For example, generalized Reed–Solomon codes are not convenient because their structure can be recovered using Sidelnikov–Shestakov algorithm [19]; concatenated codes which were initially suggested by Niederreiter are not appropriated either [20], [21]. But the family of irreducible 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 permuted generator matrix. This class can even be extended to all $[1024, 524, 101]$-binary Goppa codes defined by a monic square-free polynomial of degree 50 in GF$(1024)[X]$ which has no root in GF(1024). The cardinality of Γ is then $2^{498.5}$.

In the case the used family of codes satisfies the above properties, the equivalent code $\mathcal{C}'$ defined by the public key presents no particular structure; recovering a plaintext from the corresponding ciphertext then comes down to decoding any linear code.

B. Stern's Public-Key Identification Scheme

Stern presented at Crypto'93 [4] a public-key identification scheme which relies on the hardness of finding a small-weight codeword of a 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. The minimal parameters proposed by Stern are $n = 512$, $k = 256$, and $w = 56$.

Véron [5] also proposed a dual version of this scheme similar to McEliece's original approach: it uses a generator matrix of the code instead of a parity-check matrix. He then suggested a new choice of the parameters in order to reduce the number of transmitted bits: $n = 512$, $k = 120$, and $w = 114$.

C. Work Factor Required by Our Algorithm for the Cryptanalysis

Table V gives the optimal parameters and the number of binary operations involved in an attack of the previous cryptosystems. Cryptanalyzing McEliece's cipher with its original parameters then requires $2^{64.2}$ binary operations. This new attack is certainly still

TABLE V
WORK FACTOR REQUIRED FOR CRYPTANALYZING SOME PUBLIC-KEY SYSTEMS BASED ON ERROR-CORRECTING CODES

cryptosystem	McEliece	Stern	Véron
code	[1024,524]	[512,256]	[512,120]
w	50	56	114
optimal parameters	$p=2$ $\sigma=18$	$p=2$ $\sigma=15$	$p=2$ $\sigma=13$
average number of iterations	$9.85\ 10^{11}$	$2.16\ 10^{14}$	$1.74\ 10^{12}$
standard deviation of the number of iterations	$9.85\ 10^{11}$	$2.16\ 10^{14}$	$1.74\ 10^{12}$
work factor	$2^{64.2}$	$2^{69.9}$	$2^{61.2}$

unfeasible but it runs 128 times faster than Lee–Brickell's attack [7]. These results also show that reducing the size of the public key by using some codes of length less than 1024 is not conceivable: decoding a $[512,\ 260]$-binary Goppa code up to its error-correcting capability, for instance, requires only $2^{40.1}$ binary operations.

A simple method for speeding up the cryptanalysis is to parallelize the attack by performing several decodings together as suggested in [10] and [22]. But the more decodings are performed together, the more operations have to be done in each iteration. The obtained gain is then not very important. Another approach consists in parallelizing the algorithm itself, as van Tilburg does. But this parallelization is largely inefficient. For example, the cryptanalysis of Stern's identification scheme using van Tilburg's sequential algorithm has an average number of iterations of $2^{57.0}$, and an estimated work factor of $2^{72.9}$ [22, p. 93]. The parallelized instance of the same algorithm on a machine with 2^{31} dedicated circuits (which is the optimal number of circuits) needs $2^{43.5}$ iterations and $2^{58.5}$ operations [22, p. 107]. But distributing the sequential algorithm on 2^{14} computers would lead to the same result. With our algorithm, a network of 3000 computers yields a better result.

However, the standard deviation of the number of iterations involved in cryptanalyzing all these systems roughly equals its average. This spread implies that an unfeasible average work factor is not sufficient to guarantee that these cryptosystems are secure: it is necessary to estimate the probability that our algorithm will be successful after a feasible number of iterations. This can be done by raising the transition matrix of the associated Markov chain to the corresponding power as described in Proposition 6. We then obtain that the work factor required for decoding a $[1024,\ 524,\ 101]$-binary code up to its error-correcting capability only represents 69% of the average work factor. And if the work factor is limited to 2^{51}, i.e., to 10^8 iterations, the probability that a message in McEliece's cipher will be decrypt is 10^{-4}. Since 1000 iterations of the optimized algorithm are performed in 10 min on a workstation DEC alpha 500/266, decrypting one message out of 10 000 requires two months and 25 days with ten such computers (see Fig. 6). The proportion of decrypted messages in a reasonable time is therefore relatively high as long as the enemy has a few ten fast workstations.

A similar study shows that the parameters proposed in Stern's identification scheme make it much more secure (see Fig. 7). An eleven-month computation time on ten DEC alpha 500/266 enables us to recover the secret key of a user in only one case out of 100 000. This only implies that the lifetime of the keys must be less than one year. The parameters proposed by Véron significantly reduce the number of transmitted bits in each identification procedure but they impose a much shorter lifetime of the keys since 56 days on ten of our workstations are sufficient to find the secret key of a user with a probability greater than $1/3500$.

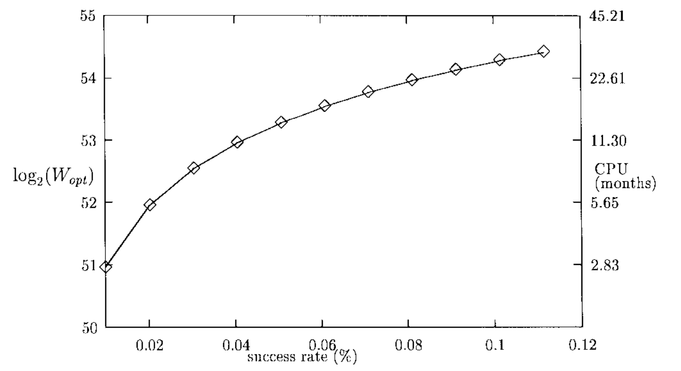

Fig. 6. Computational effort required for cryptanalyzing McEliece's cryptosystem as a function of the rate of messages successfully decrypted: the CPU time is given for ten workstations DEC alpha 500/266 in parallel.

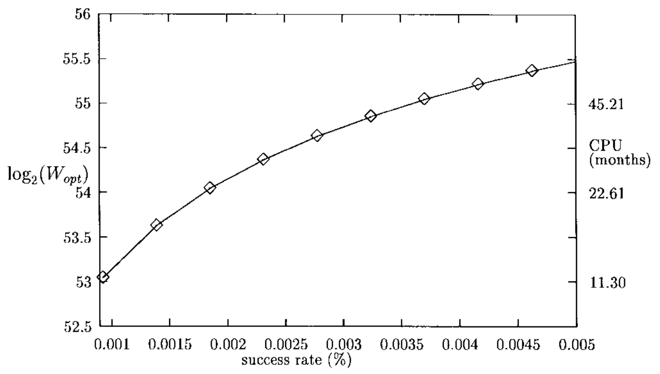

Fig. 7. Computational effort required for cryptanalyzing Stern's identification scheme as a function of the success rate: the CPU time is given for ten workstations DEC˜alpha 500/266 in parallel.

D. Partial Attacks on McEliece's and Niederreiter's Cryptosystems

McEliece's and Niederreiter's cryptosystems otherwise present some weaknesses since the knowledge of a small number of bits of the plaintext is sufficient to recover it in its entirety. The knowledge of some plaintext bits in McEliece's cipher allows to accordingly reduce the dimension of the code we consider in the attack. If we assume that 2^{50} binary operations is a feasible work factor, it is then possible to decode up to distance 50 a $[1024,\ 404]$-binary code with our algorithm. This means that the knowledge of 120 plaintext bits (i.e., 23% of the plaintext) is sufficient to recover the whole plaintext in a reasonable time.

A similar attack on Niederreiter's cryptosystem consists in assuming that some error positions are known by the enemy. The problem is then to determine the distance up to which a $[1024,\ 524]$-binary code can be decoded. Table VI shows that the knowledge of 15 error positions out of the 50 introduced in McEliece's and Niederreiter's systems enables us to recover the plaintext. This small proportion notably implies that generating the error vector with a noisy channel is insecure if this provides some errors whose weight is too small.

E. Optimization of the Parameters Used in McEliece's Cryptosystem

Adams and Meijer [23] have already noticed that using binary Goppa codes of length 1024 whose dimension is 524 as proposed by McEliece did not maximize the work factor of Lee–Brickell's attack. We now refine these parameters and we replace them by those which maximize the work factor of our algorithm, i.e., $n = 1024,\ k =$

TABLE VI
ACCESSIBLE DECODING DISTANCE

Code	[1024,524]	[512,260]
error-correcting capability	50	28
accessible decoding distance	35	37
average work factor	$p=2$ $\sigma=18$ $2^{49.9}$	$p=2$ $\sigma=15$ $2^{50.1}$

614, $t = 41$. The corresponding average work factor of the attack is then $2^{66.0}$.

As noticed in Section V, the maximum work factor required for finding a minimum-weight word in a binary code is maximum when the expansion rate of the code equals $1/2$. The parameters proposed by Stern are therefore optimal from this point of view.

VII. True Minimum-Distance of Binary Narrow-Sense BCH Codes of Length 511

In this section we apply our algorithm to the narrow-sense BCH codes of length 511 since only a lower bound on the minimum distance is known for this famous class of cyclic codes. These codes have been intensively studied since 1959 and some sufficient conditions for this lower bound to be reached have been exhibited. But there is still no general method able to find their true minimum distance. Quite recently, Augot, Charpin, and Sendrier [24] completed the table which gives the minimum distance of BCH codes of length 255, but for the length 511 the true minimum distance of 12 narrow-sense BCH codes is still unknown. In this context improving the knowledge of the minimum distance of relatively short BCH codes is a challenge. This is then a good way for testing the efficiency of such and such algorithm. But the main interest of any new numerical result is to suggest some important conjectures.

We here only consider primitive binary narrow-sense BCH codes. Let $q = 2^m$, $n = 2^m - 1$, and α be a primitive nth root of unity in GF(q).

Definition 4: The primitive binary narrow-sense BCH code of length $n = 2^m - 1$ and designed distance δ, denoted by $B(n, \delta)$, is the largest binary cyclic code of length n having zeros

$$\alpha, \alpha^2, \ldots, \alpha^{\delta-1}.$$

This construction immediately leads to a bound on the minimum distance d of $B(n, \delta)$, named the BCH-bound: $d \geq \delta$. This bound is actually very strong since it is generally reached: the smallest primitive narrow-sense BCH code whose minimum distance is greater than its designed distance is $B(127, 29)$ and it is the only one for this length.

It is well known that the automorphism group of an extended primitive binary BCH code contains the affine group on GF(2^m); we then have the following proposition.

Proposition 8: Let $(A_0, \cdots, A_n)$ be the weight distribution of a primitive binary BCH code of length n. Then we have for all index j

$$(n + 1 - 2j)A_{2j-1} = 2jA_{2j}.$$

This notably implies that if $B(2^m - 1, \delta)$ contains a word of even weight w then it contains a word of weight $w - 1$.

From the particular structure of BCH codes we can deduce that the designed distance is reached for several infinite classes of codes.

- Farr [25, p. 259] deduced from the Hamming bound that the minimum distance of $B(2^m - 1, 2t + 1)$ equals its designed distance if
$$\sum_{i=0}^{t+1} \binom{2^m - 1}{i} > 2^{mt}.$$
- If the minimum distance of $B(n, \delta)$ equals its designed distance then the BCH bound is reached for all $B(hn, h\delta)$ [26].
- Using the inclusion relations between BCH codes and punctured Reed–Muller codes, Kasami and Lin [27] proved that for all $1 \leq i \leq m - s - 2$, where $0 \leq s \leq m - 2i$, the intersection of the BCH code of length $2^m - 1$ and designed distance $\delta = 2^{m-s-1} - 2^{m-s-i-1} - 1$ with the punctured Reed–Muller code of order $s + 2$ contains a word of weight δ.
- Helgert and Stinaff [28] exhibited a codeword of weight δ or $\delta + 1$ in some shortened codes of $B(n, \delta)$.
- Augot and Sendrier [29] found some particular codewords of weight δ or $\delta + 1$ among the idempotents of $B(n, \delta)$, i.e., the words whose coefficients of the locator polynomial lay in GF(2).

Some methods also allow to show that the minimum distance of some BCH codes is greater than its designed distance:

- Kasami and Tokura [30] combined the inclusion of some BCH codes in a punctured Reed-Muller code and the divisibility of the weights of Reed-Muller codes. They therefore proved that some BCH codes have minimum distance $\delta + 2$ and $\delta + 4$.
- Augot, Charpin and Sendrier [24] found some cases where the BCH bound is not reached by exhibiting some contradictions in the Newton identities.

But for binary primitive BCH codes it is usually conjectured that the true minimum distance does not exceed $\delta + 4$.

After [24] the minimum distance was still unknown for 12 narrow-sense BCH codes of length 511. We have then tried to find with our algorithm a word of weight equal to the designed distance δ or to $\delta + 1$ in these codes and we have obtained the following result.

Theorem 3: The minimum distance of the narrow-sense BCH codes of length 511 with designed distance $\delta \in \{29, 37, 41, 43, 51, 87\}$ equals δ.

Proof: In each of these codes we found a word of weight $\delta + 1$. Finding a codeword of weight $\delta + 1$ is in fact easier than finding one of weight δ since the number of codewords of weight $\delta + 1$ is $(n - \delta)/(\delta + 1)$ times greater than the number of codewords of weight δ. Let α be a primitive element in GF(2^9) defined by $\alpha^9 + \alpha^4 + 1 = 0$. The support of these codewords consists of the values α^i for the following exponents.

- $\delta = 29$:
$$(26, 31, 38, 51, 64, 72, 112, 126, 139, 142, 157, 188, 222, 227, 265, 270, 301, 306, 307, 317, 347, 354, 368, 369, 412, 415, 423, 431, 494, 498)$$
- $\delta = 37$:
$$(4, 13, 27, 48, 56, 94, 102, 103, 115, 118, 132, 149, 152, 159, 197, 202, 215, 232, 240, 249, 250, 251, 290, 324, 327, 349, 359, 360, 367, 383, 396, 423, 461, 493, 494, 499, 504, 509)$$
- $\delta = 41$:
$$(9, 20, 30, 37, 38, 42, 43, 53, 66, 68, 83, 93, 95, 106, 108, 110, 111, 175, 185, 202, 234, 250, 262, 270, 321, 342, 362, 363 379, 382, 385, 401, 402, 410, 426, 436, 462, 467, 478, 482, 499, 507)$$

TABLE VII
BCH CODES OF LENGTH 511

n	k	δ	d	argument	in	n	k	δ	d		argument	in
511	502	3	3	H	[31]	511	241	73	73		PS(7,1)	[26]
	493	5	5	H	[31]		238	75	≥ 75		-	-
	484	7	7	H	[31]		229	77	≥ 77		-	-
	475	9	9	H	[31]		220	79	79		ID	[24]
	466	11	11	RM(6)	[27]		211	83	83		ID	[24]
	457	13	13	R	[28]		202	85	≥ 85		-	-
	448	15	15	RM(15)	[27]		193	87	87		**	**
	439	17	17	ES	[24]		184	91	91		ID	[24]
	430	19	19	ID	[24]		175	93	95	#	4-DI	[30]
	421	21	21	PS(73,3)	[26]		166	95	95		RM(3)	[27]
	412	23	23	RM(5)	[27]		157	103	103		ID	[24]
	403	25	25	R	[28]		148	107	≥ 107		-	-
	394	27	27	RM(5)	[27]		139	109	111	#	4-DI	[30]
	385	29	29	**	**		130	111	111		RM(3)	[27]
	376	31	31	RM(4)	[27]		121	117	119	#	4-DI	[30]
	367	35	35	PS(73,5)	[26]		112	119	119		RM(3)	[27]
	358	37	37	**	**		103	123	127	##	NI	[24]
	349	39	39	ID	[24]		94	125	127	#	4-DI	[30]
	340	41	41	**	**		85	127	127		RM(2)	[27]
	331	43	43	**	**		76	171	171		ES	[24]
	322	45	45	ID	[24]		67	175	175		ES	[24]
	313	47	47	RM(4)	[27]		58	183	183		ES	[24]
	304	51	51	**	**		49	187	187		ES	[24]
	295	53	53	NI	[24]		40	191	191		RM(2)	[27]
	286	55	55	RM(4)	[27]		31	219	219		PS(7,3)	[26]
	277	57	57	ID	[24]		28	223	223		RM(2)	[27]
	268	59	≥ 59	-	-		19	239	239		RM(2)	[27]
	259	61	≥ 61	-	-		10	255	255		RM(1)	[27]
	250	63	63	RM(3)	[27]							

#	$d = \delta + 2$
##	$d = \delta + 4$
**	new result
H	Hamming bound
ES	exhaustive search
ID	idempotent
NI	contradiction in Newton's identities
RM(s)	intersection with the punctured sth-order Reed-Muller code
R	code shortening
PS(n_2, δ_2)	product subcode: $n = n_1 n_2$, $\delta = n_1 \delta_2$ where $B(n_2, \delta_2)$ has minimum distance equal to δ_2
4-DI	4-divisibility of RM(4) since $B(511, \delta)$ is included in the punctured Reed-Muller code of order 4 for all $\delta > 85$

- $\delta = 43$:

  (0, 16, 35, 38, 56, 57, 58, 80, 82, 87,
  115, 134, 147, 148, 156, 165, 167, 190, 196,
  206, 229, 240, 242, 258, 269, 284, 295, 296,
  309, 317, 321, 322, 324, 325, 326, 361, 375,
  394, 405, 418, 429, 444, 460, 492)

- $\delta = 51$:

  (6, 10, 11, 17, 22, 54, 57, 64, 76, 79,
  85, 87, 93, 97, 101, 121, 122, 139, 140, 144,
  154, 171, 177, 182, 198, 258, 287, 290, 294, 299,
  309, 313, 333, 335, 350, 359, 361, 369, 370, 371,
  395, 399, 405, 412, 435, 437, 452, 469, 474, 488,
  491, 508)

- $\delta = 87$:

  (18, 19, 23, 25, 27, 43, 50, 51, 64, 70,
  73, 77, 81, 88, 96, 101, 102, 116, 117, 143,
  146, 152, 158, 163, 165, 166, 173, 179, 192, 193,
  195, 197, 199, 203, 210, 212, 225, 230, 240, 244,
  252, 263, 272, 283, 287, 290, 292, 293, 295, 297,
  301, 306, 320, 323, 327, 330, 339, 353, 361, 382,
  385, 392, 394, 400, 411, 414, 417, 421, 432, 436,
  446, 453, 459, 461, 466, 474, 475, 476, 480, 481,
  483, 487, 489, 492, 503, 504, 505, 510).

Table VII gives the list of all narrow-sense BCH codes of length 511, their minimum distance, and the way they were found.

As shown in Section V, the computation time for finding a minimum-weight codeword increases when the expansion rate of the code is closer to $1/2$. Finding a word of weight $\delta + 1$ on a DEC alpha 3000/900 workstation actually required less than 1 min for $B(511, 29)$, three days for $B(511, 37)$, and between one and three weeks for $\delta \in \{41, 43, 87\}$. A word of weight 52 in $B(511, 52)$ was found after 50 days on 35 SPARC 5 workstations. A comparison of these computation times with the theoretical work factor of the algorithm points out that the number of minimum-weight words in these codes is very high. For instance only 35 377 iterations were performed for finding a word of weight 30 in $B(511, 29)$ whereas 7.10^{15} would be necessary if this code contained only one minimum-weight word.

A second remark is that we did not succeed in finding a word of weight 107 or 108 in $B(511, 107)$ after 3 months whereas the algorithm should run 10 times faster than for $B(511, 87)$. We then conclude that either the BCH bound is not reached for this code or the number of minimum-weight codewords is very small. This second assumption is moreover supported by a result due to Augot *et al.* [24] proving that the minimum-weight codewords of $B(2^m - 1, 2^{m-2} - 1)$ are those of the punctured Reed–Muller code of same length and order 2. This implies that the minimum-weight codewords in a primitive narrow-sense BCH code become scarcer when the designed distance is close to $2^{m-2} - 1$.

These results finally suggest that the only BCH codes of length 511 for which the BCH bound may be not reached are those whose designed distance is close to 127.

Acknowledgment

The authors wish to thank H. Chabanne who initiated this work and N. Sendrier for enriching discussions. They would also like to thank the anonymous referees for many helpful suggestions which considerably improved this paper.

References

[1] E. Berlekamp, R. McEliece, and H. van Tilborg, “On the inherentintractability of certain coding problems,” *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 384–386, 1978.
[2] R. McEliece, “A public-key cryptosystem based on algebraic coding theory,” Jet Propulsion Lab. DSN Progress Rep., pp. 114–116, 1978.
[3] H. Niederreiter, “Knapsack-type cryptosystems and algebraic coding theory,” *Probl. Contr. and Inform. Theory*, vol. 15, pp. 159–166, 1986.
[4] J. Stern, “A new identification scheme based on syndrome decoding,” in *Advances in Cryptology—CRYPTO’93*, D. Stinson, Ed. New York: Springer-Verlag, 1993, pp. 13–21.
[5] P. Véron, “Problème SD, Opérateur Trace, schémas d’identification et codes de Goppa,” Ph.D. dissertation, Univ. de Toulon et du Var, 1995.
[6] F. Chabaud, Recherche de performance dans l’algorithmique des corps-finis. Applications à la cryptographie,” Ph.D. dissertation, Ecole Polytechnique, 1996.
[7] P. Lee and E. Brickell, “An observation on the security of McEliece’s public-key cryptosystem,” in *Advances in Cryptology—EUROCRYPT ’88*, C. Günter, Ed. New York: Springer-Verlag, 1988, pp. 275–280.
[8] J. Leon, “A probabilistic algorithm for computing minimum weights of large error-correcting codes,” *IEEE Trans. Inform. Theory*, vol. 34, pp. 1354–1359, 1988.
[9] J. Stern, “A method for finding codewords of small weight,” in *Coding Theory and Applications*, G. Cohen and J. Wolfmann, Eds. New York: Springer-Verlag, 1989, pp. 106–113.
[10] F. Chabaud, “On the security of some cryptosystems based onerror-correcting codes,” in *Advances in Cryptology—EUROCRYPT’94*, A. De Santis, Ed. New York: Springer-Verlag, 1994, pp. 131–139.
[11] J. Omura, “Iterative decoding of linear codes by a modulo-2 linearprogramm,” *Discr. Math.*, no. 3, pp. 193–208, 1972.
[12] J. van Tilburg, “On the McEliece public-key cryptosystem,” in *Advances in Cryptology—CRYPTO’88*, S. Goldwasser, Ed. New York: Springer-Verlag, 1988, pp. 119–131.
[13] A. Canteaut and H. Chabanne, “A further improvement of the work factor in an attempt at breaking McEliece’s cryptosystem,” in *EUROCODE 94—Livre des Résumés*, P. Charpin, Ed., INRIA, 1994.
[14] J. Kemeny and J. Snell, *Finite Markov Chains*. New York: Springer-Verlag, 1960.
[15] E. Brickell and A. Odlyzko, “Cryptanalysis: A survey of recent results,” in *Comtemporary Cryptology—The Science of Information Integrity*, G. Simmons, Ed. New York: IEEE Press, 1992, pp. 501–540.
[16] Y. Li, R. Deng, and X. Wang, “On the equivalence of McEliece’s and Niederreiter’s public-key cryptosystems,” *IEEE Trans. Inform. Theory*, vol. 40, pp. 271–273, 1994.
[17] A. Canteaut, “Attaques de cryptosystèmes à mots de poids faible et construction de fonctions t-résilientes,” Ph.D. dissertation, Univ. Paris 6, 1996.
[18] N. Sendrier, “An algorithm for finding the permutation between two equivalent binary codes,” Tech. Rep. RR-2853, INRIA, Apr. 1996.
[19] V. Sidelnikov and S. Shestakov, “On cryptosystems based on generalized Reed-Solomon codes,” *Diskret. Mat.*, vol. 4, pp. 57–63, 1992.
[20] N. Sendrier, “On the structure of a randomly permuted concatenated code,” in *EUROCODE 94—Livre des résumés*, P. Charpin, Ed., INRIA, 1994, pp. 169–173.
[21] N. Sendrier, “On the structure of a randomly permuted concatenated code,” Tech. Rep. RR-2460, INRIA, Jan. 1995.
[22] J. van Tilburg, “Security-analysis of a class of cryptosystems based on linear error-correcting codes,” Ph.D. dissertation, Tech. Univ. Eindhoven, Eindhoven, The Netherlands, 1994.
[23] C. Adams and H. Meijer, “Security-related comments regarding McEliece’s public-key cryptosystem,” in *Advances in Cryptology—CRYPTO’87*, C. Pomerance, Ed. New York: Springer-Verlag, 1987, pp. 224–228.
[24] D. Augot, P. Charpin, and N. Sendrier, “Studying the locator polynomial of minimum weight codewords of BCH codes,” *IEEE Trans. Inform. Theory*, vol. 38, pp. 960–973, 1992.
[25] F. MacWilliams and N. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1977.
[26] T. Kasami, S. Lin, and W. Peterson, “Some results on cyclic codes which are invariant under the affine group,” Tech. Rep. AFCRL-66-622, Air Force Cambridge Res. Labs., Bedford, MA, 1966.
[27] T. Kasami and S. Lin, “Some results on the minimum weight of primitive BCH codes,” *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 824–825, 1972.
[28] H. Helgert and R. Stinaff, “Shortened BCH codes,” *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 818–820, 1973.
[29] D. Augot and N. Sendrier, “Idempotents and the BCH bound,” *IEEE Trans. Inform. Theory*, vol. 40, pp. 204–207, 1994.
[30] T. Kasami and N. Tokura, “Some remarks on BCH bounds and minimum weights of primitive binary BCH codes,” *IEEE Trans. Inform. Theory*, vol. IT-15, pp. 408–413, 1969.
[31] E. Farr, unpublished.