Weak keys in the McEliece public-key cryptosystem

Pierre Loidreau and Nicolas Sendrier
Project CODES, INRIA Rocquencourt
Domaine de Voluceau, B.P. 105
78 153 Le Chesnay CEDEX, France
{Pierre.Loidreau, Nicolas.Sendrier}@inria.fr

September 12, 2000

Abstract

We show that it is possible to know whether the secret Goppa code of an instance of the McEliece public-key cryptosystem was chosen with a binary generator polynomial. Furthermore, whenever such a weak key is used, we present an attack which can be completed, for codes of length 1024 and dimension 524, with a large, but feasible amount of computation.

1 Introduction

In the paper we consider the security of the McEliece public-key cryptosystem [1]. In this system the public key is a generator matrix of a linear code. The encryption consists in choosing a codeword in this code to which an error vector of a given weight is added. The decryption is the decoding of these errors. The trap is the knowledge of a decoder for the public code. The security of the cryptosystem lies in the following two assumptions

- the parameters of the public code are large enough to avoid decoding by a general purpose decoder,
- it is difficult to build a fast (polynomial-time) decoder from the knowledge of the public code alone.

The issues regarding the first assumption were investigated at length in [2, 3, 4]. Here we deal with attacks related to the second assumption. In the original construction of the McEliece system the secret code Γ is picked in a family of binary Goppa codes of length $n = 2^m$ and error-correcting capability t where

$m = 10$ and $t = 50$. The public code is obtained by permuting the coordinates of Γ.

The support splitting algorithm [5] allows the computation of the permutation between two equivalent binary linear codes. Hence, this algorithm can be used to derive an attack by enumerating all the Goppa codes with suitable parameters. Because of the huge number of Goppa codes this attack remains unrealistic (for McEliece parameters it can be roughly estimated at 10^{130} years on a workstation). However, subfamilies of Goppa codes can be recognized – the weak keys – thanks to their particular structure. Namely, by applying the support splitting algorithm to Goppa codes with a binary generator polynomial one detects their non-trivial automorphism group. This allows an attack by enumerating the Goppa codes with a such property. Once again however, this attack remains unfeasible since it would require an unreasonable computation time (about 10^5 years on a workstation). Still, there is a way to greatly reduce its complexity by constructing the much shorter (length about n/m) projected idempotent subcode. We present a non-trivial lower bound for this subcode. From this bound we deduce the non-triviality of the code whenever the generator polynomial is binary. Finally we show how to modify the attack by using the properties of the projected idempotent subcode. With half size parameters ($m = 9$, $t = 28$) our implementation of the attack ran 15 minutes on a standard workstation. For full size parameters ($m = 10$, $t = 50$) it would require 500 years of computation [1].

2 Generalities

In this section, we briefly introduce the support splitting algorithm [5] and some of its properties which are relevant for our purposes. We also describe the McEliece public-key cryptosystem and give a straightforward attack derived from the support splitting algorithm. The cost of this attack – the best known today in the general setting – is taken as a reference to which the other attacks, presented later in the paper, should be compared.

2.1 Code equivalence – The Support Splitting Algorithm

A (n, k)-code is a linear code of length n and dimension k. The coordinates of a code of length n are labeled by an ordered set L of cardinality n. We denote by $\mathcal{S}_n$ the symmetric group of L.

Definition 1 (Equivalence of codes) *Let C be a binary (n, k)-code. For any permutation π in $\mathcal{S}_n$ and for any $a = (a_\alpha)_{\alpha \in L}$ in $\mathbf{F}_2^n$, we denote*

$$\pi(a) = (a_{\pi^{-1}(\alpha)})_{\alpha \in L} \text{ and } \pi(C) = \{\pi(a) \mid a \in C\}.$$

[1] All computations were carried out on a personal workstation equipped with a 500Mhz 21164 Digital Alpha processor

The codes C and $C' = \pi(C)$ are said to be equivalent *and we denote $C \sim C'$.*

Definition 2 (Automorphism group) *The* automorphism group *of a binary (n, k)-code C, denoted* $\mathrm{Aut}(C)$*, is the subgroup of the permutations π of $\mathcal{S}_n$ such that $\pi(C) = C$.*

Deciding the equivalence between two codes is a difficult problem [6]. Hence recovering the permutation between two equivalent codes is also difficult. In most cases however these two problems can be solved by using the support splitting algorithm [5] - denoted $\mathcal{SSA}$. In the following we consider $\mathcal{SSA}$ to be a black box taking as an argument a generator matrix of a linear code and returning a labeled partition $\mathcal{P} = \{(\mathcal{P}_s, s)\}_{s \in S}$ where S is the set of labels. The non-empty $\mathcal{P}_s$ are called the *cells* of $\mathcal{P}$ and form a partition of L. Two labeled partitions, $\mathcal{P}$ and $\mathcal{P}'$ are equivalent if for all $s \in S$, $|\mathcal{P}_s| = |\mathcal{P}'_s|$. We denote $\mathcal{P} \sim \mathcal{P}'$.

For any linear codes C and C' with generator matrices G and G', let $\mathcal{SSA}(G) = \{(\mathcal{P}_s, s)\}_{s \in S}$ and $\mathcal{SSA}(G') = \{(\mathcal{P}'_s, s)\}_{s \in S}$. The fundamental property of $\mathcal{SSA}$ is that

$$C' = \pi(C) \implies \forall s \in S,\ \mathcal{P}'_s = \pi(\mathcal{P}_s). \tag{1}$$

and implies in particular that the output of $\mathcal{SSA}$ is independent of the choice of the generator matrix. We denote $\mathcal{SSA}(C)$ the common output.

The converse of (1) is not necessarily true but is satisfied in practice when the number of cells is larger than a few units. In addition, since this property is true for all π in $\mathrm{Aut}(C)$, the orbits of the elements of the code support under the action of $\mathrm{Aut}(C)$ (we will simply talk of *orbits* or *C-orbits*) constitute the finest obtainable partition and we assume that the algorithm achieves this partition.

Assumption 1 *For any code C, the cells of $\mathcal{SSA}(C)$ are the C-orbits.*

This assumption remained experimentally valid in the computations we ran for this work. Yet, it may be false in some cases, for instance when one takes codes of short length (less than 50).

Algorithmic complexity of $\mathcal{SSA}$. The cost of the algorithm is $O(n^3 + 2^h n^2 l(n))$ binary operations, where n is the length of the input code C, h is the dimension of the hull $(C \cap C^\perp)$ and $l(n)$ is conjectured to be equal to $\log n$. The dimension of the hull is a small constant for random codes (see [7]), as is the case for all the codes considered in this paper. In practical terms the cost of the algorithm is $O(n^3)$.

2.2 The McEliece type cryptosystems

Let $\mathcal{G}$ be a family of binary (n, k)-codes for which a fast t-error correcting procedure is known. The McEliece type cryptosystems are described the following way:

- the *secret key* consists of an element Γ of $\mathcal{G}$, called the *hidden code*, and a permutation π of S_n,
- the *public key* is a random generator matrix G of $C = \pi(\Gamma)$ called the *public code*,
- the *encryption* of a message $m \in \mathbf{F}_2^k$ is $y = mG + e$, where e is a random vector of Hamming weight t,
- the *decryption* consists in recovering m from $mG + e$ and can easily be derived from π and the decoding procedure of Γ.

The security of the system relies on two assumptions. First, it is difficult to decode t errors in C simply from the knowledge of G. Second, it is difficult to deduce Γ and π from G.

The first assumption is connected with the NP-completeness of the decoding problem [8] and provides a good level of security for codes of length more than 1000. The second assumption is open to debate as it is greatly dependent on the structure of the family of hidden codes. It seems to be valid for Goppa codes but not for other classes of codes such as generalized Reed-Solomon codes [9, 10] or concatenated codes [11].

2.2.1 Enumerative attack on the McEliece system

McEliece proposed that Goppa codes be the family of hidden codes. So far, no efficient structural attack has been proposed. In this section we give a definition of these codes and a new (unrealistic) attack based on the $\mathcal{SSA}$ algorithm.

Definition 3 (Goppa codes) *Let $L = (\alpha_1, \ldots, \alpha_n)$ be an ordered subset of $\mathbf{F}_{2^m}$ and g be a polynomial of degree t over $\mathbf{F}_{2^m}$ with no roots on $\mathbf{F}_{2^m}$.*

The Goppa code $\Gamma(L, g)$ of generating vector *L and* generator polynomial *g is the set of binary words $a = (a_{\alpha_1}, \ldots, a_{\alpha_n}) \in \mathbf{F}_2^n$ such that*

$$\forall j,\ 0 \leq j < t,\ \sum_{\alpha \in L} a_\alpha \frac{\alpha^j}{g(\alpha)} = 0.$$

When g is square-free $\Gamma(L, g) = \Gamma(L, g^2)$ and the Berlekamp-Massey decoding algorithm for alternant codes can be used to correct up to t errors [12]. McEliece proposes choosing the hidden code in a family $\mathcal{G}$ of Goppa codes where the generating vector L is a fixed arbitrary labeling of $\mathbf{F}_{2^m}$ and the generator polynomial is an irreducible polynomial of degree t over $\mathbf{F}_{2^m}$. The suggested values for the parameters are $m = 10$ and $t = 50$. The support splitting algorithm allows a structural attack by enumerating the elements of $\mathcal{G}$ and testing their equivalence with the public code. The cardinality of the family is the number of irreducible monic polynomials of degree t over $\mathbf{F}_{2^m}$, that is $\approx 2^{mt}/t$ [13, p. 93].

This attack is best implemented using extended codes (by adding an overall parity-check bit). In general there are $m2^m(2^{2m}-1)$ monic polynomials g' such that the extension of $\Gamma(L,g')$ is equivalent to the extension of $\Gamma(L,g)$ [14, p. 347]. This number has to be divided by the cardinality of the automorphism group which is almost always reduced to the identity when $g(z) \in \mathbf{F}_{2^m}[z]$. Thus the number of codes to check is $\approx 2^{m(t-3)}/mt$. This remains far beyond computer range ($2^{461.0}$ with McEliece parameters) though an improvement on the other possible exhaustive attacks which enumerate the permutations [15]. However if one manages to identify a small subset of $\mathcal{G}$ which contains a Goppa code equivalent to the public code the cost can be decreased by searching this subset alone.

3 Weak keys in the McEliece cryptosystem

The weak keys are the Goppa codes with binary generator polynomials. They have two important properties. First their automorphism group is generated by the Frobenius field automorphism. This can be used to detect when a weak key has been used. However, we will show that the exhaustive search over the space of weak keys remains too costly.

The second property presented below deals with a particular construction: the projected idempotent subcode. We prove that this code is smaller and non trivial and we show how it can be efficiently used to reduce the cost of the exhaustive search.

3.1 Goppa codes with binary generator polynomials

We present the particular structure of Goppa codes $\Gamma(L,g)$ when g has coefficients in $\mathbf{F}_2$. We state that the automorphism group of the code is not trivial and that the dimension of the idempotent subcode is lower bounded.

3.1.1 Automorphism group

Proposition 4 *Let $L = \mathbf{F}_{2^m}$ and let $g(z)$ be an element of $\mathbf{F}_2[z]$. The automorphism group of the Goppa code $\Gamma(L,g)$ contains the Frobenius field automorphism $\sigma(z) = z^2$.*

The proof of the proposition is derived from a theorem by O. Moreno [14, p. 347]. In general the group generated by the Frobenius field automorphism is exactly the automorphism group.

3.1.2 Idempotent subcode

The conjugacy cosets are the orbits of the elements of $\mathbf{F}_{2^m}$ under the action of the Frobenius field automorphism. Since the Frobenius field automorphism of

$\mathbf{F}_{2^m}$ over $\mathbf{F}_2$ has order m these cosets have at most m elements. The conjugacy coset of α is $L_\alpha = \{\alpha, \alpha^2, \alpha^{2^2}, \ldots, \alpha^{2^{m-1}}\}$. The cardinality of L_α is equal to the extension degree over $\mathbf{F}_2$ of $\mathbf{F}_2[\alpha]$ – the smallest subfield containing α.

Example: Given $m = 4$, α a primitive element of $\mathbf{F}_{2^4}$ there are 6 conjugacy cosets

$$L = (\underbrace{0}_{L_0}, \underbrace{1}_{L_1}, \underbrace{\alpha, \alpha^2, \alpha^4, \alpha^8}_{L_\alpha}, \underbrace{\alpha^{12}, \alpha^3, \alpha^6, \alpha^9}_{L_{\alpha^3}}, \underbrace{\alpha^5, \alpha^{10}}_{L_{\alpha^5}}, \underbrace{\alpha^{11}, \alpha^{13}, \alpha^{14}, \alpha^7}_{L_{\alpha^7}})$$

The support of a word of $\mathbf{F}_2^n$ is the subset of L labeling its non-zero coordinates. In the case of a word that is stable under the action of the Frobenius field automorphism the support has a particular form.

Definition 5 *A word $a = (a_{\alpha_1}, \ldots, a_{\alpha_n})$ in $\Gamma(L, g)$ is called an* idempotent *if either one of the following equivalent assertions is satisfied*

(*i*) *$a = \sigma(a)$, where σ denotes the Frobenius field automorphism of $\mathbf{F}_{2^m}$,*

(*ii*) *the support of a is a union of conjugacy cosets of $\mathbf{F}_{2^m}$.*

The set of all idempotents of $\Gamma(L, g)$ is a linear subcode of $\Gamma(L, g)$. It is denoted by $I(L, g)$ and is called the idempotent subcode *of $\Gamma(L, g)$.*

Since by definition the coordinates labeled by conjugates are identical, we only need one label per conjugacy coset. Let $R \subset L$ be a set of representatives of the conjugacy cosets and let $N = |R|$ be the number of conjugacy cosets. We consider the projection

$$\begin{array}{rccc} \kappa_R : & \mathbf{F}_2^n & \to & \mathbf{F}_2^N \\ & (a_\alpha)_{\alpha \in L} & \mapsto & (a_\alpha)_{\alpha \in R} \end{array} \tag{2}$$

The vector $\tilde{a} = \kappa_R(a)$ is obtained from a by keeping the coordinates labeled by R. For all $\tilde{a}$ in $\mathbf{F}_2^N$ we denote by $\kappa_R^{-1}(\tilde{a})$ the (unique) idempotent such that $\kappa_R(\kappa_R^{-1}(\tilde{a})) = \tilde{a}$. By applying κ_R to all the elements of $I(L, g)$ we obtain a linear code

$$I(R, g) = \{\kappa_R(a) \mid a \in I(L, g)\}$$

with the same dimension as $I(L, g)$ while its length is shorter by a factor close to m.

Remark 1 Any other choice of representatives of the conjugacy cosets produces a code equivalent to $I(R, g)$.

Whenever the generator polynomial is binary, the idempotent subcode of a Goppa code is not trivial. The parity-check matrix of its restriction to a set of conjugacy cosets representatives can be expressed in terms of g and R. This provides an efficient construction as well as a lower bound on the dimension. For all α in $\mathbf{F}_{2^m}$ the smallest field containing α is $\mathbf{F}_2[\alpha]$. We denote by Tr_α the trace operator of $\mathbf{F}_2[\alpha]$ over $\mathbf{F}_2$ and by L_α the conjugacy coset of α

$$\begin{array}{rccc} \mathrm{Tr}_\alpha : & \mathbf{F}_2[\alpha] & \to & \mathbf{F}_2 \\ & \beta & \mapsto & \displaystyle\sum_{i=0}^{|L_\alpha|-1} \beta^{2^i} \end{array}$$

In order to simplify the notations we assume that the first N elements of the generating vector L form a set R of representatives of the conjugacy cosets of $\mathbf{F}_{2^m}$. We write $R = (\alpha_1, \ldots, \alpha_N)$.

Proposition 6 *Let $g(z)$ be a polynomial of degree t over $\mathbf{F}_2$. The binary matrix*

$$H_I = \begin{pmatrix} \mathrm{Tr}_{\alpha_1}\left(1/g(\alpha_1)\right) & \cdots & \mathrm{Tr}_{\alpha_N}\left(1/g(\alpha_N)\right) \\ \mathrm{Tr}_{\alpha_1}\left(\alpha_1/g(\alpha_1)\right) & \cdots & \mathrm{Tr}_{\alpha_N}\left(\alpha_N/g(\alpha_N)\right) \\ \vdots & \ddots & \vdots \\ \mathrm{Tr}_{\alpha_1}\left(\alpha_1^{t-1}/g(\alpha_1)\right) & \cdots & \mathrm{Tr}_{\alpha_N}\left(\alpha_N^{t-1}/g(\alpha_N)\right) \end{pmatrix}$$

is a parity-check matrix of $I(R, g)$ – the projected idempotent subcode of $\Gamma(L, g)$.

Proof: Let $\tilde{a} = (a_\alpha)_{\alpha \in R} \in \mathbf{F}_2^N$ and let $a = (a_\alpha)_{\alpha \in L} = \kappa_R^{-1}(\tilde{a})$. From the definition of Tr_α and since $g(z) \in \mathbf{F}_2[z]$ and a is an idempotent, we obtain

$$\forall j = 0, \ldots, t-1, \quad \sum_{\alpha \in L} a_\alpha \frac{\alpha^j}{g(\alpha)} = \sum_{\alpha \in R} a_\alpha \,\mathrm{Tr}_\alpha\left(\frac{\alpha^j}{g(\alpha)}\right).$$

And from Definition 3 we deduce

$$\tilde{a} \in I(R, g) \Leftrightarrow H_I \cdot {}^t\tilde{a} = 0.$$

Since $g(z) \in \mathbf{F}_2[z]$, $\alpha^j/g(\alpha)$ always lies in $\mathbf{F}_2[\alpha]$ and the coefficients of H_I are in $\mathbf{F}_2$. Therefore H_I is a parity-check matrix of $I(R, g)$. □

Corollary 7 *If $g(z)$ is a polynomial of degree t over $\mathbf{F}_2$, the dimension of $I(R, g)$ is at least $N - t$.*

3.2 Weak keys in the McEliece cryptosystem

When the hidden Goppa code in an instance of the McEliece cryptosystem has a binary generator polynomial, applying the support splitting algorithm to the

public key produces a partition whose cell cardinalities are exactly those of the conjugacy cosets. This enables us to detect when a such polynomial was used. Once detection has been performed, we search exhaustively over the set of binary polynomials of degree t to recover the secret key. Finally we show how to use the property of the idempotent subcode to greatly improve the efficiency of the attack.

3.2.1 Detection of weak keys

We consider an instance of the McEliece cryptosystem for which a binary Goppa generator polynomial is used. From (1), applying the support splitting algorithm to the public code and to the hidden Goppa code produces equivalent labeled partitions. Furthermore, because of Proposition 4, and under Assumption 1, the common cell cardinalities of these partitions will be those of the conjugacy cosets of $\mathbf{F}_{2^m}$ over $\mathbf{F}_2$. This allows us to derive a straightforward attack to detect the use of a weak key. Let G be the public key of an instance of the McEliece cryptosystem:

1. the attacker computes the labeled partition $\mathcal{P} = \mathcal{SSA}(G)$,
2. if the cardinalities of the cells of $\mathcal{P}$ are equal to the cardinalities of the conjugacy cosets then, with a high probability, the code spanned by G is equivalent to a Goppa code whose generator polynomial has binary coefficients.

Example: We take $n = 16$ and $t = 3$, the public key is the generator matrix of a binary $C(16, 4)$ code. Suppose the code is labeled by $\{1, \ldots, 16\}$ and that the cells obtained by applying $\mathcal{SSA}$ are

$$\{3\}, \{7\}, \{1, 6\}, \{2, 9, 12, 13\}, \{4, 5, 14, 16\}, \{8, 10, 11, 15\}.$$

The cells of this partition have the same size as the conjugacy cosets, that is two cells are singletons, one is a 2-tuple and the last three have cardinality 4. Although this is not a sufficient condition, it nevertheless provides a very good detector for binary generator polynomials.

Note that if Assumption 1 is invalid, we obtain a coarser partition which may still help to make a decision.

3.2.2 Enumerative attack on weak keys

Once a weak key has been detected the attacker must solve the following problem

General problem : *Find a polynomial g of degree t with binary coefficients such that $\Gamma(L, g)$ is equivalent to the public code C.*

The direct approach consists in enumerating all the irreducible binary polynomials g of a given degree t and testing the equivalence between $\Gamma(L, g)$ and C.

Algorithm 1
input: a binary $k \times n$ matrix G, an integer t
output: an irreducible binary polynomial of degree t such that $C \sim \Gamma(L, g)$
compute $\mathcal{P} = \mathcal{SSA}(G)$
for all $g(z) \in \mathbf{F}_2[z]$ irreducible of degree t do
compute a generator matrix G_Γ of $\Gamma(L, g)$
if $\mathcal{SSA}(G_\Gamma) \sim \mathcal{P}$ then return g

Running the algorithm costs at least $\lambda n^3 I_t$ binary operations where λ is a small constant implementation defined and where I_t is the number of irreducible binary polynomials of degree t. The parameter λ can be much smaller than 1 if the binary nature of the problem is taken into account.

With McEliece parameters $n = 2^{10}$ and $I_t \approx 2^{44.3}$, the cost of one detection (the λn^3 part) can be made as low as 0.15 seconds. The entire computation time is greater than 10^5 years, still remaining out of range. With smaller parameters such as $n = 512$ and $t = 28$ the expected computation time is a few days.

3.2.3 Using the idempotent subcode

If we performed the search over the projected idempotent subcodes instead of Goppa codes, the cost would be reduced by a factor close to m^3. To achieve this, we derive from the public code C – equivalent to some Goppa code $\Gamma(L, g)$– a shorter code that is equivalent to $I(R, g)$.

Let C be a binary code of length n, let $\mathcal{P} = \mathcal{SSA}(C)$ and let $E_\mathcal{P}$ be the set of binary words whose support is a union of cells of $\mathcal{P}$. Let us define the code

$$\mathcal{I}(C) = E_\mathcal{P} \cap C. \tag{3}$$

Since $\mathcal{I}(C)$ has many repeated coordinates – as for the idempotent subcode of a Goppa code – we pick one element in each cell to produce a shorter code of the same dimension. Let T denote a subset of the code support containing exactly one element per cell of $\mathcal{P}$. Let κ_T (as defined by (2)) be the projection on the positions labeled by T. We define the code

$$\tilde{\mathcal{I}}(C) = \{\kappa_T(x) \mid x \in \mathcal{I}(C)\}.$$

Proposition 8 *If the cell cardinalities of $\mathcal{SSA}(C)$ match the cardinalities of the conjugacy cosets and if $C \sim \Gamma(L, g)$ for some binary polynomial g then $\tilde{\mathcal{I}}(C) \sim I(R, g)$.*

Proof: For any permutation π, let $\mathcal{P} = \mathcal{SSA}(C)$ and $\mathcal{P}' = \mathcal{SSA}(\pi(C))$. Because of (1), we have $E_{\mathcal{P}'} = \pi(E_\mathcal{P})$ and thus $\mathcal{I}(C) \sim \mathcal{I}(C')$. By taking the projections, we deduce

$$C \sim C' \Rightarrow \tilde{\mathcal{I}}(C) \sim \tilde{\mathcal{I}}(C'). \tag{4}$$

From Proposition 4, equation (1) and the assumption on $\mathcal{SSA}(C)$, the cells of $\mathcal{SSA}(\Gamma(L,g))$ are the conjugacy cosets. Thus $\mathcal{I}(\Gamma(L,g)) = \Gamma(L,g)$. By removing the repeated coordinates, we get

$$\tilde{\mathcal{I}}(\Gamma(L,g)) = I(R,g). \tag{5}$$

From (4) and (5) we easily obtain the result. □

A generator matrix of the projected code $\tilde{\mathcal{I}}(C)$ can be computed from any generator matrix G of C. Once $\mathcal{P} = \mathcal{SSA}(G)$ has been computed, we can compute a generator matrix of $\mathcal{I}(C)$ from equation (3). By eliminating duplicate columns we obtain a generator matrix of $\tilde{\mathcal{I}}(C)$ and the general problem is reduced to the following:

Reduced problem : *Find a polynomial g of degree t with binary coefficients such that $I(R,g) \sim \tilde{\mathcal{I}}(C)$.*

Algorithm 2
input: a binary $k \times n$ matrix G, an integer t
output: an irreducible binary polynomial of degree t such that $C \sim \Gamma(L,g)$

```
compute P = SSA(G)
compute a parity-check matrix H of Ĩ(C)
compute P' = SSA(H)
for all g(z) ∈ F_2[z] irreducible of degree t do
    compute a parity-check matrix H_I of I(R,g)
    if SSA(H_I) ~ P' then return g
```

There is no difference whether a parity-check matrix is used instead of a generator matrix to test code equivalence. It simply means that we consider the duals of the codes. This is used in Algorithm 2 because Proposition 6 gives an explicit formula for H_I.

3.2.4 Recovering the secret permutation

In the last step of the attack the problem becomes: given the public code C and given a polynomial $g(z) \in \mathbf{F}_2[z]$ such that $\Gamma(L,g) \sim C$, how can we find π satisfying $C = \pi(\Gamma(L,g))$. At this stage, we have

$$\mathcal{SSA}(G_\Gamma) = \{(L_\alpha, s_\alpha)\}_{\alpha \in R} \text{ and } \mathcal{SSA}(G) = \{(\mathcal{P}_\alpha, s_\alpha)\}_{\alpha \in R},$$

where $\mathcal{P}_\alpha = \pi(L_\alpha)$ for all $\alpha \in R$.

If we pick $\alpha \in L$ such that $\mathrm{Aut}(\Gamma(L \setminus \{\alpha\}, g)) = \{1\}$ – most α satisfy this condition – and $\beta \in \pi(L_\alpha)$ then $\Gamma(L \setminus \{\alpha\}, g)$ – Γ punctured in α – and $C_{\{\beta\}}$ – C punctured in β – are equivalent. Applying $\mathcal{SSA}$ to both codes produces two equivalent labeled partitions containing only cells of cardinality one (under Assumption 1) and the matches between the cells provide the permutation π.

3.3 Implementation

For a Goppa code of length 1024 with a binary generator polynomial of degree 50, the average running time of the support splitting algorithm on our workstation is half a second. This produces the orbits under action of $\mathrm{Aut}(C)$.

The implementation of Algorithm 2 requires at least $\lambda I_t N^3$ binary operations. However, enumerating irreducible polynomials raises the problem of irreducibility testing. In our implementation we enumerate the polynomials of degree t with a constant term equal to 1 and with an odd number of monomials. There are 2^{t-2} such polynomials instead of the I_t irreducible polynomials. In addition, a polynomial is tested with $\mathcal{SSA}$ only if it is smaller – for the lexicographic order – than those obtained by the 5 transformations $z \mapsto z+1$, $z \mapsto 1/z$, $z \mapsto 1/z+1$, $z \mapsto 1/(z+1)$ and $z \mapsto z/(z+1)$ (this can be implemented in time $O(t)$ and a storage size of $O(t^2)$). It can be shown [14, p. 347] that these transformations leave the family of extended Goppa codes with binary polynomials globally invariant. This divides the size of the search space by 6 with a negligible additional cost. Using then an adapted time/memory trade-off, the cost for the construction of matrix H_I (Proposition 6) is negligible compared with the cost of $\mathcal{SSA}$. The overall average cost becomes $\lambda 2^t N^3/48$ ($N = n/m$), where λ is small and only depends on $\mathcal{SSA}$.

The program ran for the parameters $m = 9$ and $t = 28$. In this case the projected extended idempotent subcode has length 61 and dimension 32 and the result was obtained on average after 15 minutes. The computation time for a full size instance ($m = 10$, $t = 50$) with this implementation would be 500 years. This is high but not unrealistic since the algorithm allows massive distribution.

4 Conclusion

The method used in the paper to detect Goppa codes with binary generator polynomials can be naturally generalized to detect a Goppa code with a generator polynomial over any other subfield of $\mathbf{F}_{2^{10}}$, that is $\mathbf{F}_4$ or $\mathbf{F}_{32}$ in the case of McEliece parameters. Still, this only classifies too small a ratio of the secret keys and in both cases the number of generators is much too high to allow any kind of exhaustive search.

We also tried to point out how a more general attack on the McEliece type public-key cryptosystems should work. Performing a general structural attack on such cryptosystems could be achieved by partitioning the set of Goppa codes into many small identifiable and enumerable subsets. Note that with a good choice of the hidden code family one can protect the system against our attack. For instance a system proposed by Sidel'nikov [16] uses as a *single* hidden code a low rate Reed-Muller code. This code is weakly self-dual and thus equal to its hull. Since the support splitting algorithm computes the weight enumerator of

the hull, the running time becomes prohibitive in this case.

References

[1] R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory," *DSN Prog. Rep.,* Jet Prop. Lab., California Inst. Technol., Pasadena, CA, pp. 114–116, Jan. 1978.

[2] P. J. Lee and E. F. Brickell, "An observation on the security of McEliece's public-key cryptosystem," in *Advances in Cryptology – EUROCRYPT'88* (C. G. Günther, ed.), no. 330 in LNCS, pp. 275–280, Springer-Verlag, 1988.

[3] J. van Tilburg, "On the McEliece cryptosystem," in *Advances in Cryptology - CRYPTO'88* (S. Goldwasser, ed.), no. 403 in LNCS, pp. 119–131, Springer-Verlag, 1990.

[4] A. Canteaut and F. Chabaud, "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," *IEEE Transactions on Information Theory*, vol. 44, pp. 367–378, Jan. 1998.

[5] N. Sendrier, "Finding the permutation between equivalent codes: the support splitting algorithm," *IEEE Transactions on Information Theory*, vol. 46, pp. 1193–1203, July 2000.

[6] E. Petrank and R. M. Roth, "Is code equivalence easy to decide?," *IEEE Transactions on Information Theory*, vol. 43, pp. 1602–1604, Sept. 1997.

[7] N. Sendrier, "On the dimension of the hull," *SIAM Journal on Discrete Mathematics*, vol. 10, pp. 282–293, May 1997.

[8] E. R. Berlekamp, R. J. McEliece, and H. C. van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Transactions on Information Theory*, vol. 24, May 1978.

[9] R. M. Roth and G. Seroussi, "On generator matrices of MDS codes," *IEEE Transactions on Information Theory*, vol. 31, pp. 826–830, May 1985.

[10] V. M. Sidel'nikov and S. O. Shestakov, "On cryptosystem based on generalized Reed-Solomon codes," *Discrete mathematics (in russian)*, vol. 4, no. 3, pp. 57–63, 1992.

[11] N. Sendrier, "On the concatenated structure of a linear code," *AAECC*, vol. 9, no. 3, pp. 221–242, 1998.

[12] N. J. Patterson, "The algebraic decoding of Goppa codes," *IEEE Transactions on Information Theory*, vol. 21, pp. 203–207, Mar. 1975.

[13] R. Lidl and H. Niederreiter, *Finite Fields*. Cambridge University Press, 1983.

[14] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[15] J. K. Gibson, "Equivalent Goppa codes and trapdoors to McEliece's public key cryptosystem," in *Advances in Cryptology - EUROCRYPT'91* (D. W. Davies, ed.), no. 547 in LNCS, pp. 517–521, Springer-Verlag, 1991.

[16] V. M. Sidel'nikov, "A public-key cryptosystem based on Reed-Muller codes," *Discrete Mathematics and Applications*, vol. 4, no. 3, pp. 191–207, 1994.