How to achieve a McEliece-based Digital Signature Scheme

Nicolas Courtois[1,2], Matthieu Finiasz[1,3], and Nicolas Sendrier[1]

[1] Projet Codes, INRIA Rocquencourt
BP 105, 78153 Le Chesnay - Cedex, France
Nicolas.Sendrier@inria.fr

[2] Systèmes Information Signal (SIS), Toulon University
BP 132, F-83957 La Garde Cedex, France
courtois@minrank.org
http://www.minrank.org/

[3] École Normale Supérieure, 45, rue d'Ulm, 75005 Paris.
finiasz@ens.fr

Abstract. McEliece is one of the oldest known public key cryptosystems. Though it was less widely studied than RSA, it is remarkable that all known attacks are still exponential. It is widely believed that code-based cryptosystems like McEliece do not allow practical digital signatures. In the present paper we disprove this belief and show a way to build a practical signature scheme based on coding theory. It's security can be reduced in the random oracle model to the well-known *syndrome decoding problem* and the distinguishability of permuted binary Goppa codes from a random code. For example we propose a scheme with signatures of 81-bits and a binary security workfactor of 2^{83}.

Key Words: digital signature, McEliece cryptosystem, Niederreiter cryptosystem, Goppa codes, syndrome decoding, short signatures.

1 Introduction

The RSA and the McEliece [11] public key cryptosystems, have been proposed back in the 70s. They are based on intractability of respectively *factorization* and *syndrome decoding problem* and both have successfully resisted more than 20 years of cryptanalysis effort.

RSA became the most widely used public key cryptosystem and McEliece was not quite as successful. Partly because it has a large public key, which is less a problem today, with huge memory capacities available at very low prices. However the main handicap was the belief that McEliece could not be used in signature. In the present paper we show that it is indeed possible to construct a signature scheme based on Niederreiter's variant [12] on the McEliece cryptosystem.

The cracking problem of RSA is the problem of extracting e-th roots modulo N called the RSA problem. All the general purpose attacks for it are structural attacks that factor the modulus N. It is a hard problem but sub-exponential. The cracking problem for McEliece is the problem of decoding an error correcting code called Syndrome Decoding (SD). There is no efficient structural attacks that might distinguish between a permuted Goppa code used by McEliece and a random code. The problem SD is known to be NP-hard since the seminal paper of Berlekamp, McEliece and van Tilborg [3], in which authors show that complete decoding of a random code is NP-hard.

All among several known attacks for SD are fully exponential (though faster than the exhaustive search [4]), and nobody has ever proposed an algorithm that behaves differently for *complete decoding* and the *bounded decoding* problems within a (slightly smaller) distance accessible to the owner of the trapdoor. In [6] Kobara and Imai review the overall security of McEliece and claim that

> *[...] without any decryption oracles and any partial knowledge on the corresponding plaintext of the challenge ciphertext, no polynomial-time algorithm is known for inverting the McEliece PKC whose parameters are carefully chosen.*

Thus it would be very interesting to dispose of signature schemes based on such hard decoding problems. The only solution available up to date was to use zero-knowledge schemes based on codes such as the SD scheme by Stern [19]. It gives excellent security but the signatures are very long. All tentatives to build practical schemes failed, see for example [20].

Any trapdoor function allows digital signatures by using the unique capacity of the owner of the public key to invert the function. However it can only be used to sign messages the hash value of which lies in the ciphertext space. Therefore a signature scheme based on trapdoor codes must achieve complete decoding. In the present paper we show how to achieve complete decoding of Goppa codes for some parameter choices.

The paper is organized as follows. First we explain in §2 and §3 how and for which parameters to achieve complete decoding of Goppa codes. In §4 we present a practical and secure signature scheme we derive from this technique. Implementation issues are discussed in §5, and in particular, we present several tradeoffs to achieve either extremely short signatures (81 bits) or extremely fast verification. In §6 we present an asymptotic analysis of all the parameters of the system, proving that it will remain practical and secure with the evolution of computers. Finally in §7 we prove that the security of the system relies on the syndrome decoding problem and the distinguishability of Goppa codes from random codes.

2 Signature with McEliece

The McEliece cryptographic scheme is based on error correcting codes. It consists in randomly adding errors to a codeword (as it would happen in a noisy channel) and uses this as a cipher. The decryption is done exactly as it would be done to correct natural transmission errors. The security of this scheme simply relies on the difficulty of decoding a word without any knowledge of the structure of the code. Only the legal user can decode easily using the trap. The Niederreiter variant - equivalent on a security point of view [8] - uses a syndrome (see below) as ciphertext, and the message is an error pattern instead of a codeword (see Table 1).

2.1 A brief description of McEliece's and Niederreiter's schemes

Let $\mathbf{F}_2$ be the field with two elements $\{0,1\}$. In the present paper, C will systematically denote a binary linear code of length n and dimension k, that is a subspace of dimension k of the vector space $\mathbf{F}_2^n$. Elements of $\mathbf{F}_2^n$ are called words, and elements of C are codewords. A code is usually given in the form of a generating matrix G, lines of which form a basis of the code. The parity check matrix H is a dual form of this generating matrix: it is the $n \times (n-k)$ matrix of the application of kernel C. When you multiply a word (a codeword with an error for example) by the parity check matrix you obtain what is called a syndrome: it has a length of $n-k$ bits and is characteristic of the error added to the codeword. It is the sum of the columns of H corresponding to the non-zero coordinates of the error pattern. Having a zero syndrome characterizes the codeword and we have $G \times H = 0$.

Let C be a binary linear code of length n and dimension k correcting t errors (i.e. minimum distance is at least $2t+1$). Let G and H denote respectively a generator and a parity check matrix of C. Table 1 briefly describes the two main encryption schemes based on code. In both case the *trap* is a t-error correcting procedure for C. It enables decryption (*i.e.* finding the closest codeword to a given word or equivalently the word of smallest Hamming weight with a prescribed syndrome).

	McEliece	**Niederreiter**
public key:	G	H
cleartext:	$x \in \mathbf{F}_2^k$	$x \in \mathbf{F}_2^n,\ w_H(x) = t$
ciphertext:	$y = xG + e,\ w_H(e) = t$	$y = Hx^T$
ciphertext space:	$\mathbf{F}_2^n$	$\mathbf{F}_2^{n-k}$

Table 1. McEliece and Niederreiter code-based cryptosystems

The *secret key* is a code C_0 (usually a Goppa code) whose algebraic structure provides a fast decoder. The public code is obtained by randomly permuting the coordinates of C_0 and then choosing a random generator or parity check matrix:

$$G = UG_0P \quad \text{or} \quad H = VH_0P$$

where G_0 and H_0 are a generator and a parity check matrix of C_0, U and V are non-singular matrices ($k \times k$ and $(n-k) \times (n-k)$ respectively) and P is a $n \times n$ permutation matrix.

The security of these two systems is proven to be equivalent [8] and is based on two assumptions:

– solving an instance of the decoding problem is difficult,
– recovering the underlying structure of the code is difficult.

The first assumption is enforced by complexity theory results [3, 2, 16], and by extensive research on general purpose decoders [7, 18, 4]. The second assumption received less attention. Still the Goppa codes used in McEliece are known by coding theorists for thirty years and so far no polynomially computable property is known to distinguish a permuted Goppa code from a random linear code.

2.2 How to make a signature

In order to obtain an efficient digital signature we need two things: an algorithm able to compute a signature for any document such that they identify their author uniquely, and a fast verification algorithm available to everyone.

A public-key encryption function can be used as a signature scheme as follows:

1. hash (with a public hash algorithm) the document to be signed,
2. decrypt this hash value as if it were an instance of ciphertext,
3. append the decrypted message to the document as a signature.

Verification just applies the public encryption function to the signature and verifies that the result is indeed the hash value of the document. In the case of Niederreiter or any other cryptosystem based on error correcting codes the point 2 fails. The reason is that if one considers a random syndrome it usually corresponds to an error pattern of weight greater than t. In other word, it is difficult to generate a random ciphertext unless it is explicitly produced as an output of the encryption algorithm.

One solution to the problem is to obtain for our code an algorithm to decode any syndrome, or at least a good proportion of them. It is the object of the next section.

2.3 Complete decoding

Complete decoding consists of finding a nearest codeword to any given word of the space. In a syndrome language that is being able to find an error pattern corresponding to any given syndrome. This means decoding syndromes corresponding to errors of weight greater than t.

An approach to try to perform complete decoding would be to try to correct a fixed additional number of errors (say δ). To decode a syndrome corresponding to an error of weight $t + \delta$ one should then add δ random columns from the parity check matrix to the syndrome and try to decode it. If all of the δ columns correspond to some error positions then the new syndrome obtained will correspond to a word of weight t and can be decoded by our trapdoor function. Else we will just have to try again with δ other columns, and so on until we can

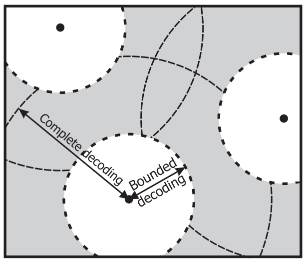

Fig. 1. from bounded decoding to complete decoding

decode one syndrome. Like this we can decode any syndrome corresponding to an error of weight less than or equal to $t + \delta$. If δ is large enough we should be able to decode any possible syndrome. However, a large δ will lead to a small probability of success for each choice of δ columns. This means that we will have to adapt the parameters of our code to obtain a δ small enough and in the same time keep a good security for our system.

This can be viewed from an different angle. Adding a random column of the parity check matrix to a syndrome really looks like choosing another random syndrome and trying to decode it. Choosing parameters for the code such that δ is small enough simply consists of increasing the density of the decodable syndromes in the space of all the syndromes, this is increasing the probability for a random syndrome to be decodable. This method will therefore take a first random syndrome (given by the hash function) and try to decode it, then modify the document and hash it again until a decodable syndrome is obtained.

The object of the next section will be to choose parameters such that the number of necessary attempts is small enough for this method to work in a reasonable time.

3 Finding the proper parameters

The parameters of the problem are the dimension k of the code, its length n and the maximum number t of errors the code can correct. These parameters affect all aspects of the signature scheme: its security, the algorithmic complexity for computing a signature, the length of the signature... We will start by exploring the reasons why the classical McEliece parameters are not acceptable and continue with what we wish to obtain.

3.1 Need for new parameters

With the classical McEliece parameters ($n = 1024$, $k = 524$, $t = 50$) we have syndromes of length $n - k = 500$. This makes a total of 2^{500} syndromes. Among these only those corresponding to words of weight less than 50 are decodable. The number of such syndromes is: $\sum_{i=1}^{50} \binom{1024}{i} \simeq 2^{284}$

Therefore there is only a probability of 2^{-216} of success for each syndrome. This would mean an average number of decoding attempts of 2^{216} which is far too much. We will hence have to change the values of n, k and t.

3.2 Choosing Parameters

Binary Goppa codes are subfield subcodes of particular alternant codes [10, Ch. 12]. For a given integer m, there are many (about $2^{tm}/t$) t-error correcting Goppa codes of dimension $n - tm$ and length $n = 2^m$.

We are looking for parameters which lead to a good probability of success for each random syndrome. The probability of success will be the ratio between the number of decodable syndromes $\mathcal{N}_{dec}$ and the total number of syndromes $\mathcal{N}_{tot}$. As n is large compared with t we have:

$$\mathcal{N}_{dec} = \sum_{i=1}^{t} \binom{n}{i} \simeq \binom{n}{t} \simeq \frac{n^t}{t!}$$

and for Goppa codes $\mathcal{N}_{tot} = 2^{n-k} = 2^{mt} = n^t$. Therefore the probability of success is:

$$\mathcal{P} = \frac{\mathcal{N}_{dec}}{\mathcal{N}_{tot}} \simeq \frac{1}{t!}$$

This probability doesn't depend of n and the decoding algorithm has a polynomial complexity in m ($= \log_2 n$) and t. Therefore the signature time won't change a lot with n. As the security of the Goppa code used increases rapidly with n we will then be sure to find suitable parameters, both for the signature time and the security.

3.3 Secure parameters

A fast bounded decoding algorithm can perform about one million decoding in a few minutes[1]. From the previous section, the number of decoding attempt to get one signature will be around $t!$, so get a reasonable signature scheme, t should not be more than 10. However for the codes correcting such a little number of errors we need to have very long codewords in order to achieve good security.

The Table 2 shows the binary workfactors for the Canteaut-Chabaud attack [4] on the McEliece cryptosystem (see section 6 for more details on the complexity of these attacks). We assume that an acceptable security level is of 2^{80} CPU operations, corresponding roughly to a binary workfactor of 2^{86}. Therefore, in our signature scheme, we need a length of at least 2^{15} with 10 errors or 2^{16} with 9 errors.

Though it is slightly below or security requirement, the choice $(2^{16}, 9)$ is better as it runs about 10 times faster.

[1] our implementation performs one million decodings in 5 minutes, but it can still be improved

	n						
	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}
$t=8$	$2^{52.0}$	$2^{56.8}$	$2^{61.4}$	$2^{65.3}$	$2^{67.8}$	$2^{70.5}$	$2^{73.3}$
$t=9$	$2^{54.6}$	$2^{59.9}$	$2^{69.3}$	$2^{74.0}$	$2^{78.8}$	$2^{83.7}$	$2^{88.2}$
$t=10$	$2^{60.9}$	$2^{66.8}$	$2^{72.3}$	$2^{77.4}$	$2^{87.4}$	$2^{90.9}$	$2^{94.6}$

Table 2. Cost for decoding

4 The signature scheme

With the chosen parameters we have a probability of $1/9!$ to decode each syndrome. We will therefore have to try to decode about $9!$ random syndromes. To do so we will simply use a counter i and hash it with the document: the hashed syndrome obtained will then depend of i, and by changing i we can have as many as we need. The signature scheme works as follows.

Let h be a hash function returning a binary word of length $n-k$ (the length of a syndrome). Let D be our document and $s = h(D)$. We denote $[\cdots s \cdots \mid \cdot i \cdot]$ the concatenation of s and i and $s_i = h([\cdots s \cdots \mid \cdot i \cdot])$.

The signature algorithm will compute the s_i for i starting at 0 and increasing by 1 at each try, until one of the syndromes s_i is decodable. We will note i_0 the first index for which s_i is decodable, and we will use this syndrome for the signature. As explained in section 2.2 the signature will then be the decrypted message, which is in our case the word z of length n and weight 9, such that $Hz^T = s_{i_0}$. However the signature will also have to include the value of i_0 for the verification. The signature will therefore be $[\cdots z \cdots \mid \cdot i_0 \cdot]$.

Signature length: the length of the signature will mainly depend of the way used to store z. It is a word of length $n = 2^{16}$ so the dumb method would be to use it directly to sign. However its weight is only 9 so we should be able to compress it a little. There are $\binom{2^{16}}{9} \simeq 2^{125.5}$ word of weight 9 so they could be indexed with a 126 bit counter. Let $i_1 < \ldots < i_9$ denote the positions of the non-zero bits of z. We define the index I_z of z by:

$$I_z = 1 + \binom{i_1}{1} + \binom{i_2}{2} + \ldots + \binom{i_9}{9}$$

The number of bits used to store i_0 isn't reducible: in average it's length is $\log_2(9!) \simeq 18.4$ bits. So the signature will be $[\cdots I_z \cdots \mid \cdot i_0 \cdot]$ with an average total length of $125.5 + 18.4 \simeq 144$ bits.

Note that using McEliece encryption scheme instead of Niederreiter's would not be satisfactory here. The signature would have a size larger than k bits (the size of a plaintext). And it would grow radiply with m if t is small. With the parameters above, the signature would have a length of 65411 bits!

Signature algorithm

- hash the document D into $s = h(D)$
- compute $s_i = h([\cdots s \cdots \mid \cdot i \cdot])$ for $i = 0, 1, 2 \ldots$
- find i_0 the smallest value of i such that s_i is decodable
- use our trapdoor function to compute z such that $Hz^T = s_{i_0}$
- compute the index I_z of z in the space of words of weight 9
- use $[\cdots I_z \cdots \mid \cdot i_0 \cdot]$ as a signature for D

Verification algorithm is much simpler (and faster)

- recover z from its index I_z
- compute $s_1 = Hz^T$ with the public key H
- compute $s_2 = h([\cdots h(D) \cdots \mid \cdot i_0 \cdot])$ with the public hash function
- compare s_1 and s_2: if they are equal the signature is valid

4.1 Attacks on the signature length

Having such short signatures enables attacks independent on the strength of the trapdoor function used, which are inherent to the commonly used method of computing a signature by inversion of the function. This generic attack runs in the square root of the exhaustive search. Let F be any trapdoor function with an output space of cardinality 2^r. The well known birthday paradox forgery attack computes $2^{r/2}$ hash[2] values $\mathrm{MD}(m_i)$ for some chosen messages, and picks at random $2^{r/2}$ possible signatures. One of these signatures is expected to correspond to one of the messages.

With our parameters the syndromes have a length of 144 bits and the complexity of the attack is the complexity of sorting the $2^{144/2} = 2^{72}$ values which is $2^{72} \times 72 \times 144 \simeq 2^{85}$ binary operations. This attack is not more threatening than the decoding attack, and in addition it requires a memory of about $2^{72} \times 72$ bits. Note also that the above attack depends on the syndrome length and not on the signature length, this will remain true later, even in the variants with shorter signature length.

5 Implementation aspects

For any signature scheme there is an easy security preserving tradeoff between signature length and verification time. One may remove any h bits from the signature if one accepts exhaustive verification in 2^h for each possible value of the h missing bits. In the case of syndrome-based signature, one can do much better. As the signature consists of an error pattern of weight t, one may send

[2] MD denotes a cryptographic hash function with output of r bits

only $t-1$ out of the t errors. The verifier needs to decode the remaining error and this is much faster that the exhaustive search. More generally we are going to show that concealing a few errors (between 1 and 3) remains an excellent compromise as summarized in Table 3.

5.1 Cost of a verification

Let s denote the hash value of the message and z denote the error pattern of weight t such that $Hz^T = s$. As z is the signature, we can compute $y = Hz^T$ by adding the t corresponding columns. The signature is accepted if y is the equal to s. The total cost of this verification is t column operations[3].

If u is a word of weight $t-1$ whose support is included in the support of z, we compute $y = s + Hu^T$, which costs $t-1$ column operations, and we check that y is a column of H, which does not cost more than one column operation if the matrix H is properly stored in a hash table.

Omitting two errors. Let us assume now that the word u transmitted as signature has weight $t-2$. There exists a word x of weight 2 such that $Hx^T = y = Hu^T$. We are looking for two columns of H whose sum is equal to y. All we have to do is to add y to any column of H and look for a match in H. Again if the columns of H are properly stored, the cost is at most $2n$ column operations.

This can be improved as the signer can choose which 2 errors are left to verifier to correct and omits in priority the positions which will be tested first, this divides the complexity in average by t (*i.e.* the match will be found in average after n/t tries).

Omitting more errors. In general, if u has weight $t-w$, we put $y = s + Hu^T$ and we need to compute the sum of y plus any $w-1$ columns of H and check for a match among the columns of H. Proper implementation will cost at most $3\binom{n}{w-1}$ column operations (yes, it is always 3, don't ask why!).

Again, if the signer omits the set of w errors which are tested first, the average cost can be divided by $\binom{t}{w-1}$.

Note that if more than 2 errors are not transmitted, the advantage is not better than the straightforward time/length tradeoff.

5.2 Partitioning the support

Punctured code. Puncturing a code in p positions consist in removing the corresponding coordinates from the codewords. The resulting code has length $n-p$ and, in general, the same dimension[4] k. Without loss of generality we can

[3] In this section we will count all complexities in terms of column operations, one column operation is typically one access to a table and one operation like an addition or a comparison

[4] the actual dimension is the rank of a matrix derived from a generating matrix by removing the p columns

remaining errors	cost [(a)] of verification		signature length
0	t	9	144 bits
1	t	9	132 bits
2	$2n/t$	2^{14}	119 bits
3	$3\binom{n}{2}/\binom{t}{2}$	2^{27}	105 bits
4	$3\binom{n}{3}/\binom{t}{3}$	2^{40}	92 bits

[(a)] in column operations (≈ 4 to 8 CPU clocks).

Table 3. Tradeoffs for the 9-error correcting Goppa code of length 2^{16}

assume that the punctured positions come first. A parity check matrix H' of C' can be derived from any parity check matrix H of C by a Gaussian elimination: for some non-singular $(n-k)\times(n-k)$ matrix U we have

$$UH = \left(\begin{array}{c|c} I & R \\ \hline 0 & H' \end{array}\right),$$

where I denotes the $p \times p$ identity matrix.

Given a syndrome s we wish to find $z \in \mathbf{F}_2^n$ of weight t such that $s = Hz^T$. We denote s'' and s' respectively the p first and the $n-p-k$ last bits of Us and z' the last $n-p$ bits of z. Let $w \leq t$ denote the weight of z'.

$$\begin{cases} s = Hz^T \\ w_H(z) \leq t \end{cases} \Leftrightarrow \begin{cases} s' = H'z'^T \\ w_H(z') + w_H(Rz'^T + s'') \leq t \end{cases}$$

Shorter signatures. We keep the notations of the previous section. We partition the support of C into n/l sets of size l. Instead of giving the $t-w$ positions, we give the $t-w$ sets containing these positions. These $p = l(t-w)$ positions are punctured to produce the code C'. To verify the signature s we now have to correct w errors in C', *i.e.* find z' of weight w such that $s' = Hz'^T$. The signature is valid if there exists a word z' such that

$$w_H(z') \leq w \tag{1}$$

$$w_H(Rz'^T + s'') \leq t - w \tag{2}$$

We may find several values of z' verifying (1), but only one of them will also verify (2). If l is large, we have to check equation (2) often. On the other hand, large values of l produce shorter signatures. The best compromise is $l = m$ or a few units more.

The cost for computing H' is around $tm2^{m-1}$ column operations (independently of l and w). The number of column operations for decoding errors in C' is the same as in C but columns are smaller.

The signature size will be $\log_2\left(\binom{n/l}{t-w}t!\right)$. If more than 3 errors are not transmitted, the length gain is not advantageous.

remaining errors (w)	cost $^{(a)}$ of verification	signature length
1	2^{22}	100 bits
2	2^{22}	91 bits
3	2^{27}	81 bits
4	2^{40}	72 bits

$^{(a)}$ in column operations (≈ 2 to 6 CPU clocks).

Table 4. Tradeoffs for $m = 16$, $t = 9$ and $l = m$

5.3 New short signature schemes

With parameters $m = 16$ and $t = 9$, there are three interesting trade-offs between verification time and signature length. All three of them have the same complexity for computing the signature (in our our implementation the order of magnitude is one minute) and the same security level of about 2^{80} CPU operations.

Fast verification (CFS1). We transmit 8 out of the 9 error positions, the verification is extremely fast and the average signature length is $\log_2\left(t!\binom{n}{t-1}\right) = 131.1 < 132$ bits.

Short signature (CFS3). We partition the support in 2^{12} cells of 16 bits and we transmit 6 of the 9 cells. The verification time is relatively long, around one second and the average signature length is $\log_2\left(\binom{n/l}{t-w}t!\right) = 80.9 < 81$ bits.

Half & half (CFS2). We transmit the rightmost 7 error positions (out of 9). The verification algorithm starting with the left positions will be relatively fast in average, less than one millisecond. The average signature length is $\log_2\left(t!\binom{n}{t-2}\right) = 118.1 < 119$ bits.

In all three cases, to obtain a constant length signature one should be able to upper bound the number of decoding attempts. This is not possible, however by adding 5 bits to the signature the probability of failing to sign a message is less than 2^{-46}, and with 6 bits it drops to 2^{-92}.

5.4 Related work

It seems that up till now the only signature scheme that allowed such short signatures was Quartz [14] based on HFE cryptosystem [13]. It is enabled by a specific construction that involves several decryptions in order to avoid the birthday paradox forgery described in 4.1 that runs in the square root of the exhaustive search. This method is apparently unique to multivariate quadratic cryptosystems such as HFE and works only if the best attack on the underlying trapdoor is well above the square root of the exhaustive search [13, 14]. Such is not the case for the syndrome decoding problems.

6 Asymptotic behavior

In order to measure the scalability of the system, we will examine here how the complexity for computing a signature and the cost of the best known attack evolve asymptotically. We consider a family of binary t-error correcting Goppa codes of length $n = 2^m$. These codes have dimension $k = n - tm$.

6.1 Signature cost

We need to make $t!$ decoding attempts, for each of these attempts we need the following.

1. *Compute the syndrome.* As we are using Niederreiter's scheme we already have the syndrome, we only need to expand it into something usable by the decoder for alternant codes, the vector needed has a size of $2tm$ bits and is obtained from the syndrome by a linear operation, this costs $O(t^2m^2)$ operations in $\mathbf{F}_2$.
2. *Solve the key equation.* In this part, we apply Berlekamp-Massey algorithm to obtain the locator polynomial $\sigma(z)$, this costs $O(t^2)$ operations in $\mathbf{F}_{2^m}$.
3. *Find the roots of the locator polynomial.* If the syndrome is decodable, the polynomial $\sigma(z)$ splits in $\mathbf{F}_{2^m}[z]$ and its roots give the error positions. Actually we only need to check that the polynomial splits: that is $\gcd(\sigma(z), z^{2^m} - z) = \sigma(z)$. This requires t^2m operations in $\mathbf{F}_{2^m}$.

We will assume that one operation in $\mathbf{F}_{2^m}$ requires m^2 operations in $\mathbf{F}_2$, the total number of operations in $\mathbf{F}_2$ to achieve a signature is thus proportional to $t!t^2m^3$.

6.2 Best attacks complexity

Decoding attacks. The best known (and implemented) attack by decoding is by Canteaut and Chabaud [4] and its asymptotic time complexity is (empirically) around $(n/\log_2 n)^{f(t)}$ where $f(t) = \lambda t - c$ is an affine function with λ not much smaller than 1 and c is a small constant between 1 and 2.

Good estimates of the asymptotic behavior of the complexity of the best known general decoding techniques are given by Barg in [2]. In fact, when the rate $R = k/n$ of the code tends to 1, the time *and space* complexity becomes $2^{n(1-R)/2(1+o(1))}$, which, for Goppa codes, gives $n^{t(1/2+o(1))}$.

Structural attack. Very little is know about the distinguishability of Goppa codes. In practice, the only structural attack [9] consists in enumerating all Goppa codes and then testing equivalence with the public key. The code equivalence problem is difficult in theory [15] but easy in practice [17]. There are $2^{tm}/t$ binary t-error correcting Goppa codes of length $n = 2^m$, because of the properties of extended Goppa codes [10, Ch. 12, §4] only one out of mn^3 must be tested and, finally, the cost for equivalence testing cannot be lower than $n(tm)^2$ (a Gaussian elimination). Putting everything together leads to a structural attack whose cost is not less than tmn^{t-2} elementary operations.

signature cost	$t!t^2m^3$
signature length[1]	$(t-1)m + \log_2 t$
verification cost[1]	t^2m
public key size	$tm2^m$
cost of best decoding attack	$2^{tm(1/2+o(1))}$
cost of best structural attack	$tm2^{m(t-2)}$

[1]One error position omitted

Table 5. Characteristics of the signature scheme based on a $(n = 2^m, k = n - tm, d \geq 2t + 1)$ binary Goppa code

6.3 Intrinsic strengths and limitations

In Table 5 all complexities are expressed in terms of t and $m = \log_2 n$ and we may state the following facts:

- the signature cost depends exponentially of t,
- the public-key size depends exponentially of m,
- the security depends exponentially of the product tm.

From this we can draw the conclusion that if the system is safe today it can only be better tomorrow, as its security will depend exponentially of the signature size. On the other hand the signature cost and the key size will always remain high, as we will need to increase t or m or both to maintain a good security level. However, relatively to the technology, this handicap will never be as important as it is today and will even decrease rapidly.

7 Security arguments

In this section we reduce the security of the proposed scheme in the random oracle model to two basic assumptions concerning hardness of general purpose decoding and pseudo-randomness of Goppa codes. We have already measured the security in terms of the work factor of the best known decoding and structural attacks. We have seen how the algorithmic complexity of these attacks will evolve asymptotically. The purpose of the present section is to give a formal proof that breaking the CFS signature scheme implies a breakthrough in one of two well identified problems. This reduction gives an important indication on where the cryptanalytic efforts should be directed.

One of these problem is decoding, it has been widely studied and a major improvement is unlikely in the near future. The other problem is connected to the classification of Goppa codes or linear codes in general. Classification issues are in the core of coding theory since its emergence in the 50's. So far nothing significant is known about Goppa codes, more precisely there is no known property invariant by permutation and computable in polynomial time which characterizes Goppa codes. Finding such a property or proving that none exists would be an important breakthrough in coding theory and would also probably seal the fate, for good or ill, of Goppa code-based cryptosystems.

7.1 Indistinguishability of permuted Goppa codes

Definition 7.1.1 (Distinguishers) *A T-time distinguisher is a probabilistic Turing machine running in time T or less such that it takes a given F as an input and outputs $\mathcal{A}^F$ equal to* 0 *or* 1*. The probability it outputs 1 on F with respect to some probability distribution $\mathcal{F}$ is denoted as:*

$$Pr[F \leftarrow \mathcal{F} : \mathcal{A}^F = 1]$$

Definition 7.1.2 ((T, ε)-PRC) *Let $\mathcal{A}$ be a T-time distinguisher. Let* $\textsc{rnd}(n, k)$ *be the uniform probability distribution of all binary linear (n, k)-code. Let $\mathcal{F}(n, k)$ be any other probability distribution. We define the distinguisher's advantage as:*

$$Adv_{\mathcal{F}}^{PRC}(\mathcal{A}) \stackrel{def}{=} \left| Pr[F \leftarrow \mathcal{F}(n, k) : \mathcal{A}^F = 1] - Pr[F \leftarrow \textsc{rnd}(n, k) : \mathcal{A}^F = 1] \right|.$$

*We say that $\mathcal{F}(n, k)$ is a (T, ε)-***PRC** *(Pseudo-Random Code) if we have:*

$$\max_{T-\text{time } \mathcal{A}} \ Adv_{\mathcal{F}}^{PRC}(\mathcal{A}) \quad \leq \varepsilon.$$

7.2 Hardness of decoding

In this section we examine the relationships between signature forging and two well-known problems, the *syndrome decoding problem* and the *bounded-distance decoding problem*. The first is NP-complete and the second is conjectured NP-hard.

Definition 7.2.1 (Syndrome Decoding - SD)

Instance: *A binary $r \times n$ matrix H, a word s of $\mathbf{F}_2^r$, and an integer $w > 0$.*
Problem: *Is there a word x in $\mathbf{F}_2^n$ of weight $\leq w$ such that $Hx^T = s$?*

This decision problem was proven NP-complete [3]. Achieving complete decoding for any code can be done by a polynomial (in n) number of calls to SD. Actually the instances of SD involved in breaking code-based systems are in a particular subclass of SD where the weight w is bounded by the half of the minimum distance of the code of parity check matrix H. Is has been stated by Vardy in [16] as:

Definition 7.2.2 (Bounded-Distance Decoding - BD)

Instance: *An integer d, a binary $r \times n$ matrix H such that every $d-1$ columns of H are linearly independent, a word s of $\mathbf{F}_2^r$, and an integer $w \leq (d-1)/2$.*
Problem: *Is there a word x in $\mathbf{F}_2^n$ of weight $\leq w$ such that $Hx^T = s$?*

It is probably not NP because the condition on H is NP-hard to check. However several prominent authors [1, 16] conjecture that BD is NP-hard.

Relating signature forging and BD. An attacker who wishes to forge for a message M a signature of weight t with the public key H, has to find a word of weight t whose syndrome lies in the set $\{h(M,i) \mid i \in \mathbf{N}\}$ where $h()$ is a proper cryptographic hash function (see §4). Under the random oracle model, the only possibility for the forger is to generate any number of syndromes of the form $h(M,i)$ and to decode one of them this cannot be easier than $\mathrm{BD}(2t+1, H, h(M,i), t)$ for some integer i.

Relating signature forging and SD. Let us consider the following problem:

Definition 7.2.3 (List Bounded-Distance Decoding - LBD)

Instance: *An integer d, a binary $r \times n$ matrix H such that every $d-1$ columns of H are linearly independent, a subset S of $\mathbf{F}_2^r$, and an integer $w \le \lfloor (d-1)/2 \rfloor$.*
Problem: *Is there a word x in $\mathbf{F}_2^n$ of weight $\le w$ such that $Hx^T \in S$?*

Using this problem we will show how we may relate the forging of a signature to an instance of SD:

- In practice the forger must at least solve $\mathrm{LBD}(2t+1, H, S, t)$ where $S \subset \{h(M,i) \mid i \in \mathbf{N}\}$. The probability for the set S to contain at least one correctable syndrome is greater than $1 - \mathbf{e}^{-\lambda}$ where $\lambda = |S|\binom{n}{t}/2^r$. This probability can be made arbitrarily close to one if the forger can handle a set S big enough.
- Similarely, from any syndrome $s \in \mathbf{F}_2^r$, one can derive a set $R_{s,\delta} \subset \{s + Hu^T \mid u \in \mathbf{F}_2^n, w_H(u) \le \delta\}$ where $\delta = d_{vg} - t$ and d_{vg} is an integer such that $\binom{n}{d_{vg}} > 2^r$. With probability close to $1 - \mathbf{e}^{-\mu}$ where $\mu = |R_{s,\delta}|\binom{n}{t}/2^r$, we have $\mathrm{LBD}(2t+1, H, R_{s,\delta}, t) = \mathrm{SD}(H, s, d_{vg})$. Thus solving $\mathrm{LBD}(2t+1, H, R_{s,\delta}, t)$ is at least as hard as solving $\mathrm{SD}(H, s, d_{vg})$.
- We would like to conclude now that forging a signature is at least as hard as solving $\mathrm{SD}(H, s, d_{vg})$ for some s. This would be true if solving $\mathrm{LBD}(2t+1, H, S, t)$ was harder than solving $\mathrm{LBD}(2t+1, H, R_{s,\delta}, t)$ for some s, which seems difficult to state. Nevertheless, with sets S and $R_{s,\delta}$ of same size, it seems possible to believe that the random set (S) will not be the easiest to deal with.

Though the security claims for our signature scheme will rely on the difficulty of $\mathrm{BD}(2t+1, H, s, t)$, it is our belief that it can reduced to the hardness of $\mathrm{SD}(H, s, d_{vg})$ (note that d_{vg} depends only of n and r, not of t). If we assume the pseudo-randomness of the hash function $h()$ and of Goppa codes these instances are very generic.

7.3 Security reduction

We assume that the permuted Goppa code used in our signature scheme is a $(T_{Goppa}, 1/2)$-**PRC**, i.e. it cannot be distinguished from a random code with an advantage greater than 1/2 for all adversaries running in time $< T_{Goppa}$.

We assume that an instance of $\mathrm{BD}(2t+1, H, s, t)$ where H and s are chosen randomly cannot be solved with probability greater than 1/2 by an adversary running in time $< T_{BD}$.

Theorem 7.3.1 (Security of CFS) *Under the random oracle assumption, a T-time algorithm that is able to compute a valid pair message+signature for CFS with a probability $\geq 1/2$ satisfies:*

$$T \geq \min\left(T_{Goppa}, T_{BD}\right).$$

Proof (sketch): Forging a signature is at least as hard as solving $\mathrm{BD}(2t+1, H, s, t)$ where $s = h(M, i)$ (see §7.2) and H is the public key. Under the random oracle assumption, the syndrome $h(M, i)$ can be considered as random. If someone is able to forge a signature in time $T < T_{BD}$, then with probability 1/2 the matrix H has been distinguished from a random one and we have $T \geq T_{Goppa}$. $\square$

8 Conclusion

We demonstrated how to achieve digital signatures with the McEliece public key cryptosystem. We propose 3 schemes that have tight security proofs in random oracle model. They are based on the well known hard *syndrome decoding problem* that after some 30 years of research is still exponential. The Table 6 summarizes the concrete security of our schemes compared to some other known signature schemes.

base cryptosystem	RSA	ElGamal	EC	HFE	McEliece/Niederreiter		
signature scheme	RSA	DSA	ECDSA	Quartz	CFS1	CFS2	CFS3
data size(s)	1024	160/1024	160	100	144		
security							
structural problem	factoring	DL(p)	Nechaev group?	HFEv-	Goppa $\stackrel{?}{=}$ PRCode		
best structural attack	$\mathbf{2^{102}}$	2^{102}	∞	$> 2^{97}$	2^{119}		
inversion problem	RSAP	DL(q)	EC DL	MQ	SD		
best inversion attack	2^{102}	$\mathbf{2^{80}}$	$\mathbf{2^{80}}$	2^{100}	$\mathbf{2^{83}}$		
efficiency							
signature length	1024	320	321	128	132	119	**81**
public key [kbytes]	0.2	0.1	0.1	71	1152		
signature time 1 GHz	9 ms	1.5 ms	5 ms	15 s	$10 - 30$ s		
verification time 1 GHz	9 ms	2 ms	6 ms	40 ms	< 1 μs	< 1 ms	≈ 1s

Table 6. McEliece compared to some known signature schemes

The proposed McEliece-based signature schemes have unique features that will make it an exclusive choice for some applications while excluding other. On one hand, we have seen that both key size and signing cost will remain high, but will evolve favorably with technology. On the other hand the signature length and verification cost will always remain extremely small. Therefore if there is no major breakthrough in decoding algorithms, it should be easy to keep up with the Moore's law.

References

1. A. Barg. Some new NP-complete coding problems. *Problemy Peredachi Informatsii*, 30:23–28, 1994 (in Russian).
2. A. Barg. *Handbook of Coding theory*, chapter 7 – Complexity issues in coding theory. North-Holland, 1999.
3. E. R. Berlekamp, R. J. McEliece, and H. C. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3), May 1978.
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*, 44(1):367–378, January 1998.
5. N. Courtois, M. Finiasz, and N. Sendrier. How to achieve a McEliece-based digital signature scheme. Cryptology ePrint Archive, Report 2001/010, February 2001. `http://eprint.iacr.org/` et RR-INRIA 4118.
6. K. Kobara and H. Imai. Semantically secure McEliece public-key cryptosystems -Conversions for McEliece PKC-. In *PKC'2001*, LNCS, Cheju Island, Korea, 2001. Springer-Verlag.
7. P. J. Lee and E. F. Brickell. An observation on the security of McEliece's public-key cryptosystem. In C. G. Günther, editor, *Advances in Cryptology – EUROCRYPT'88*, number 330 in LNCS, pages 275–280. Springer-Verlag, 1988.
8. Y. X. Li, R. H. Deng, and X. M. Wang. On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. *IEEE Transactions on Information Theory*, 40(1):271–273, January 1994.
9. P. Loidreau and N. Sendrier. Weak keys in McEliece public-key cryptosystem. *IEEE Transactions on Information Theory*, 47(3):1207–1212, April 2001.
10. F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.
11. R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN Prog. Rep.,* Jet Prop. Lab., California Inst. Technol., Pasadena, CA, pages 114–116, January 1978.
12. H. Niederreiter. Knapsack-type crytosystems and algebraic coding theory. *Prob. Contr. Inform. Theory*, 15(2):157–166, 1986.
13. J. Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms. In *Eurocrypt'96*, LNCS, pages 33–48, 1996.
14. J. Patarin, L. Goubin, and N. Courtois. 128-bit long digital signatures. In *Cryptographers' Track Rsa Conference 2001*, San Francisco, April 2001. Springer-Verlag. to appear.
15. E. Petrank and R. M. Roth. Is code equivalence easy to decide? *IEEE Transactions on Information Theory*, 43(5):1602–1604, September 1997.

16. A. Vardy. The Intractability of Computing the Minimum Distance of a Code. *IEEE Transactions on Information Theory*, 43(6):1757–1766, November 1997.
17. N. Sendrier. Finding the permutation between equivalent codes: the support splitting algorithm. *IEEE Transactions on Information Theory*, 46(4):1193–1203, July 2000.
18. J. Stern. A method for finding codewords of small weight. In G. Cohen and J. Wolfmann, editors, *Coding theory and applications*, number 388 in LNCS, pages 106–113. Springer-Verlag, 1989.
19. J. Stern. A new identification scheme based on syndrome decoding. In D. R. Stinson, editor, *Advances in Cryptology - CRYPTO'93*, number 773 in LNCS, pages 13–21. Springer-Verlag, 1993.
20. J. Stern. Can one design a signature scheme based on error-correcting codes ? In *Asiacrypt 1994*, number 917 in LNCS, pages 424–426. Springer-Verlag, 1994. Rump session.