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 Research Unit
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] Ecole 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 that RSA, it is remarkable that all known attacks are still exponential. It is widely believed that McEliece does not allow practical digital signatures. In the present paper we disprove this belief and show several ways to build a practical signature scheme based on McEliece. The security is provably 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 are able to propose a scheme with signatures of 111-bits and a binary workfactor of 2^{85}.

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 the McEliece cryptosystem or Niederreiter's variant [12].

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 unfortunately subexponential. 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 [2], 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.

Thus it would be very interesting to dispose of signature schemes based on such (supposedly) hard 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 to achieve a signature scheme through complete decoding. In the three subsequent sections we present several practical implementations:

[A] is based on the Niederreiter's variant of the McEliece scheme (§4). It is equivalent from the security point of view and much more practical. For the proposed values the signature length will be 151 bits for a binary workfactor of 2^{83}.

[B] is a modified version of the signature scheme in which the complete decoding is replaced by randomized partial decoding (§5). For the same public key it achieves better security of 2^{85} and a signature length of 150 bits.

[C] is an improved version of [B] we consider in §6. We establish an advantageous tradeoff between signature length and verification time. For the same security the signature may be of 111 bits and even shorter.

The workfactors given above are not merely based on a best attack we can think of, but proven lower bounds with respect to a carefully studied famous *syndrome decoding problem*. In the Appendix of the paper we show how the security of [A-C] can be provably reduced to two very precise assumptions, motivated by the current state of knowledge on Goppa codes and general linear codes. We provide a straightforward, complete proof of security in the random oracle model that works in both asymptotic and concrete complexity setting [1].

Specific upper and lower bounds on a best attack to forge a signature for [A-C] are given.

2 Signature with McEliece

2.1 The McEliece scheme

Let $GF(2)$ 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 $GF(2)^n$. Elements of $GF(2)^n$ are called words, and elements of C are codewords. A code is usually given in the form of a generating matrix, lines of which form a basis of the code. A permuted code is a code such that the columns of the generating matrix has been permuted. The distance between two words of $GF(2)^n$ will be the Hamming distance, that is the number of positions in which they differ. The weight of a word of $GF(2)^n$ is its Hamming distance to the all-zero word.

The McEliece public-key cryptosystem is based on the difficulty of decoding linear codes, that is finding the nearest codeword to a given word. The secret key is a binary linear code, for which a fast decoding procedure for correcting up to t errors is known, and the public key is a random permutation of the columns of its generating matrix. The encryption process consists of multiplying the cleartext by the public key and adding a random error pattern of weight t. The decryption consists of correcting the error by use of the trapdoor fast decoding procedure (see Table 1).

Secret key:	– C a binary t-error correcting linear code. – a $k \times k$ non-singular matrix S, – a $n \times n$ permutation matrix P.
Public key:	$G' = SGP$, where G is a generating matrix of C.
Encryption:	$m \mapsto mG' + e$, where e is a random word of weight t.
Decryption:	$y \mapsto \Phi_C(yP^{-1})S^{-1}$, where $\Phi_C(z)$ is the (only) element of C at distance t of z.

Table 1. McEliece cryptosystem

The security of the system is based on two assumptions:

– decoding 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 [2], and by extensive research on general purpose decoders [8, 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 McEliece or any other cryptosystem based on decoding the point 2 fails. The reason is that if one considers a random word of length n, it usually is at distance greater than the decoding capacity t of the code. 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 word of the space: find an algorithm for complete decoding. It is the object of the next section, while a different solution will be presented in §6.

2.3 Complete decoding

Complete decoding consists of finding a nearest codeword to any given word of the space. Thus we decode not only words within the spheres of radius t around the codewords, but also anv word not covered by these spheres.

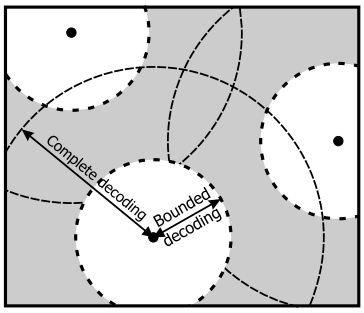

Fig. 1. from bounded decoding to complete decoding

The first step towards complete decoding is to decode $t+1$ errors instead of t. The simplest way to do this is to change a random bit in our word and then try to decode it. Suppose we have a word that differs in at least $t+1$ different positions from any codeword. We change one of its bits. If it can be decoded this means that the bit we changed was one of the $t+1$ bits of difference with a nearest code, as only then the t errors left could have been decoded with our

bounded decoding algorithm. If it can't be decoded it means the bit we changed wasn't one of the different bits and we just have to try again changing another bit.

If we try to flip every bit in turn, we are sure to decode any word with $t+1$ errors. If we want to correct $t+2$ errors we can do exactly the same changing two random bits each time. With this method, for any δ, we can decode words with $t+\delta$ errors, except that the greater δ is, the longer the decoding will be, as we will have to try a lot of different combinations of δ random bits. If n is the length of the word containing $t+\delta$ errors we want to decode the probability of success for each try is:

$$\mathcal{P} = \frac{\binom{t+\delta}{\delta}}{\binom{n}{\delta}},$$

which grows exponentially with δ. Therefore, with a large δ the average number of tries of our algorithm will be intractable.

It is important to note that with this algorithm we are able to find, for a given word, a codeword at a distance at most $t+\delta$ (if there exists one), but it isn't a true decoding, as the word we get isn't necessarily a nearest codeword.

In the next chapter we evaluate δ, it should be small and such that any word can be decoded. The maximum distance to a given linear code is called the covering radius, and it is usually hard to determine it exactly. In fact, it is an old and difficult open problem for Goppa codes. However, we are not interested in its precise value: consider the smallest integer δ_{min} for which the volume of a sphere of radius $t+\delta_{min}$ is greater than 2^{n-k}. This value is only a lower bound for the covering radius but as we will see, in some cases it is good enough in practice. Our goal is not exactly to achieve the complete decoding, but to be able to find a codeword at a distance less than $t+\delta_{min}$ for all codewords except for a negligible number. We call it *almost complete decoding.*

3 Finding the proper parameters

The parameters of the problem are: the length k of the messages (dimension of the code), the length n of the words (length of the code) 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 probability to obtain a signature for any document... We start by exploring the reasons why the classical McEliece parameters are not acceptable and on what is that we wish to obtain.

3.1 Need for new parameters

With the classical McEliece parameters ($n = 1024$, $k = 524$, $t = 50$) we obtain a value for δ_{min} of:

$$\delta_{min} = \min\left\{\delta \in \mathbb{N} \;\middle|\; \sum_{i=0}^{50+\delta} \binom{1024}{i} > 2^{1024-524}\right\} = 61$$

This would lead to a probability of success for our extended decoding of:

$$\mathcal{P} = \frac{\binom{111}{61}}{\binom{1024}{61}} \simeq 2^{-222}$$

It by far too little and moreover the value of 61+50 of $\delta_{min} + t$ is probably smaller than the real value of the covering radius. It seems therefore necessary to use other parameters, especially parameters for which δ_{min} would be smaller.

3.2 Parameters of Goppa codes

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 $n \leq 2^m$. Choosing $n < 2^m$ is not interesting except for tuning the tradeoff between key size and security. Thus in practice, we will consider t-error correcting Goppa codes of length $n = 2^m$ and dimension $k = 2^m - tm$. The number of Goppa codes with such parameters is following [9, page 97] equal to the number of monic irreducible polynomials of degree t over $GF(2^m)$, that is approximately $2^{tm}/t$.

3.3 Optimizing parameters

In this section we are looking for parameters which lead to a small δ_{min} and for which the average number of tries necessary to decode a word is small. The theoretical number of tries Σ to find a particular codeword at distance $t + \delta_{min}$ is the inverse of the probability $\mathcal{P}$ of choosing a good set of random bits. As there might be more than one such codeword (say an average of σ) the average number of tries will be Σ/σ. The *average* value of σ is close to $\binom{n}{t+\delta_{min}}/2^{n-k}$. This number turns out to be consistent with experimental values. Various size of parameters, decoding cost and cryptanalysis cost are given in Table 2.

length	error weight		cost		failure	binary workfactor	
$n = 2^m$	t	δ_{min}	Σ/σ	σ	rate$^{(a)}$	CC$^{(b)}$	LB$^{(b)}$
2048	8	2	41197	1.1	0.35	$2^{51.0}$	$2^{50.9}$
(2^{11})	9	3	374238	17.3	2^{-25}	$2^{53.1}$	$2^{50.1}$
	10	3	3764444	1.3	0.29	$2^{59.3}$	$2^{57.0}$
16384	8	2	40428	73.7	2^{-106}	$2^{64.4}$	$2^{67.0}$
(2^{14})	9	2	364078	6.7	$2^{-9.6}$	$2^{73.5}$	$2^{76.4}$
	10	3	3643228	702.9	2^{-1014}	$2^{76.6}$	$2^{75.4}$
32768	8	2	40374	295.4	2^{-426}	$2^{66.6}$	$2^{72.5}$
(2^{15})	9	2	363478	26.8	2^{-38}	$2^{78.3}$	$2^{82.8}$
	10	2	3636006	2.2	0.12	$2^{86.6}$	$2^{93.0}$
65536	8	2	40347	1182.7	2^{-1706}	$2^{68.5}$	$2^{78.1}$
(2^{16})	9	2	363179	107.5	2^{-155}	$2^{83.2}$	$2^{89.2}$
	10	2	3632401	8.9	2^{-13}	$2^{89.8}$	$2^{100.3}$

$^{(a)}$ estimated probability of not being able to decode $\approx e^{-\sigma}$
$^{(b)}$ derived from [4] and [8] and divided by σ

Table 2. Cost for decoding

3.4 Secure parameters

A fast bounded decoding algorithm can perform about one million decoding in a few minutes[1], so if we want to have an algorithm able to compute a signature in a decent time we see that 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 two last columns of Table 2 show the binary workfactors for both the Canteaut-Chabaud attack [4] and the Lee-Brickell attack [8] on McEliece cryptosystem. 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 is about 8 times faster and a negiglibible probability of decoding failure of 2^{-155} instead of 0.12 for the second choice.

3.5 First signature scheme based on decoding

The secret key of our algorithm is a randomly selected Goppa code C with $n = 65536$, $k = 65392$, such that we are able to decode it for $t = 9$ and $\delta_{min} = 2$. The public key is the permuted version G' of the generating matrix G of C.

We compute signatures as follows:

- hash the document into a word of n bits,
- use the complete decoding algorithm with 2 random bits to find a codeword at distance 11 from the hashed value
- get the k bits long message corresponding to this codeword and use it as signature

To check the signature we need to recode the k-bits long message using the public generating matrix G, hash the document using the public hash function and compare the two words. The signature will be considered as valid if they differ on at most 11 out of n positions.

This signature scheme achieves good security, but the signatures have 65392 bits, about... 500 times what we are looking for.

4 Using syndromes

In order to get a practical signature scheme we will switch to the Niederreiter version of McEliece cryptosystem summarized in Table 3. Instead of using a generating matrix of C it uses a dual representation in terms of so called parity check matrix. A $(n-k) \times n$ binary matrix is called a *parity check matrix* of C if the code C is exactly the set of codewords x satisfying $Hx^T = 0$. The *syndrome*

[1] our implementation performs one million decoding in 5 minutes, we estimate that it can be reduced by a factor 10

of any word y in $GF(2)^n$ is equal to Hy^T. It is used to detect and correct errors. Indeed, decoding consists of finding a word with a minimum number of modifications that would give a all-zero syndrome. Due to the linearity of the syndrome it amounts to adding to the received word an error pattern of the smallest weight which gives this very syndrome Hy^T. Thus finding a word of the smallest weight (that will be actually $\leq t+\delta$) that has a given syndrome, is perfectly equivalent to decoding.

Secret key: – C a binary t-error correcting linear code. – a $(n-k)\times(n-k)$ non-singular matrix S, – a $n \times n$ permutation matrix P.
Public key: $H' = SHP$, where H is a parity check matrix of C.
Encryption: $m \mapsto H'm^T$, the message m is a word of weight t.
Decryption: $y \mapsto \Psi_C(S^{-1}y)P$, where $\Psi_C(s)$ is the (only) word x of weight t and of syndrome s.

Table 3. Niederreiter cryptosystem

To compute a signature, instead of hashing the document into a word of length n, we hash it into a much shorter $n-k$ bits syndrome. Instead of decoding some word of length n with a given syndrome, the signature is computed as the small weight error pattern corresponding to this syndrome. If, as in our cases, $n-k \ll n$ it results in a much less cumbersome signature scheme with a substantial reduction in the speed, hashed message size, signature length and the public key size.

4.1 Implementing syndrome signatures [A]

In the Niederreiter version the signature will be a word of weight 11 and length 65536 (instead of a message of length 65392). We call [**A**] this precise instance of the signature scheme.

To sign a document one just has to decode a random syndrome obtained by hashing. Following §3.4 and Table 2 we evaluated the time for computing one signature on a 1 Ghz PC to be of 10 seconds. Then the signature of weight 11 is given in a compressed form. The naive method consists of writing the indexes of the 11 bits: as the word is 2^{16} bits long this would take $11 \times 16 = 176$ bits. A better compression is achieved if we number all the words of weight 11 and use the corresponding number x as a signature. The word with 1's at positions $i_1 < i_2 < \ldots < i_t$ is encoded as:

$$x = \binom{n-i_1}{t} + \binom{n-i_2}{t-1} + \ldots + \binom{n-i_t}{1} + 1. \tag{1}$$

The length of signatures is exactly $\lceil \log_2 \binom{65536}{11} \rceil = 151$ bits.

The verification will first recover the word of legth 11. Fast decoding of (1) is achieved by precomputing and storing the binomial coefficients in t binary trees. It takes tm integer operations, *i.e.* less than 1 μs. Then we compute the syndrome which consists of adding only 11 columns of the parity check matrix and takes less than $0.1\mu s$ (see §6.1). Excluding hashing, the whole verification is expected to take less than 1 μs.

4.2 Attacks on the signature length

Having such short signatures enables attacks independent on the strength of the trapdoor function used, and 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. One might consider this attack as a threat to our system, but it is not, as the required memory is about $2^{72} \times 72$ bits i.e. more than 10^{11} Terabytes. It is possible to avoid the above attack and achieve even shorter signatures, in §6 we propose such a method and compare with another known solution.

5 Improving security workfactor

A relative performance of McEliece-based signature schemes should be evaluated with respect to their main drawback which is the public key size. Our goal is to achieve better performance while the public key size remains the same. The present section improves the security against inversion attacks on the trapdoor function, the following section will improve on signature length. In both we use the same code as in [A] with $m = 16$, $t = 9$, $n = 2^m$ and $k = n - mt$.

5.1 Improving security against inversion

Let H' be the (public) parity check matrix of Niederreiter signature scheme called [A]. Let $W(2^m, \leq w) \subset GF(2)^{2^m}$ be the set of all codewords of the weight at most w. We define two functions based on H' with a different domain:

$$\begin{aligned} H_A &: W(2^m, \leq t + \delta_{min}) \to GF(2)^{mt}, \qquad H_A \stackrel{def}{=} x \mapsto H'x^T \\ H_B &: W(2^m, \leq t) \qquad\quad \to GF(2)^{mt}, \qquad H_B \stackrel{def}{=} x \mapsto H'x^T \end{aligned}$$

The function H_A, is precisely the trapdoor function used in [A] signature scheme: $\sigma = H_A^{-1}[\mathrm{MD}(m)]$. The value δ_{min} have been chosen so that it is invertible except for a negligible number of syndromes. The inversion is slow.

The second function H_B is invertible only on it's range, using standard decoding of the Goppa code, which is quite fast. Let $H_B^{-1} : GF(2)^{mt} \to W(2^m, \leq t) \cup \perp$ be a partial inverse that returns $\perp$ if it fails. In [A] we do an (almost) complete decoding, in [B] we use randomized incomplete decoding. For a given $r \in \mathbb{N}$ we try to decode 2^r times as follows:

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

```
σ ← ⊥
for all(T ∈ {0,1}^r) do
    if  H_B^{-1}[MD(m||T)] ≠ ⊥)
    {
        σ ← H_B^{-1}[MD(m||T)] || T
        break
    }
return σ
```

Table 4. The randomized McEliece Signature Scheme [**B**]

The density of the decodable syndromes is

$$\rho = \frac{2^{mt}}{\sum_{i=0}^{t} \binom{n}{i}}$$

After trying all possible 2^t values for T, the chance it still doesn't work is:

$$(1-\rho)^{2^r}$$

In our example $r = 24$ we get a probability of success for [B] of $1 - 2^{-66}$ and for $r = 25$ it is $1 - 2^{-133}$. For $r = 24$ the compressed size of signatures in [B] is:

$$r + \lceil \log_2(\sum_{i=0}^{t} \binom{n}{i}) \rceil = 150 \text{ bits.}$$

6 Short signatures with McEliece

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 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 5 below.

6.1 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].

[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

6.2 Decoding one error

We assume now that instead of z, the signature is a word u of weight $t-1$ such that $u+z$ has weight one. If $y = s + Hu^T$ is a column of H then s is the syndrome of a word of weight at most t, and the signature can be accepted.

Computing y requires $t-1$ column operations, finding the index of y should take no more than one column operation if the columns of H are properly stored. If the index does not exist the verification fails, else it works. The total complexity is t column operations, exactly the same as for a normal verification.

6.3 Decoding two errors

The word u used as signature has weight $t-2$. Let $y = s + Hu^T$ and let x be the word of weight 2 such that $Hx^T = y$. 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 leave in priority the positions which will be tested first, this divides the complexity in average by t.

6.4 Decoding more errors

In general, to correct w errors, 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 $2\binom{n}{w-1}$ column operations.

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

remaining errors	cost [a] of verification		signature length
0	t	9	150 bits
1	t	9	137 bits
2	$2n/t$	2^{14}	124 bits
3	$2\binom{n}{2}/\binom{t}{2}$	2^{27}	111 bits
4	$2\binom{n}{3}/\binom{t}{3}$	2^{40}	98 bits
5	$2\binom{n}{4}/\binom{t}{4}$	2^{53}	84 bits

[a] in column operations ($\approx$ 4 to 8 CPU clocks).

Table 5. Tradeoffs for the 9-error correcting Goppa code of length 2^{16}

If we allow verification times of few hours, it is possible to propose a scheme with signature length of 98 bits. Moreover, with variable length encoding, we may achieve an average length of 92 bits with a security binary workfactor of 2^{85}.

6.5 Proposed short signature scheme [C]

We call [C] the signature scheme with $w = 3$. The signatures have 111 bits and the verification takes less than one second.

6.6 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 4.2 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.

7 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 following summarizes the concrete security of our schemes compared to some other known signature schemes.

base cryptosystem	RSA	ElGamal	EC	HFE	McEliece			
signature scheme	RSA	DSA	ECDSA	Quartz	§2	[A]	[B]	[C]
data size(s)	1024	160/1024	160	100	65536	144	144	144
security								
structural problem	factoring	DL(p)	Nechaev group ?	HFEv-	Goppa $\stackrel{?}{=}$ PRCode			
best structural attack	$\mathbf{2^{102}}$	2^{102}	∞	$> 2^{97}$	2^{149}	2^{149}	2^{149}	2^{149}
inversion problem	RSAP	DL(q)	EC DL	MQ	R-SD			
best inversion attack	2^{102}	$\mathbf{2^{80}}$	$\mathbf{2^{80}}$	2^{100}	$\mathbf{2^{83}}$	$\mathbf{2^{83}}$	$\mathbf{2^{85}}$	$\mathbf{2^{85}}$
an attack on the signature scheme itself	-	2^{80}	2^{80}	$\mathbf{2^{80}}$	-	2^{85}	2^{85}	2^{85}
efficiency								
signature length	1024	320	162	128	65392	151	150	**111**
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 s			
verification time 1 GHz	9 ms	2 ms	6 ms	40 ms	32 ms	$< 1\ \mu$s		1 s

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.

First, it seems difficult to build a cryptosystem based on McEliece and avoid it's major drawback; the public key size. However there are very few security schemes for which the security is provably reduced to (apparently) exponential problems. Therefore if there is no major breakthrough in decoding algorithms, it should be easy to keep up with the Moore's law.

Signature with McEliece has either faster verification than any known signature scheme [A,B], either it gives the shortest known digital signatures with 111 bits for [C], and possibly even 98 bits, as shown in Table 5.

References

1. M. Bellare: *Practice-oriented provable-security;* International Workshop on Information Security (ISW 97), LNCS 1396, Springer-Verlag 1998.
2. 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.
3. R. Canetti, O. Goldreich and S. Halevi: *The random oracle methodology, revisited (preliminary version);* ACM symposium on Theory of computing, ACM, pp. 209–218, May 24 - 26, 1998, Dallas, TX USA.
4. A. Canteaut, F. Chabaud: *A new algorithm for finding minimum-weight words in a linear code: application to McEliece's cryptosystem and to BCH Codes of length 511;* IEEE Transactions on Information Theory, 44(1):367–378, January 1998.
5. A. Canteaut, N. Sendrier: *Cryptoanalysis of the Original McEliece Cryptosystem. In Advances in Cryptology;* Asiacrypt 1998, LNCS 1514, pp.187-199.
6. *I. I. Dumer; Suboptimal decoding of linear codes: partition technique;* IEEE-IT 42(6), November 1996, pp. 1971 - 1986.
7. J. K. Gibson: *Equivalent Goppa codes and trapdoors to McEliece's public key cryptosystem;* In Advances in Cryptology, Eurocrypt'91; LNCS 547, pp. 517-521, Springer-Verlag.
8. P. J. Lee and E. F. Brickell. *An observation on the security of McEliece's public-key cryptosystem;* In Advances in Cryptology , Eurocrypt'88, LNCS 330, pp. 275–280. Springer-Verlag, 1988.
9. R. Lidl, H. Niederreiter: *Finite Fields*; Encyclopedia of Mathematics and its applications, Volume 20, Cambridge University Press, 1983.
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 Propulsion Laboratory, California Inst. Technol., Pasadena, CA, pp. 114–116, January 1978.
12. H. Niederreiter. *Knapsack-type cryptosystems and algebraic coding theory;* Prob. Contr. Inform. Theory, 15(2), pp. 157-166, 1986.
13. J. Patarin: *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of Asymmetric Algorithms;* Eurocrypt'96, LNCS, pp. 33-48.
14. J. Patarin, L. Goubin, N. Courtois: Quartz, **1**28-*bit long digital signatures*; Cryptographers' Track Rsa Conference 2001, San Francisco 8-12 Avril 2001, to appear in Springer-Verlag.
15. E. Petrank and R. M. Roth. *Is code equivalence easy to decide ?* IEEE Transactions on Information Theory, 43(5), pp. 1602-1604, September 1997.
16. N. Sendrier. *Finding the permutation between equivalent codes: the support splitting algorithm;* IEEE Transactions on Information Theory, 46(4), pp.1193-1203, July 2000.
17. D. Pointcheval, J. Stern: *Security arguments for Digital signatures and Blind Signatures;* Journal of Cryptology, Vol.13(3), Summer 2000, pp.361-396.
18. J. Stern. *A method for finding codewords of small weight;* Coding theory and applications, LNCS 388, pp. 106-113. Springer-Verlag, 1989.
19. J. Stern. *A new identification scheme based on syndrome decoding;* In Advances in Cryptology, Crypto'93, LNCS 773, pp. 13–21. Springer-Verlag, 1993.
20. J. Stern: *Can one design a signature scheme based on error-correcting codes ?;* Rump session Asiacrypt 1994, LNCS 917, pp.424-426.

8 Appendix - Security Proofs

In this section we give relative proofs of security for the proposed schemes [A-C]. The proofs are extremely simple using the random oracle model and two basic assumptions concerning hardness of general purpose decoding and pseudo-randomness of Goppa codes. Usually the security claims and proofs are formulated either in terms of asymptotic security or Bellare-style concrete security [1] with fixed workfactors e.g. 2^{80}. For most cryptosystems asymptotic complexity is defined in terms of polynomial or not algorithms. Not polynomial is meaningless when substituted with given parameter values. However in the case of McEliece-derived cryptosystems, but also for the elliptic curves, the security may be formulated in terms of exponential complexity. Indeed for these two cases there is hope that they might be really as secure, while many other public key cryptosystems prove sooner or later subexponential. We will formulate our asymptotic claims in terms of precise exponential bounds on workfactors extrapolated from known algorithms. Thus we obtain both concrete and asymptotic lower bounds on security of our signature schemes [A-C]. We also obtain a tight security bound on [A-C] when substituted with the minimum workfactors of the all known attacks on the two problems.

First we explain why Goppa codes are good codes and what does it mean.

8.1 Indistinguishability of permuted Goppa codes

Definition 8.1.1 (Distinguishers). A T-time distinguisher is a T-time Adersary, i.e. a probabilistic Turing machine, such that it takes a given F as an input and outputs 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:

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

Definition 8.1.2 ((T,ε)-PRC). Let A be a T-time distinguisher. Let $\mathrm{RND}(n,k)$ be the uniform probability distribution of a random linear (n,k)-code over $GF(2)$. Let $\mathcal{F}(n,k)$ be any other probability distribution. We define the distinguisher's advantage as:

$$Adv_{\mathcal{F}}^{PRC}(A) \stackrel{def}{=} \Big| Pr[F \leftarrow \mathcal{F}(n,k) : A^F = 1] - Pr[F \leftarrow \mathrm{RND}(n,k) : A^F = 1] \Big|.$$

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

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

Design Criterion 8.1.3 (PRC trapdoors). A good cryptographic trapdoor function based on codes should be a PRC.

Let $\mathrm{Goppa}(2^m, 2^m - tm)$ be a probability distribution of a random permuted $(2^m, 2^m - tm)$ Goppa code over $GF(2)$. Following [9, page 97] the number of

(non-permuted) Goppa codes is equal to the number of monic irreducible polynomials of degree t over $GF(2^m)$ which gives about $2^{tm}/t$. The exact number of non-equivalent Goppa codes is not known. Following the current state of knowledge on identifying permuted Goppa codes we conjecture with a good margin of improvement that:

- the number of non-equivalent Goppa codes is at least $2^{(t-1)m}$ [7]
- solving CODE EQUIVALENCE PROBLEM [15] is in at least $Min(n, n-k)^3$ [16].

Conjecture 8.1.4 (Indistinguishability of permuted Goppa codes). If a T_{Goppa}-time adversary distinguishes a random permuted Goppa code with an advantage

$$Adv^{PRC}_{Goppa(2^m, 2^m - tm)} \geq 1/2$$

then the complexity is conjectured to be at least:

$$T_{Goppa} \geq 2^{(t-1)m} \cdot Min(n, n-k)^3.$$

A more precise evaluation is not necessary because as we will see later the number will be big enough.

8.2 Hardness of decoding

It is less obvious to give an estimation of the hardness of the *syndrome decoding* problem for a random code. Though all the known algorithms are exponential, they are faster than the exhaustive search and the evaluation of the best complexity for known attacks on a given instance is non-trivial.

There are several versions of the decoding problem:

Definition 8.2.1 (Random Syndrome Decoding R-SD). Find a word of a small weight $\leq t$ that gives one out of a list of random syndromes.

Definition 8.2.2 ((One) Decodable Syndrome Decoding D-SD). Find one word of a small weight $\leq t$ that produces a given randomly selected decodable syndrome.

The algorithms are usually made for D-SD. The situation is slightly different in [A], [B] and [C]. The adversary that wants to forge a signature may produce as many messages as he wants, and then try to decode one of the syndromes. All of them will be random as they are produced by a cryptographic hash function. Therefore solving R-SD allows to forge signatures.

R-SD and D-SD are different problems. It is obvious that for a fixed code the R-SD difficulty decreases with t, while for D-SD it seems to increase. The relationship between the 2 problems is not obvious. Let $\rho \approx \binom{n}{t}/2^{n-k}$ be the average number of words of weight $\leq t$ that correspond to a random syndrome. Any attack on D-SD is conjectured to give an algorithm for R-SD with

Conjecture 8.2.3 (R-SD vs D-SD).

$$\text{R-SD}(n,k,t) \in \left[\frac{\text{D-SD}(n,k,t)}{\sqrt{\rho}}, \frac{\text{D-SD}(n,k,t)}{\rho}\right].$$

Indeed if $\rho > 1$ and as the decoding algorithms have important brute search subcomponents, the chance that they succeed should be multiplied by ρ and the complexity decreases by $\frac{1}{\rho}$.

If $\rho < 1$ we must try the whole algorithm $1/\rho$ times in order to find a decodable syndrome. Still thare are good chances that we may also benefit from a birthday paradox-like tradoff between the number of syndromes to decode and the number of guesses giving rather $\frac{1}{\sqrt{\rho}}$.

Following many years of research on general algorithms for decoding [2, 4, 6, 8, 18] we conjecture that:

Conjecture 8.2.4 (Hardness of Syndrome Decoding). All the algorithms for solving D-SD(t) for a randomly selected (n,k)-code are exponential both on average and in worst case with a complexity of

$$T_{D-SD} \geq 2^{\alpha \cdot n \cdot e(\frac{k}{n})}$$

For some $\alpha > 0$ and some function $e : [0,1] \to [0,1]$ that varies from one attack to another [2, 4, 6, 8, 18].

We note that the above conjecture does not allow precomputation on the code and it may be easier in this case as demonstrated in §6.

A similar conjecture seems also plausible for R-SD if ρ is not too big.

8.3 Lower Bounds on Security

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

We assume that the corresponding instance of R-SD cannot be solved with probability greater than $\frac{1}{2}$ by an adversary running in time $< T_{R-SD}$.

Theorem 8.3.1 (Provable Security of [A] and [B]). Any T-time algorithm that is able to compute a valid pair message+signature for one of the schemes [A] or [B] with a probability $\geq \frac{1}{2}$ satisfies:

$$T \geq Min\left(T_{Goppa}, T_{R-SD}\right).$$

Proof. First we establish the fact for [A]. We suppose that such an adversary exists and computes a valid pair (m,σ) with

$$\sigma = H_A^{-1}[\text{MD}(m)].$$

We use a well known random oracle technique. It assumes that MD is a random function, and that the output of MD is random for each new entry. [4]

We assume that a T-time adversary is able to forge signatures with a probability $\geq \frac{1}{2}$. We may assume without loss of generality that the adversary only produces correct signatures, if any. Otherwise we simply add the public signature verification to the adversary. Since the output of MD is supposed to be random, an adversary able to give a valid signature for some message must have computed the hash function of the message, otherwise he cannot know that the signature is correct. Still as it is random, he cannot distinguish between outputs of MD. When we substitute it successively by several random syndromes to decode. If he manages to forge a signature, he must decode one of the given syndromes and it is exactly the R-SD problem as defined in 8.2.1.

If the code used is a random code, we have $T \geq T_{R-SD}$. Otherwise, we assume that $T < T_{R-SD}$ for a randomly selected permuted Goppa code. Using our adversary gives an algorithm in time T distinguishing between a random code and the permuted Goppa code with an advantage very close to $\frac{1}{2}$. Indeed, it will decode a random instance of a permuted Goppa code with probability $\frac{1}{2}$, and it has a negligible probability to do the same for a random code. Actually this probability is at most $\frac{1}{2T_{R-SD}}$. Thus $T \geq T_{Goppa}$.

Thus either $T \geq T_{R-SD}$ or $T \geq T_{Goppa}$, and $T \geq Min\,(T_{Goppa}, T_{R-SD})$.

The proof in the case [B] is exactly the same. □

Theorem 8.3.2 ([C] is provably as secure as [B]). Any T_C-time algorithm that is able to compute a false signature for the signature scheme [C] with probability $\frac{1}{2}$ is able to compute a signature for [B] with with probability $\frac{1}{2}$ and

$$T_B \leq T_C + \binom{n}{w-1} / \binom{t}{w-1}$$

Proof. We transform it into an algorithm that forges the signatures for [B] with the time increased by the verification time of the [C] that recovers a complete signature of type [B] which is at most the time we obtained in §6.

For values of w used in practice $\binom{n}{w-1}/\binom{t}{w-1} z$ is negligible compared to T_{R-SD} and we have also:

Corollary 8.3.3. If w is small, [C] is provably as secure as [B].

If we combine the theorems 8.3.1 and 8.3.2 we have:

Theorem 8.3.4 (Provable Security of [C]). Any T time algorithm that is able to compute a false signature for the signature scheme [C] for a given parameter w and with a probability $\geq \frac{1}{2}$ satisfies:

$$T \geq Min\,(T_{Goppa}, T_{R-SD}) - \binom{n}{w-1} / \binom{t}{w-1}$$

[4] In theory this approach is incorrect in asymptotic sense [3] if MD is a publicly known function ensemble. For this reason in [17] proofs in the random oracle model are called arguments. Still it is believed that schemes proven in random oracle model are correct and secure [17]

We note that an algorithm that solves R-SD gives also an upper bound on the security of the cryptosystem as it allows directly to compute signatures in T_{R-SD}. Thus we have:

Theorem 8.3.5 (Exact security of [A-B]). The minimum workfactor T necessary to forge signatures in schemes [A-B] with a probability $\geq \frac{1}{2}$ is in the range:

$$T_{R-SD} \geq T \geq Min\,(T_{Goppa}, T_{R-SD})$$

Theorem 8.3.6 (Exact security of [C]). For [C] we have

$$T_{R-SD} \geq T \geq Min\,(T_{Goppa}, T_{R-SD}) - \binom{n}{w-1} / \binom{t}{w-1}$$

Corollary 8.3.7 (Tight security of [A-C]). If (as it is currently believed) $T_{Goppa} > T_{R-SD}$ and for the case of [C], if also $\binom{n}{w-1}/\binom{t}{w-1}$ is small compared to T_{R-SD}, the security of schemes [A-C] is exactly

$$T = T_{R-SD}$$

8.4 Concrete security of [A], [B] and [C]

The code used in [A], [B] and [C] is the same with with $m = 16$, $t = 9$, $n = 2^m$, $k = n - mt$. The only difference is that we decode $t = 9$ errors in [B] and [C] and $t + \delta_{min} = 11$ for [A].

Following 8.1.4 above we have: $T_{Goppa} \geq 2^{(t-1)m} \cdot Min(n, n-k)^3 \approx 2^{149}$.

We evaluated the workfactor needed for decoding our pseudo-random codes D-SD$(2^{16}, 2^{16}-144, 9)$ and D-SD$(2^{16}, 2^{16}-144, 9)$ with the best known algorithm due to Anne Canteaut and Florent Chabaud [4]. The best variant of the algorithm gives a binary workfactor $T_{D-SD}(9) \approx 2^{76}$ and $T_{D-SD}(11) \approx 2^{90}$. Thus following the Conjecture 8.2.3 for which a different bound is smaller, depending whether $\rho > 1$ or not, we get:

$$T_{R-SD}(9) \geq \frac{2^{76}}{\sqrt{\rho_t}} \approx 2^{85} \qquad \text{and} \qquad T_{R-SD}(11) \geq \frac{2^{90}}{\rho_{t+\delta_{min}}} \approx 2^{83}$$

We have $T_{Goppa} > T_{R-SD}$ in both cases as with a good margin. As long as it remains true and following Theorem 8.3.7 the lower and upper bound on the security of the signature schemes [A], [B], [C] coincide and give:

$$WF_{[A]} = T_{R-SD}(11) = 2^{83}$$

$$WF_{[B]} = WF_{[C]} = T_{R-SD}(9) = 2^{85}$$