Code-based cryptography

Raphael Overbeck[1] and Nicolas Sendrier[2]

[1] EPFL, I&C, LASEC.
[2] INRIA Rocquencourt, projet SECRET.

1 Introduction

In this chapter, we consider the theory and the practice of code-based cryptographic systems. By this term, we mean the cryptosystems in which the algorithmic primitive (the underlying one-way function) uses an error correcting code $\mathcal{C}$. This primitive may consist in adding an error to a word of $\mathcal{C}$ or in computing a syndrome relatively to a parity check matrix of $\mathcal{C}$.

The first of those systems is a public key encryption scheme and it was proposed by Robert J. McEliece in 1978 [48]. The private key is a random binary irreducible Goppa code and the public key is a random generator matrix of a randomly permuted version of that code. The ciphertext is a codeword to which some errors have been added, and only the owner of the private key (the Goppa code) can remove those errors. Three decades later, some parameter adjustment have been required, but no attack is known to represent a serious threat on the system, even on a quantum computer.

Similar ideas have been used to design other cryptosystems. Among others, let us mention some public key systems, like the Niederreiter encryption scheme [52] or the CFS signature scheme [14], and also identification schemes [73, 76], random number generators [19, 30] or a cryptographic hash function [3]. Some of the most important of those proposals are reviewed in §2.

As for any class of cryptosystems, the practice of code-based cryptography is a trade-off between security and efficiency. Those issues are well understood, at least for McEliece's scheme. Even though, no practical application of code-based cryptography is known to us. This might partly be due to the large size of the public key (100 kilobytes to several megabytes), but maybe also to a lack of publicity in a context were alternative solution were not urgently needed. Anyway, apart from the key size that we already mentioned, the McEliece encryption scheme has many strong features. First, the security reductions are tight (see [38] for instance). Also, the system is very fast, as both encryption and decryption procedures have a low complexity.

We will discuss in details the two aspects of security in §3 and §4. The first security assumption is the hardness of decoding in a random linear code [6]. This is an old problem of coding theory for which only exponential time solutions are known [4]. The second security assumption, needed only for public key systems, is the indistinguishability of Goppa codes [66]. Though it is not as old, in this form, as the first one, it relates to old problems of algebraic coding theory and is believed to be valid.

We will conclude this chapter with some practical aspects, first on the implementation, then on the key size issue, and we finish with a key point for the practicality of McEliece and related systems: how to efficiently construct a semantically secure (IND-CCA2) variant.

2 Cryptosystems

The first cryptosystem based on coding theory was a public key encryption scheme, presented in 1978 by McEliece [48]. Nearly all subsequently proposed asymmetric cryptographic schemes based on coding theory have a common disadvantage: the large memory requirements. Several other schemes followed, as the identification scheme by Stern [73], hash functions [3], random number generators [19] and efforts to build a signature scheme. The latter however all failed (compare [79], [32], [1] and [74]), until finally in 2001 Courtois, Finiasz and Sendrier made a promising proposal [14]. However, even if the latter is not broken, it is not suited for standard applications since besides the public key sizes the signing costs are large for secure parameter sets.

In 1986, Niederreiter proposed a knapsack-type PKC based on error correcting codes. This proposal was later shown to have a security equivalent to McEliece's proposal [42]. Among others, Niederreiter estimated GRS codes as suitable codes for his cryptosystem which were assumed to allow smaller key sizes than Goppa codes. Unfortunately, in 1992 Sidelnikov and Shestakov were able to show that Niederreiter's proposal to use GRS codes is insecure. In the following a couple of proposals were made to modify McEliece's original scheme (see e.g. [27], [26], [28], [70] and [35]) in order to reduce the public key size. However, most of them turned out to be insecure or inefficient compared to McEliece's original proposal (see e.g. [54] or [38]). The most important modifications for McEliece's scheme are the conversions by Kobara and Imai in 2001. These are CCA2-secure, provably as secure as the original scheme [37] and have almost the same transmission rate as the original system.

The variety of possible cryptographic applications provides sufficient motivation to have a closer look at cryptosystems based on coding theory as an serious alternative to established PKCs like the ones based on number theory. In this section we will concentrate on the most important cryptographic schemes based on coding theory.

2.1 McEliece PKC

The McEliece cryptosystem we are going to present in this section remains unbroken in its original version, even if about 15 years after it's proposal security parameters had to be adapted. Although the secret key of the McEliece PKC is a Goppa code (see §6.2) in the original description, the secret key could be drawn from any subclass of the class of alternant codes. However, such a choice might not reach the desired security as we will see in the following sections. The trapdoor for the McEliece cryptosystem is the knowledge of an efficient error correcting algorithm for the chosen code class (which is available for each Goppa code) together with a permutation. The McEliece PKC is summarized in Algorithm 2.1.

Algorithm 2.1 The McEliece PKC

- **System Parameters:** $n, t \in \mathbb{N}$, where $t \ll n$.
- **Key Generation:** Given the parameters n, t generate the following matrices:

  G : $k \times n$ generator matrix of a code $\mathcal{G}$ over $\mathbb{F}$ of dimension k and minimum distance $d \geq 2t+1$. (A binary irreducible Goppa code in the original proposal.)
  S : $k \times k$ random binary non-singular matrix
  P : $n \times n$ random permutation matrix

  Then, compute the $k \times n$ matrix $\mathsf{G}^{\text{pub}} = \mathsf{SGP}$.
- **Public Key:** $\left(\mathsf{G}^{\text{pub}}, t\right)$
- **Private Key:** $(\mathsf{S}, D_{\mathcal{G}}, \mathsf{P})$, where $D_{\mathcal{G}}$ is an efficient decoding algorithm for $\mathcal{G}$.
- **Encryption ($E_{(\mathsf{G}^{\text{pub}},t)}$):** To encrypt a plaintext $\mathbf{m} \in \mathbb{F}^k$ choose a vector $\mathbf{z} \in \mathbb{F}^n$ of weight t randomly and compute the ciphertext $\mathbf{c}$ as follows:

$$\mathbf{c} = \mathbf{m}\mathsf{G}^{\text{pub}} \oplus \mathbf{z} \ .$$

- **Decryption ($D_{(S,D_{\mathcal{G}},P)}$):** To decrypt a ciphertext $\mathbf{c}$ calculate

$$\mathbf{c}\mathsf{P}^{-1} = (\mathbf{m}\mathsf{S})\,\mathsf{G} \oplus \mathbf{z}\mathsf{P}^{-1}$$

  first, and apply the decoding algorithm $D_{\mathsf{G}^{\text{pub}}}$ for $\mathcal{G}$ to it. Since $\mathbf{c}\mathsf{P}^{-1}$ has a hamming distance of t to $\mathcal{G}$ we obtain the codeword

$$\mathbf{m}\mathsf{S}\mathsf{G} = D_{\mathcal{G}}\left(\mathbf{c}\mathsf{P}^{-1}\right) \ .$$

  Let $J \subseteq \{1, \cdots, n\}$ be a set, such that $\mathsf{G}^{\text{pub}}_{\cdot J}$ is invertible, then we can compute the plaintext $\mathbf{m} = (\mathbf{m}\mathsf{S}\mathsf{G})_J \left(\mathsf{G}_{\cdot J}\right)^{-1} S^{-1}$

The choice of security parameters for the McEliece PKC has to be taken in respect to the known attacks. The optimal choice of parameters for a given security level (in terms of the public key size) unfortunately can not be given as a closed formula. We are going to discuss the latter later on. The problem

to attack the McEliece PKC differs from the general decoding problem, which we will examine in §3:

Problem 1. (McEliece Problem) Let $\mathbb{F} = \{0,1\}$ and $\mathcal{G}$ be a binary irreducible Goppa code in Algorithm 2.1.

- Given a McEliece public key $\left(\mathsf{G}^{\text{pub}}, t\right)$ where $\mathsf{G}^{\text{pub}} \in \{0,1\}^{k\times n}$ and a ciphertext $\mathbf{c} \in \{0,1\}^n$,
- Find the (unique) message $\mathbf{m} \in \{0,1\}^k$ s.t. $\text{wt}\left(\mathbf{m}\mathsf{G}^{\text{pub}} - \mathbf{c}\right) = t$.

It is easy to see that someone who is able to solve the Syndrome Decoding Problem (compare §3) is able to solve the McEliece problem. The reverse is presumably not true, as the code $\mathcal{G} = \left\langle \mathsf{G}^{\text{pub}} \right\rangle$ is not a random one, but permutation equivalent to a code of a known class (a Goppa code in our definition). We can not assume that the McEliece-Problem is $\mathcal{NP}$-hard. Solving the McEliece-Problem would only solve the General Decoding Problem in a certain class of codes and not for all codes.

In the case of McEliece's original proposal, Canteaut and Chabaud state the following: "The row scrambler S has no cryptographic function; it only assures for McEliece's system that the public matrix is not systematic otherwise most of the bits of the plain-text would be revealed" [11]. However, for some variants of McEliece's PKC, this statement is not true, as e.g. in the case of CCA2-secure variants (see §5.1 and §5.3) or in the case, where the messages are seeds for PRNGs. The matrix P is indispensable because for most codes the code positions are closely related to the algebraic structure of the code. We will come back to this in §4.3.

The Niederreiter variant

The dual variant of the McEliece PKC is a knapsack-type cryptosystem and is called the Niederreiter PKC. In difference to the McEliece cryptosystem, instead of representing the message as a codeword, Niederreiter proposed to encode it into the error vector by a function $\phi_{n,t}$:

$$\phi_{n,t} : \{0,1\}^{\ell} \to \mathcal{W}_{n,t}, \tag{1}$$

where $\mathcal{W}_{n,t} = \{\mathbf{e} \in \mathbb{F}_2^n \mid \text{wt}(\mathbf{e}) = t\}$ and $\ell = \lfloor \log_2 |\mathcal{W}_{n,t}| \rfloor$. Such a mapping is presented, e.g., in [19] and is summarized in Algorithm 2.2. This algorithm is quite inefficient and has complexity $\mathcal{O}(n^2 \cdot \log_2 n)$. Its inverse is easy to define: $\phi_{n,t}^{-1}(\mathbf{e}) = \sum_{i=1}^{n} e_i \cdot \binom{i}{\sum_{j=0}^{i} e_j}$. We discuss efficient alternatives in §5.1. Representing the message by the error vector, we get the dual variant of McEliece's cryptosystem, given in Algorithm 2.3. The security of the Niederreiter PKC and the McEliece PKC are equivalent. An attacker who can break one is able to break the other and vice versa [42]. In the following, by "Niederreiter PKC" we refer to the dual variant of the McEliece PKC and to the proposal by Niederreiter to use GRS codes by "GRS Niederreiter PKC".

Algorithm 2.2 $\phi_{n,t}$: Mapping bit strings to constant weight codewords

Input: $\mathbf{x} \in \{0,1\}^{\ell}$
Output: a word $\mathbf{e} = (e_1, e_2, \cdots, e_n)$ of weight w and length n.

$c \leftarrow \binom{n}{w}$, $c' \leftarrow 0$, $j \leftarrow n$.
$i \leftarrow$ Index of $\mathbf{x}$ in the lexicographic order (an integer).
while $j > 0$ **do**
  $c' \leftarrow c \cdot \frac{j-w}{j}$
  if $i \leq c'$ **then**
    $e_j \leftarrow 0$, $c \leftarrow c'$
  else
    $e_j \leftarrow 1$, $i \leftarrow i - c'$, $c \leftarrow c \cdot \frac{w}{n}$
  $j \leftarrow j - 1$

Algorithm 2.3 Niederreiter's PKC

- **System Parameters:** n, $t \in \mathbb{N}$, where $t \ll n$.
- **Key Generation:** Given the parameters n, t generate the following matrices:

  H: $(n-k) \times n$ check matrix of a code $\mathcal{G}$ which can correct up to t errors.
  P: $n \times n$ random permutation matrix

  Then, compute the systematic $n \times (n-k)$ matrix $\mathsf{H}^{\text{pub}} = \mathsf{MHP}$, whose columns span the column space of HP, i.e. $\mathsf{H}^{\text{pub}}_{\{1,\cdots,n-k\}.} = \mathsf{Id}_{(n-k)}$.
- **Public Key:** $\left(\mathsf{H}^{\text{pub}}, t\right)$
- **Private Key:** $(\mathsf{P}, D_{\mathcal{G}}, \mathsf{M})$, where $D_{\mathcal{G}}$ is an efficient syndrome decoding algorithm[1] for $\mathcal{G}$.
- **Encryption:** A message $\mathbf{m}$ is represented as a vector $\mathbf{e} \in \{0,1\}^n$ of weight t, called plaintext. To encrypt it, we compute the syndrome

$$\mathbf{s} = \mathsf{H}^{\text{pub}}\mathbf{e}^{\top} \ .$$

- **Decryption:** To decrypt a ciphertext $\mathbf{s}$ calculate

$$\mathsf{M}^{-1}\mathbf{s} = \mathsf{HPe}^{\top}$$

  first, and apply the syndrome decoding algorithm $D_{\mathcal{G}}$ for $\mathcal{G}$ to it in order to recover $\mathsf{Pe}^{\top}$. Now, we can obtain the plaintext $\mathbf{e}^{\top} = \mathsf{P}^{-1}\mathsf{Pe}^{\top}$

[1] A syndrome decoding algorithm takes as input a syndrome – not a codeword. Each syndrome decoding algorithm leads immediately to an decoding algorithm and vice versa.

The advantage of this dual variant is the smaller public key size since it is sufficient to store the redundant part of the matrix H^{pub}. The disadvantage is the fact, that the mapping $\phi_{n,t}$ slows down en- and decryption. In a setting, where we want to send random strings, only, this disadvantage disappears as we can take $h(\mathbf{e})$ as random string, where h is a secure hash function.

Modifications for the trapdoor of McEliece's PKC

From McEliece's scheme one can easily derive a scheme with a different trapdoor by simply replacing the irreducible binary Goppa codes by another code class. However, such attempts often proved to be vulnerable against structural attacks. In §4.3 we will sketch a few of those attacks. To prevent structural attacks not only McEliece's proposal, but others exist as well. In Table 1 we give an overview of the principal modifications. McEliece's proposal can thus

1. **Row Scrambler [48]:** Multiply G with a random invertible matrix $\mathsf{S} \in \mathbb{F}^{k\times k}$ from the right. As $\langle \mathsf{G} \rangle = \langle \mathsf{SG} \rangle$, one can use the known error correction algorithm. Publishing a systematic generator matrix provides the same security against structural attacks as a random S.
2. **Column Scrambler / Isometry [48]:** Multiply G with a random invertible matrix $\mathsf{T} \in \mathbb{F}^{n\times n}$ from the left, where T preserves the norm, see §4.1. Obviously one can correct errors of norm up to t in $\langle \mathsf{GT} \rangle$, if G and T are known.
3. **Subcode [52]:** Let $0 < l < k$. Multiply G with a random matrix $\mathsf{S} \in \mathbb{F}^{l\times k}$ of full rank from the right. As $\langle \mathsf{SG} \rangle \subseteq \langle \mathsf{G} \rangle$, the known error correction algorithm may be used.
4. **Subfield Subcode [48]:** Take the $\mathbb{F}_{\mathrm{SUB}}$-subfield subcode of the secret code for a subfield $\mathbb{F}_{\mathrm{SUB}}$ of $\mathbb{F}$. As before, one can correct errors by the error correcting algorithm for the secret code. However, sometimes one can correct errors of larger norm in the subfield subcode than in the original code, compare Definition 9 and following.
5. **Matrix Concatenation [70]:** Take the code $\langle [\, \mathsf{G} | \mathsf{SG} \,] \rangle$ for an invertible matrix $\mathsf{S} \in \mathbb{F}^{k\times k}$. In Hamming norm, the secret key holder can correct $2t+1$ errors in this code, as he can correct errors in the first or the second n columns.[2]
6. **Random Redundancy [22]:** Add a number l of random columns at the left side of the matrix G. Errors can be corrected in the last n columns.
7. **Artificial Errors [27]:** One can choose to modify the matrix G at a small number of positions. These positions will be treated as erasures on decryption and thus change the norm t of the errors that can be decoded.
8. **Reducible Codes [26]:** Choose some matrices $\mathsf{Y} \in \mathbb{F}^{k\times n}$ and $\mathsf{S} \in \mathbb{F}^{l\times k}$ with $l \leq k$. Then take the code generated by

$$\left[\begin{array}{c|c} \mathsf{SG} & 0 \\ \hline \mathsf{Y} & \mathsf{G} \end{array}\right].$$

   Error correction by the algorithm for the secret code is possible if one corrects errors in sections, beginning from the right.[2] However, for correcting errors in Hamming metric, this approach does not seem to be suitable [56].

Table 1. Strategies for hiding the structure of a code

[2] One might generalize this approach by replacing one of the matrices G by a second secret code.

be seen as a combination of the strategies 1,2 and 4. Nevertheless, we have to remark, that all strategies have to be used with care, as they can but do not necessarily lead to a secure cryptosystem (compare e.g. [54, 78] and §4.3).

2.2 CFS signature

The only unbroken signature scheme based on the McEliece, or rather on the Niederreiter PKC was presented by Courtois, Finiasz and Sendrier in [14]. The security of the CFS scheme (against universal forgery) can be reduced to the hardness of Problem 1. The knowledge of the private key allows the decoder to solve this problem for a certain fraction of random words $\mathbf{c}$. The idea of the CFS algorithm is to repeatedly hash the document, randomized by a counter of bit-length r, until the output is a decryptable ciphertext. The signer uses his secret key to determine the corresponding error-vector. Together with the current value of the counter, this error vector will then serve as signature. The signature scheme is summarized in Algorithm 2.4.

The average number of attempts needed to reach a decodable syndrome can be estimated by comparing the total number of syndromes to the number of efficiently correctable syndromes:

$$\frac{\sum_{i=0}^{t}\binom{n}{t}}{2^{n-k}} = \frac{\sum_{i=0}^{t}\binom{n}{t}}{2^{mt}} \approx \frac{n^t/t!}{n^t} = \frac{1}{t!}$$

Thus each syndrome has a probability of $\frac{1}{t!}$ to be decodable, which can be tested in about t^2m^3 binary operations, see §6.1. The CFS scheme needs about $t^2m^3t!$ operations to generate a signature [14] and produces signatures of length $\log_2(r\binom{n}{t}) \approx \log_2(n^t)$. Thus, r has to be be larger than $\log_2(t!)$. The signature length $(n+r)$ can be reduced considerably, by employing a mapping like $\phi_{n,t}$.

With the parameters suggested by Courtois, Finiasz and Sendrier ($m = 16, t = 9$) the number of possible error-vectors is approximately given by $\binom{n}{t} = \binom{2^{16}}{9} \approx 2^{125.5}$ so that a 126-bit counter suffices to address each of them. However, these parameters are too low to prevent a generalized collision attack, see §3.4. As the CFS scheme does not scale well with growing parameters, secure instances of the CFS scheme require huge public keys.

2.3 Stern's identification scheme

Stern's identification scheme presented in 1994 is closely related to the Niederreiter cryptosystem. There exists a variant of this scheme by Pascal Véron [76]. However, we will explain the original scheme: Let H^{pub} be a $(n-k)\times n$ matrix common to all users. If H^{pub} is chosen randomly, it will provide a parity check matrix for a code with asymptotically good minimum distance given by the Gilbert-Varshamov (GV) bound, see Definition 1. The private key for a user

Algorithm 2.4 CFS digital signature

- **System parameters:** $m, t \in \mathbb{N}$.
- **Key Generation:** Generate a Niederreiter PKC key pair with a code drawn from the class of $[n = 2^m, k = n - mt, 2t + 1]$ binary irreducible Goppa codes.
- **Signing:**

  Input: h a public hash function, $\phi_{n,t}$, $D_{(S,D_{\mathcal{G}},P)}$, $r \in \mathbb{N}_+$ and the document d to be signed
  Output: A CFS-signature $\mathbf{s}$.

  $\mathbf{z} = h(d)$
  choose a r-bit Vector $\mathbf{i}$ at random
  $\mathbf{s} = h(\mathbf{z}||\mathbf{i})$
  while $\mathbf{s}$ is not decodable **do**
    choose a r-bit Vector $\mathbf{i}$ at random
    $\mathbf{s} = h(\mathbf{z}||\mathbf{i})$
  $\mathbf{e} = D_{(S,D_{\mathcal{G}},P)}(\mathbf{s})$
  $\mathbf{s} = (\phi_{n,t}^{-1}(\mathbf{e})||\mathbf{i})$
- **Verification:**

  Input: A signature $\mathbf{s} = (\phi_{n,t}^{-1}(\mathbf{e})||\mathbf{i})$, the document d and H^{pub}
  Output: accept or **reject**

  $\mathbf{e} = \phi_{n,t}(\phi_{n,t}^{-1}(\mathbf{e}))$
  $\mathbf{s}_1 = \mathsf{H}^{\text{pub}}(\mathbf{e}^\top)$
  $\mathbf{s}_2 = h(h(d)||\mathbf{i})$
  if $\mathbf{s}_1 = \mathbf{s}_2$ **then**
    accept $\mathbf{s}$
  else
    reject $\mathbf{s}$

will thus be a word $\mathbf{e}$ of low weight w (e.g. $w \approx$ GV bound), which sums up to the syndrome $\mathbf{e}\mathsf{H} = \mathbf{s}$, the public key. By Stern's 3-pass zero-knowledge protocol (Algorithm 2.5), the secret key holder can prove his knowledge of $\mathbf{e}$ using two blending factors: a permutation and a random vector. However, a dishonest prover not knowing $\mathbf{e}$ can cheat the verifier in the protocol with probability 2/3. Thus, the protocol has to be run several times to detect cheating provers.

The security of the scheme relies on the difficulty of the general decoding problem, that is on the difficulty of determining the preimage $\mathbf{e}$ of $\mathbf{s} = \mathsf{H}^{\text{pub}}\mathbf{e}^\top$. Without the secret key, an adversary has three alternatives to deceive the verifier:

1. To be able to answer the challenges $b \in \{1, 2\}$, the attacker commits to $c_1 = (\Pi, \mathsf{H}^{\text{pub}}\mathbf{y}^\top + \mathbf{s})$ and selects a random vector $\hat{\mathbf{e}}$ of the same weight as $\mathbf{e}$. Now, he computes $c_2 = (\mathbf{y} + \hat{\mathbf{e}})\Pi$ and $c_3 = \Pi(\mathbf{y})$.
2. He can work with a random $\hat{\mathbf{e}}$ of weight w instead of the secret key while computing c_1, c_2, c_3. He will succeed if he is asked $b \in \{0, 2\}$ but in case

Algorithm 2.5 Stern's identification scheme

- **System parameters :** $n, k, q, w \in \mathbb{N}_+$ and $\mathsf{H}^{\text{pub}} \in \mathbb{F}_q^{(n-k)\times n}$.
- **Public key :** $\mathsf{H}^{\text{pub}}\mathbf{e}^\top = \mathbf{s} \in \mathbb{F}_q^{n-k}$
- **Private key :** $\mathbf{e} \in \mathbb{F}_q^n$ of weight w.

Prover	**Verifier**
Choose random n-bit vector $\mathbf{y}$ and random permutation $\mathsf{\Pi}$, to compute $c_1 = (\mathsf{\Pi}, \mathsf{H}^{\text{pub}}\mathbf{y}^\top)$, $c_2 = \mathbf{y}\mathsf{\Pi}$, $c_3 = (\mathbf{y} + \mathbf{e})\mathsf{\Pi}$. Send commitments for (c_1, c_2, c_3)	
	Send random request $b \in \{0, 1, 2\}$
If $b = 0 \Rightarrow$ reveal $c_2, \mathsf{\Pi}$ If $b = 1 \Rightarrow$ reveal $c_3, \mathsf{\Pi}$ If $b = 2 \Rightarrow$ reveal c_2, c_3	
	If $b = 0 \Rightarrow$ check c_1, c_2 If $b = 1 \Rightarrow$ check c_1, c_3 with $\mathsf{H}^{\text{pub}}\mathbf{y}^\top = \mathsf{H}^{\text{pub}}(\mathbf{y} + \mathbf{e})^\top + \mathbf{s}$ If $b = 2 \Rightarrow$ check c_2, c_3 and the weight of $\mathbf{e}\mathsf{\Pi}$.

$b = 1$ he will not be able to produce the correct c_1, c_3 since $\mathsf{H}^{\text{pub}}\hat{\mathbf{e}}^\top \neq \mathsf{H}^{\text{pub}}\mathbf{e}^\top = \mathbf{s}$.

3. He can choose $\hat{\mathbf{y}}$ of arbitrary weight from the set of all possible preimages of $\mathbf{s}$ and replaces $\mathbf{e}$ by $\hat{\mathbf{y}}$ while computing c_1, c_2, c_3. This time he will fail to answer the request $b = 2$ since $\text{wt}(\hat{\mathbf{y}}) \neq w$.

The communication cost per round is about $n(\log_2(q) + \log_2(n))$ plus three times the size of the employed commitments (e.g. a hash function).

The standard method to convert the identification procedure into a procedure for signing, is to replace verifier-queries by values suitably derived from the commitments and the message to be signed. This leads to a blow-up of each (hashed) plaintext bit to more than $(n[\log_2(q) + \log_2(n)])/\log_2(3)$ signature bits and is therefore of theoretical interest as a signature. However, the security of the resulting signature scheme can be reduced to the average-case hardness of the $\mathcal{NP}$-hard general decoding problem in the random oracle model.

2.4 Cryptosystems based on the syndrome one-way function

Besides the classical code based PKCs there exist other cryptographic primitives with security reductions to coding theoretic problems. For symmetric cryptosystems we do not need a trapdoor and can take the computation of a syndrome of a random code as a one-way function. In this section we want to give a way of obtaining cryptographic strong hashing and generation of pseudorandom sequences using coding theoretic primitives.

Code based hashing

If in Stern's identification scheme parameters are chosen properly, one has the following inequality:

$$\binom{n}{w}(q-1)^{w-1} \cdot q^{k-n} \geq 1.$$

Thus, there are more vectors of weight w and length n than syndromes of an $[n, k]$ code. If it is still hard to recover vectors of weight w in the set of vectors with a certain syndrome, then, computing syndromes can serve as a compression function. Based on this compression function, a hash function can be constructed [3]. The compression function is realized by $\mathbf{x} \mapsto \phi_{n,w}(\mathbf{x})\mathsf{H}$, with $\phi_{n,w}$ given in Algorithm 2.2. In Figure 1 we give an intuition of the way the hash function works.

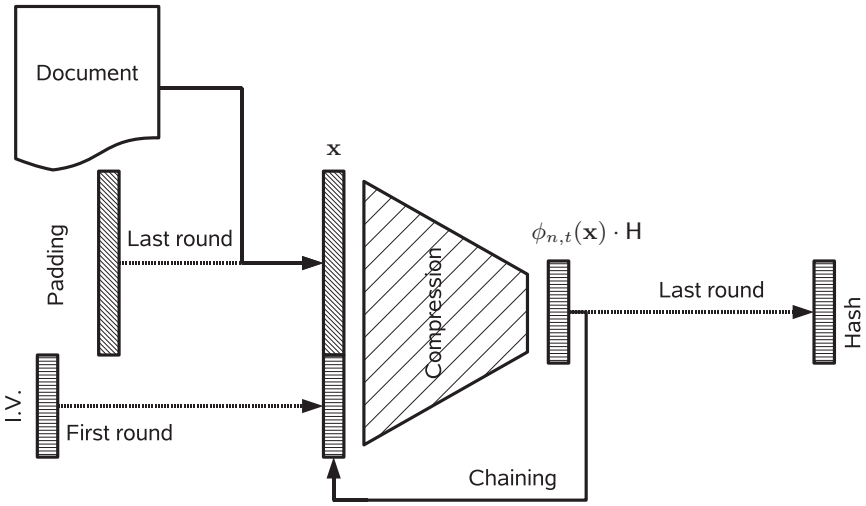

Fig. 1. Merkle-Damgård scheme of hash functions

The performance of such a hash function depends on the time needed to compute the one-to-one mapping $\phi_{n,w}$. In order to speed-up such hash function, one can for example limit the set $\mathcal{W}_{n,t}$ to the set of w'*-regular words*

$$\mathcal{W}'_{n,t} = \left\{(\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_{w/w'}) \in \mathbb{F}_q^n \mid \mathbf{e}_i \in \mathbb{F}_q^{n \cdot w'/w}, \mathrm{wt}(\mathbf{e}) = w'\right\}$$

if $w'|w$ and $(w'/w)|n$. The modified mapping $\phi_{n,w}^{w'}$ is easy to compute, if $w' = 1$. The resulting compression function is $\mathbf{x} \mapsto \phi_{n,w}^{w'}(\mathbf{x})\mathsf{H}$. Nevertheless, using regular words changes the problem of inverting the compression function. Even if it was proved, that inverting $\phi_{n,t}^{w'}(\mathbf{x})\mathsf{H}$ is $\mathcal{NP}$-hard in general, there

is no evidence if it is weak or hard in the average case [3]. Further, it was only proved, that finding preimages for $\phi_{n,t}^{w'}(\mathbf{x})\mathsf{H}$ is $\mathcal{NP}$-hard in the cases $w' \in \{1, 2\}$, but not the problem to find collisions.

For the chaining step of a hash function one possibility is obviously to concatenate the syndrome obtained with the input of the next round and to apply $\phi_{n,w}$ afterwards. In the case of w'-regular words with blocklengths $n \cdot w'/w$, there exists a second possibility: One can simply concatenate two such words of length $< n$ to obtain a new w'-regular word of length n and weight w.

One possible choice is to use $q = 2, w' = 1$ with parameters $n = 2^{14}$, $n - k = 160$ and $w = 64$ for a moderately ($2^{62.2}$) secure hash function and $n = 3 \cdot 2^{13}$, $n - k = 224$ and $w = 96$ for a ($2^{82.3}$) secure version. For more parameter proposals and comparison with other hash functions we refer to [3]. An attack against the collision resistance of the hash function is presented in §3.4.

Cryptographically strong random numbers

If in Stern's identification scheme parameters are chosen such that

$$\binom{n}{w}(q-1)^{w-1} \cdot q^{k-n} \leq 1,$$

there are less vectors of weight w and length n than syndromes of an $[n, k]$ code. If it is still hard to recover vectors of weight w in the set of vectors with a certain syndrome, then, computing syndromes can serve as a expansion function and thus to generate pseudorandom sequences [19]. Figure 2 gives an intuition of the way the pseudo random number generator (PRNG) works. For security reasons, we propose to use the same parameters as in Stern's

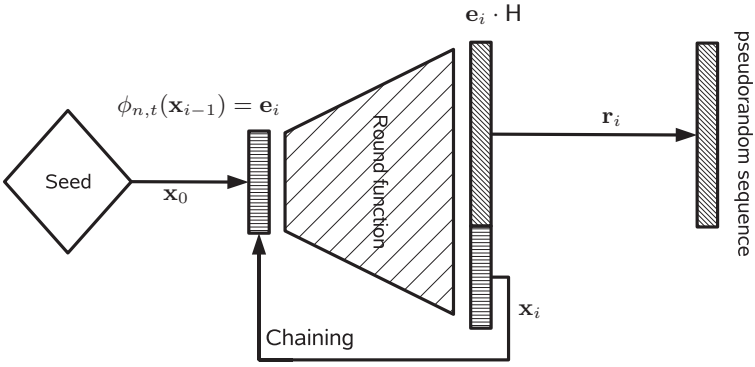

Fig. 2. Scheme of code based PRNG

identification scheme, see §2.3. Here again, w'-regular words can be used to speed-up the PRNG.

3 The security of computing syndromes as one-way function

In this section we consider the message security (opposed to key security) of code-based cryptosystems. We assume the attacker has no information on the algebraic structure of the underlying error correcting code, either because the trapdoor is sufficiently well hidden (public key systems) or because there is no trapdoor (code-based one way functions). This means correcting errors in a linear code for which one knows only a generator (or a parity check) matrix.

Unless specified otherwise, the codes we consider in this section have a binary alphabet. It is sufficient for most cryptosystems of interest. Moreover, most statements can be generalized to a larger alphabet.

3.1 Preliminaries

We consider a binary linear code $\mathcal{C}$ of length n and dimension k. We denote $r = n - k$ the codimension of $\mathcal{C}$ and H a parity check matrix of $\mathcal{C}$. We define a syndrome mapping relatively to H

$$\begin{array}{rccc} S_{\mathsf{H}} : & \{0,1\}^n & \longrightarrow & \{0,1\}^r \\ & \mathbf{y} & \longmapsto & \mathbf{y}\mathsf{H}^\top \end{array}$$

For any $\mathbf{s} \in \{0,1\}^r$, we denote the set of words of $\{0,1\}^n$ with syndrome $\mathbf{s}$ by

$$S_{\mathsf{H}}^{-1}(\mathbf{s}) = \left\{\mathbf{y} \in \{0,1\}^n \mid \mathbf{y}\mathsf{H}^\top = \mathbf{s}\right\}.$$

By definition, we have $S_{\mathsf{H}}^{-1}(\mathbf{0}) = \mathcal{C}$ for any parity check matrix H of $\mathcal{C}$. The sets $\mathbf{y} + \mathcal{C}$, for all $\mathbf{y}$ in $\{0,1\}^n$, are called the *cosets* of $\mathcal{C}$. There are exactly 2^r different cosets which form a partition of $\{0,1\}^n$ (*i.e.* pairwise disjoint). For any parity check matrix H of $\mathcal{C}$, there is a one to one correspondence between cosets and syndromes relatively to H.

Proposition 1. *For any syndrome* $\mathbf{s} \in \{0,1\}^r$ *we have*

$$S_{\mathsf{H}}^{-1}(\mathbf{s}) = \mathbf{y} + \mathcal{C} = \{\mathbf{y} + \mathbf{x} \mid \mathbf{x} \in \mathcal{C}\},$$

where $\mathbf{y}$ *is any word of* $\{0,1\}^n$ *of syndrome* $\mathbf{s}$*. Moreover, finding such a word* $\mathbf{y}$ *from* $\mathbf{s}$ *(and* H*) can be achieved in polynomial time.*

For any $\mathbf{y}$ and $\mathbf{z}$ in $S_{\mathsf{H}}^{-1}(\mathbf{s})$, we have $\mathbf{y}\mathsf{H}^\top = \mathbf{z}\mathsf{H}^\top$, thus $(\mathbf{y} + \mathbf{z})\mathsf{H}^\top = 0$ and $\mathbf{y} + \mathbf{z} \in \mathcal{C}$. It follows that $S_{\mathsf{H}}^{-1}(\mathbf{s}) = \mathbf{y} + \mathcal{C}$.

To compute one particular element of $S_{\mathsf{H}}^{-1}(\mathbf{s})$, given $\mathbf{s}$, we will consider a systematic form H_0 of the parity check matrix H. That is a $r \times n$ binary matrix

H_0 of the form $[\mathsf{Id} \mid \mathsf{X}]$ (where Id is the $r \times r$ identity matrix and X is some $r \times k$ matrix) such that $\mathsf{H}_0 = \mathsf{UH}$, with U a $r \times r$ non-singular matrix. One can obtain such a matrix U in time $\mathcal{O}(r^3)$ by inverting the first r columns[3] of H. Let $\mathbf{y} = [\mathbf{s}\mathsf{U}^\top \mid \mathbf{0}] \in \{0,1\}^n$, since $(\mathsf{U}^\top)^{-1} = (\mathsf{U}^{-1})^\top$, we have

$$\mathbf{y}\mathsf{H}^\top = \mathbf{y}(\mathsf{U}^{-1}\mathsf{H}_0)^\top = \mathbf{y}\mathsf{H}_0^\top(\mathsf{U}^{-1})^\top = (\mathbf{s}\mathsf{U}^\top \mid \mathbf{0}) \begin{bmatrix} \mathsf{Id} \\ \mathsf{X}^\top \end{bmatrix} (\mathsf{U}^\top)^{-1} = \mathbf{s}.$$

The word $\mathbf{y}$ is in $S_{\mathsf{H}}^{-1}(\mathbf{s})$ and is obtained in polynomial time.

3.2 Decoding problems

Let $\mathcal{C}$ be a binary linear code of parity check matrix H. We are given a word $\mathbf{y} \in \{0,1\}^n$ and its syndrome $\mathbf{s} = \mathbf{y}\mathsf{H}^\top \in \{0,1\}^r$. Decoding consists of solving one of the following equivalent problems:

(i) Find a codeword $\mathbf{x} \in \mathcal{C}$ closest to $\mathbf{y}$ for the Hamming distance.
(ii) Find an error $\mathbf{e} \in \mathbf{y} + \mathcal{C}$ of minimal Hamming weight.
(iii) Find an error $\mathbf{e} \in S_{\mathsf{H}}^{-1}(\mathbf{s})$ of minimal Hamming weight.

In practice, given an instance of a decoding problem, it is difficult to check if the error $\mathbf{e}$ is really of *minimal* weight in the coset (or if the codeword $\mathbf{x}$ is really the *closest* to $\mathbf{y}$). Because of that, the decoding problem as stated above is not in $\mathcal{NP}$. Instead, we will consider a slightly different abstraction of the problem, called *syndrome decoding*:

Problem 2 (Computational Syndrome Decoding). Given a binary $r \times n$ matrix H, a word $\mathbf{s}$ in $\{0,1\}^r$ and an integer $w > 0$, find a word $\mathbf{e}$ in $S_{\mathsf{H}}^{-1}(\mathbf{s})$ of Hamming weight $\leq w$.

The value of the additional parameter w will significantly affect the difficulty (see §3.3) of the resolution. In the theory of error correcting codes the problem is meaningful only if w is such that the problem has a single solution with high probability (*i.e.* w is not greater than the Gilbert-Varshamov bound (Definition 1)). For cryptographic applications, any value of w such that the problem is hard may produce a one-way function.

Decades of practice indicate that syndrome decoding in an arbitrary linear code is difficult (see [4] for instance). In addition, the associated decision problem was proved $\mathcal{NP}$-complete in [6].

We will denote $\mathrm{CSD}(\mathsf{H}, w, \mathbf{s})$ a specific instance of the computational syndrome decoding problem. Note that there is no "gap" (as for problems related to Diffie-Hellman) between the decisional and the computational problems. In fact an attacker can solve any instance of CSD with a linear number of access to a decisional syndrome decoding oracle (this is the basis for the reaction attack, see §5.3).

[3] w.l.o.g. we can assume that the first r columns of H are full rank

The problem of finding non-zero words of small Hamming weight (say $\leq w$) in a given linear code is very similar, but not identical, to decoding. We can state it as follows

Problem 3 (Codeword Finding). Given a binary $r \times n$ matrix H and an integer $w > 0$, find a non-zero word of Hamming weight $\leq w$ in $S_{\mathsf{H}}^{-1}(\mathbf{0})$.

Though it looks similar, this is not a particular instance of CSD, because of the non-zero condition. In fact, if $\mathcal{C}$ is the linear code of parity check matrix H, then any solution of $\mathrm{CSD}(\mathsf{H}, w, \mathbf{y}\mathsf{H}^\top)$ is also a solution to $\mathrm{CF}(\mathsf{H}', w)$, where H' is a parity check matrix of the code $\mathcal{C}' = \langle \mathbf{y} + \mathcal{C} \rangle$ spanned by $\mathbf{y}$ and $\mathcal{C}$. The converse is true only if $w < \mathrm{dmin}(\mathcal{C})$, the minimum distance of $\mathcal{C}$.

The minimum distance is usually unknown. However most binary linear codes of length n and codimension r have a minimum distance very close to the Gilbert-Varshamov distance $d_0(n, r)$.

Definition 1. *[4] The Gilbert-Varshamov distance $d_0(n, r)$ (or simply d_0 when there is no ambiguity) is defined as the largest integer such that*

$$\sum_{i=0}^{d_0-1} \binom{n}{i} \leq 2^r.$$

Let H, $\mathbf{y}$ and H' be defined as above. Let $\mathbf{e}$ be a solution to $\mathrm{CF}(\mathsf{H}', w)$, independently of w we have

- if $\mathrm{wt}(\mathbf{e}) < d_0$, then $\mathbf{e}$ is very likely a solution to $\mathrm{CSD}(\mathsf{H}, w, \mathbf{y})$,
- if $\mathrm{wt}(\mathbf{e}) \geq d_0$, then $\mathbf{e}$ is a solution to $\mathrm{CSD}(\mathsf{H}, w, \mathbf{y})$ with probability $\approx 1/2$.

Those informal statements hold "in average" and come from the fact that $\mathcal{C}'$ the code spanned by $\mathbf{y}$ and $\mathcal{C}$ is equal to $\mathcal{C} \cup (\mathbf{y} + \mathcal{C})$. If the weight of $\mathbf{e} \in \mathcal{C}' = \mathcal{C} \cup (\mathbf{y} + \mathcal{C})$ is smaller than the minimum distance of $\mathcal{C}$ (which is likely to be close to d_0), then it belongs to the coset $\mathbf{y}+\mathcal{C}$. On the other hand if the weight of $\mathbf{e}$ is higher than d_0, then, if it is a random solution to $\mathrm{CF}(\mathsf{H}', w)$, it is equally likely[4] to be in $\mathcal{C}$ and in $\mathbf{y} + \mathcal{C}$. In practice, most general purpose decoders, and in particular those used in cryptanalysis, are in fact searching for small weight codewords.

In the problems we have stated so far, the target weight w is an input. In many cases of interest, the target weight will instead depend of the code parameters (length and dimension). This will happen in particular in two cases that we detail below: *Complete Decoding* and *Bounded Decoding*.

As we have seen earlier, decoding will consist in finding a word of minimal weight that produces a given syndrome. If the syndrome is random, then the solution is very likely to have a weight equal to the Gilbert-Varshamov distance. Decoding is thus likely to be as hard as the following problem.

[4] Metric properties of a random code are in practice indistinguishable from those of a random set with the same cardinality

Problem 4 (Complete Decoding). Given a binary $r \times n$ matrix H and a word $\mathbf{s}$ in $\{0,1\}^r$, find a word of Hamming weight $\leq d_0(n,r)$ in $S_{\mathsf{H}}^{-1}(\mathbf{s})$.

This is in fact the most general and the most difficult computational problem for given parameters n and r.

In a public key encryption scheme, like McEliece or Niederreiter, the target weight is much smaller as it will be equal to the error correcting capability of the underlying code. A Goppa code of length $n = 2^m$ and correcting t errors has codimension $r = tm$. A message attack on the McEliece encryption scheme will thus correspond to the following computational problem

Problem 5 (Goppa Bounded Decoding). Given a binary $r \times n$ matrix H and a word $\mathbf{s}$ in $\{0,1\}^r$, find a word of Hamming weight $\leq r/\log_2 n$ in $S_{\mathsf{H}}^{-1}(\mathbf{s})$.

The associated decision problem is $\mathcal{NP}$-complete [18]. This demonstrates that the above computational problem is $\mathcal{NP}$-hard, that is difficult in the worst case. Even though this doesn't say anything on the average case complexity, at least this proves that if we reduce the target weight to the error correcting capability of a Goppa code, we do not fall into an easy case.

3.3 Decoding algorithms

Information set[5] decoding is undoubtedly the technique that has attracted most of the cryptographer's attention. The best known decoding attacks on McEliece and Niederreiter are all derived from it. There have been other attempts but with a mitigated success (iterative decoding [20] or statistical decoding [34, 55]).

Algorithm 3.1 presents a generalized version of information set decoding. Lee and Brickell [39] were the first to use it to analyze the security of

---

Algorithm 3.1 Information set decoding (for parameter p)

- **Input**: a $k \times n$ matrix G, an integer w
- **Output**: a non-zero codeword of weight $\leq w$
- **Repeat**
  - Pick a $n \times n$ permutation matrix P.
  - Compute $\mathsf{G}' = \mathsf{UGP} = (\mathsf{Id} \mid \mathsf{R})$ (w.l.o.g. we assume the first k positions form an information set).
  - Compute all the sum of p rows or less of G', if one of those sums has weight $\leq w$ then stop and return it.

McEliece's PKC. In another context, computing the minimum distance of a

[5] An information set for a given code of dimension k, is a set of k positions such that the restriction of the code to those positions contains all the k-tuples exactly once. In particular, it means that the corresponding columns in any generator matrix are independent.

code, Leon [40] proposed an improvement by looking for codewords containing zeroes in a windows of size ℓ in the redundancy (right) part of the codeword. It was further optimized by Stern [72] by dividing the information set in two parts, allowing to speed-up the search for codewords with zeroes in the window by a birthday attack technique.

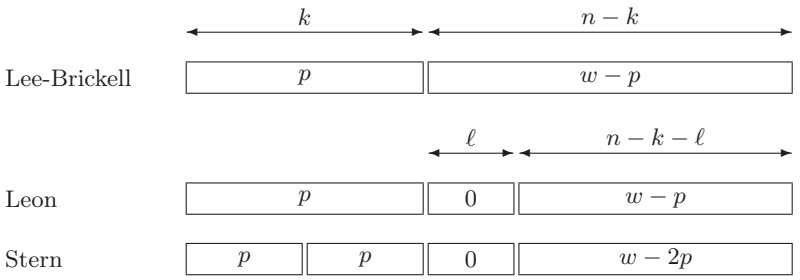

Fig. 3. Weight profile of the codewords sought by the various algorithms (the number inside the boxes is the Hamming weight of the corresponding tuples)

In Figure 3 we present the different weight profiles corresponding to a success, the probability of success of a given iteration is respectively

$$P_{LB} = \frac{\binom{k}{p}\binom{n-k}{w-p}}{\binom{n}{w}}, P_L = \frac{\binom{k}{p}\binom{n-k-\ell}{w-p}}{\binom{n}{w}}, P_S = \frac{\binom{k/2}{p}^2\binom{n-k-\ell}{w-2p}}{\binom{n}{w}}.$$

The total cost of the algorithm is usually expressed as a *binary work factor*. It is equal to the cost (in binary operation) of an iteration divided by the above probability (*i.e.* multiplied by the expected number of iterations).

The Canteaut-Chabaud decoding algorithm

The best known variant was proposed by Canteaut and Chabaud [12] and is the Stern algorithm with another improvement due to van Tilburg [75] consisting in changing only one element of the information set at each iteration. The overall binary work factor is smaller, but it is much more difficult to evaluate as for every value of the parameters p and ℓ, the probability of success is obtained by computing the stationary distribution of a Markov process. It is nevertheless possible to exhibit a rather tight lower bound on its complexity. The probability of success of an iteration is upper bounded by the success probability P_S of Stern's algorithm and for any p the best value for ℓ is close to $\log_2 \binom{k/2}{p}$. Finally, we get the following lower bound on the binary work factor for Canteaut-Chabaud algorithm:

$$WF(n,k,w) \geq \min_p \left(\frac{K\ell}{2^\ell} \frac{\binom{n}{w}}{\binom{n-k-\ell}{w-2p}} \right) \text{ where } \ell = \log_2 \binom{k/2}{p}. \qquad (2)$$

In the above formula, K is a small constant (see the remark below) which also appears in the cost of Canteaut-Chabaud's algorithm. The space complexity in bits is lower bounded by $\ell 2^\ell$. The lower bound defined by (2) is close in practice (a factor 10 at most, see Figures 5 and 6) to the estimation given in [12] which requires the computation of the fundamental matrix of a Markov chain (inversion of a real matrix of size $t + p + 1$) for every value of p and ℓ.

Remark 1. The binary work factor gives a measure of the cost of the algorithm. It is in fact a lower bound on the average number of binary operations needed to solve a problem of given size. Dividing by 32 or 64 (minus 5 or 6 on the exponent) will give a lower bound on the number of CPU operations.

The actual computation time will depend on the relative cost of the various operations involved (sorting, storing, fetching, xoring, popcounting, ...) for a particular implementation and a particular platform. In formula (2), all of this is hidden in the constant K (for practical purposes we took $K = 3$).

Let us consider now how the work factor evolves with the error weight w. For fixed values of the code length n and dimension k, the maximal cost is obtained when w is equal to the Gilbert-Varshamov distance. When $w \leq d_0(n, n-k)$, the decoding cost is $2^{w(c+o(1))}$, where the constant c depends of the ratio k/n. Typical behavior for fixed n and k when t grows is given in Figure 4.

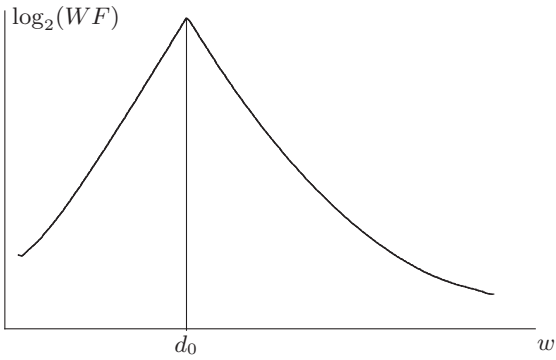

Fig. 4. Information set decoding running time (log scale) for fixed length and dimension when the error weight w varies

When w gets larger, the number of solutions grows very quickly. Formula (2) gives the cost for finding one specific codeword of weight w, the decoding cost is obtained by dividing this value by the expected number of solutions. For those values of w, information set decoding is not always the best technique, and other algorithms, like the generalized birthday attack [10, 77], may be more efficient (see §3.4).

Decoding attacks against McEliece

We consider binary Goppa codes. The length is $n = 2^m$ and the dimension is related with the error weight t, as $k = n - tm$.

In practice, the best value for parameter p in formula (2) is small. For length 2048 it is always equal to 2, and for length 4096, the best value of p varies between 2 and 5. Figures 5 and 6 (see also Table 2) give an estimate of the practical security of the McEliece cryptosystem when binary Goppa codes of length 2048 and 4096 are used. Note finally that the decoding (message) attacks are always more efficient than the structural (key) attacks (see §4.3).

3.4 Collision attacks against FSB and CFS

The fastest attacks on the CFS scheme and the FSB hash function are based on Wagner's solution for the generalized Birthday paradox. Wagner's main theorem can be seen as a generalization of the search part of Stern's algorithm for low weight code words or the algorithm of Patarin and Camion [10] and can be summarized as follows:

Theorem 1. (Generalized Birthday Problem) *Let $r, a \in \mathbb{N}$ with $(a+1)|r$ and $\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_{2^a} \subseteq \mathbb{F}_{2^r}$ be sets of cardinality $2^{\frac{r}{a+1}}$, then, a solution of the equation*

$$\sum_{i=1}^{2^a} x_i = \mathbf{0} \text{ where } x_i \in \mathcal{L}_i, \tag{3}$$

can be found in $\mathcal{O}(2^a 2^{\frac{r}{a+1}})$ Operations (over $\mathbb{F}_{2^r}$).

The algorithm proposed by Wagner is iterative: First, one searches for partial collisions of the sets $\mathcal{L}_i$ and $\mathcal{L}_{i+2^{a-1}}$, $i = 1, \cdots, 2^{a-1}$, that is, such pairs $(x_i, x_{i+2^{a-1}})$ that $LSB_{\frac{r}{a+1}}(x_i + x_{i+2^{a-1}}) = 0$. This way, one obtains 2^{a-1} Lists with approximately $2^{\frac{r}{a+1}}$ pairs, where the last $\frac{r}{a+1}$ entries are zero and can be omitted in the next step. A recursive application of this step leads to a solution of Equation (3).

As shown by J.-S. Coron and A. Joux, Wagner's solution for the generalized Birthday Paradox can be used to find collisions for the FSB hash. This is due to the fact, that the compression function of the FSB hash is inherently different to the one used by other hash functions: If we consider $\phi'_{n,t}(\mathbf{x}) \cdot \mathsf{H}$ (or $\phi_{n,t}(\mathbf{x}) \cdot \mathsf{H}$), we can see, that one collision ($\phi'_{n,t}(\mathbf{x}_1) \cdot \mathsf{H} = \phi'_{n,t}(\mathbf{x}_2) \cdot \mathsf{H}$) leads to up to $\binom{w}{w/2}$ further collisions. As the mapping $\phi_{n,t}$ can be easily inverted and the second part of the compression function is linear, we can apply Wagner's theorem employing a "divide and conquer"-strategy.

Applied to the FSB hash function we obtain the following attack against the collision resistance in the case of $\phi'_{n,t}$ as compression function: Each list $\mathcal{L}_1, \cdots, \mathcal{L}_{2^a}$ is designed to contain the syndromes of 2-quasiregular words $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_w)$, such that for all $i \neq j$ and γ:

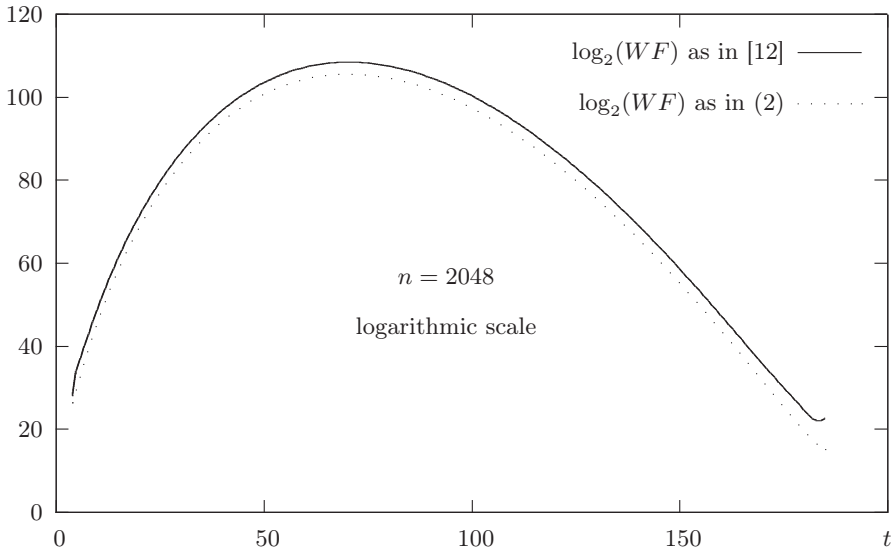

Fig. 5. Binary workfactor ($\log_2$) for finding words of weight t in a binary code of length 2048 and dimension $2048 - 11t$ (Goppa codes parameters)

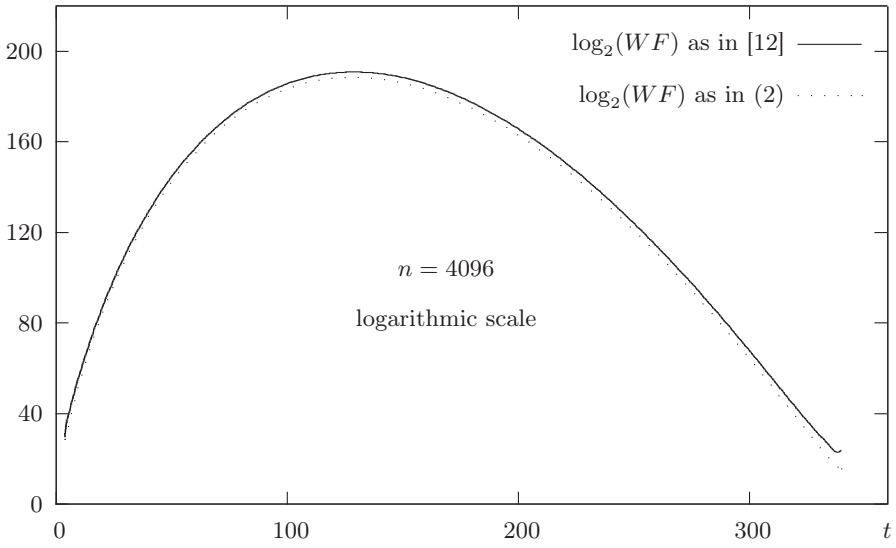

Fig. 6. Binary workfactor ($\log_2$) for finding words of weight t in a binary code of length 4096 and dimension $4096 - 12t$ (Goppa codes parameters)

$$(\exists_{\phi'_{n,t}(\mathbf{e})\mathsf{H}\in\mathcal{L}_i} : \mathbf{e}_\gamma \neq 0) \Rightarrow (\forall_{\phi'_{n,t}(\mathbf{e})\mathsf{H}\in\mathcal{L}_j} : \mathbf{e}_\gamma = 0).$$

If the lists are not of the desired cardinality $2^{\frac{r}{a+1}}$ ($r = n - k$ is the hash width), we can modify the attack accordingly, (compare [3]). We omit details and conclude that the size of the lists implies the following restriction to the attacker:

$$\frac{2^a}{a+1} \leq \frac{w}{r} \log_2 \left(\frac{n}{w}\right).$$

Therefore, the authors of [3] conclude that the work factor for an attacker grows exponentially with $n - k$ if we choose two constants $\alpha, \beta \in \mathbb{R}$ and then compute $(n, w) = (\alpha(n - k), \beta(n - k))$ as then a is upper bounded by a constant.

Likewise, Wagner's algorithm can be used to generate a valid signature in the CFS scheme (existential forgery): The attacker generates four lists: One with possible hash values, and the remaining three as syndromes of weight $t/3$ vectors. The dominating term for the cost of the attack is $2^{mt/3}$. For the $m = 16, t = 9$ CFS parameter set, this leads to an attack that can be performed in about 2^{59} operations.[6]

3.5 The impact of quantum computers

To our knowledge there is no connection between coding theory and the "Hidden Subgroup Problem" as in the case of number theoretic cryptosystems. However, there is still the possibility to employ Grover's algorithm to speed up searching the secret key or the space of possible plaintexts. In this chapter we give an intuition, why Grover's algorithm is not able give a significant speed-up for the existing attacks on code based cryptosystem.

In the following we make the simplifying assumption that by Grover's algorithm we are able to search a set of size N in $\mathcal{O}(\sqrt{N})$ operations on a quantum computer with at least $\log_2(N)$ QuBits. However, a consecutive call of Grover's algorithm is not possible, i.e. if the set to be searched is defined by the output of Grover's algorithm, we can not search this space with Grover's algorithm before writing it in complete to the (classic) memory, see Section 5 of Chapter 1 "Quantum computing".

Solving the generalized birthday problem

The iterative step of Wagner's algorithm can be realized by sorting algorithms, which can not be sped-up with quantum computers so far: Instead of searching $\mathcal{L}_i \times \mathcal{L}_{i+2^{a-1}}$ for all pairs $(x_i, x_{i+2^{a-1}})$ that $LSB_{\frac{r}{a+1}}(x_i + x_{i+2^{a-1}}) = 0$ we can sort the lists $\mathcal{L}_i$ after $LSB_{\frac{r}{a+1}}(x_i)$ and $\mathcal{L}_{i+2^{a-1}}$ after $LSB_{\frac{r}{a+1}}(x_{i+2^{a-1}}) = 0$. The merged list of pairs can now be directly read from the sorted lists (the

[6] Although we do not have a reference, we attribute this attack to Bleichenbacher

halves of the pairs are sorted into the same positions/boxes). If both lists have the same size $\sqrt{N}$, this means that the merging can be done in $\sqrt{N}$ operations instead of N, which is the same speed-up that can be achieved by Grover's algorithm. Thus, even with a quantum computer we can not expect to get attacks for FSB or CFS more efficient than the existing ones.

Algorithms for searching low weight codewords

The crucial point of algorithms for finding low weight codewords is to guess part of the structure at the beginning and then search for the vector in the remaining space. This can be seen as a "divide-and-conquer" strategy. However, this particular strategy of the attacks prevents an effective use of Grover's algorithm - or to be more precise - achieves the same speed-up as Grover's algorithm would achieve: The search step in the algorithms for finding low weight codewords is realized in the same way as in Wagner's algorithm for the generalized birthday paradox. Thus there is no possibility to significantly speed-up the search step by Grover's algorithm.

One might argue, that the guessing phase can be seen as a search phase, too. However, as mentioned before, this would either require an iterative application of Grover's algorithm (which is not possible) or a memory of size of the whole search space, as the search function in the second step depends on the first step. This would clearly ruin the "divide-and-conquer" strategy and is thus not possible either.

Table 2 gives an overview for the advantage of quantum computers over classical computers in attacking the McEliece PKC. One can see, that the

McEliece parameters m, t	Workload Cryptanalysis (in binary operations) classic computer	quantum computer	Minimal number of Qubits	Quantum-computer bit security[7]
$11, 32$	2^{91}	2^{86}	25	80
$11, 40$	2^{98}	2^{94}	50	88
$12, 22$	2^{93}	2^{87}	29	80
$12, 45$	2^{140}	2^{133}	28	128

[7] Compare remark 1

Table 2. Attacking the McEliece PKC

expected advantage does not lead to significantly different security estimations for the McEliece PKC.

4 Codes and structures

In this section we will consider structural attacks, i.e. attacks on the private key of code based PKCs. All codes with an efficient error correction algorithm have either an algebraic structure or are specially designed. For most codes, the knowledge of the canonical generator matrix allows efficient error correction. This is true for all codes one could consider for cryptographic applications (i.e. the ones of large dimension):

- Goppa/alternant codes [48]
- GRS codes [52]
- Gabidulin codes [27]
- Reed-Muller codes [70]
- Algebraic geometric codes [35]
- BCH codes [28]
- Graph based codes (LDPC-, expander-, LT- or turbo-codes)

While graph based codes almost immediately reveal their structure because of their sparse check matrix, this is not obvious for the algebraic codes. In this chapter we thus view how algebraic structures or permutations of a code can be recovered by an attacker.

4.1 Code equivalence

In code-based public key cryptography, one may try to hide a secret code $\mathcal{C}$ by applying an isometry f to it and publish a basis of the code $\mathcal{C}' = f(\mathcal{C})$. If the isometry f is known, a decoder for $\mathcal{C}'$ can be obtained. Hopefully, the isometry will scramble the code structure, making the decoding intractable. In the binary case (the most common) the isometry is "just" a permutation of the support.

An *isometry* of a metric space is a mapping which preserves the distance. Thus, codes that are images of one another by an isometry share all their metric properties and will be functionally equivalent. When the metric space is a vector space, we define the *semi-linear* isometries as those which preserve vector subspaces (i.e. the image of any vector subspace is another vector subspace). The semi-linear isometries of the Hamming space $\mathbb{F}_q^n$ are of the form

$$\begin{array}{rccc} \Psi_{V,\pi,\sigma}: & \mathbb{F}_q^n & \to & \mathbb{F}_q^n \\ & (x_i)_{i\in\mathcal{I}} & \mapsto & (v_i \pi(x_{\sigma^{-1}(i)}))_{i\in\mathcal{I}} \end{array} \tag{4}$$

where $V = (v_i)_{i\in\mathcal{I}}$ is a sequence of non zero elements of $\mathbb{F}_q$, π is a field automorphism of $\mathbb{F}_q$ and σ a permutation of the code support $\mathcal{I}$ (unless otherwise specified, we will now consider codes of length n and support $\mathcal{I}$).

Note that if $\mathcal{C}$ and $\mathcal{C}'$ are linear codes over $\mathbb{F}_q$ with $\mathcal{C}' = f(\mathcal{C})$ for some isometry f of the Hamming space $\mathbb{F}_q^n$, then there exists a semi-linear isometry g such that $\mathcal{C}' = g(\mathcal{C})$ (except in the degenerate case where $\mathcal{C}$ is decomposable,

that is the direct sum of two codes with disjoint support, see [51]). So as long as we only consider linear codes there is no loss of generality if we restrict ourselves to semi-linear isometries.

In the binary case ($q = 2$) semi-linear isometries are reduced to the support permutations (the v_i are all equal to 1 and the only field automorphism is the identity).

Definition 2. *Two linear codes $\mathcal{C}$ and $\mathcal{C}'$ are* equivalent *if one is the image of the other by a semi-linear isometry.*

Definition 3. *Two linear codes $\mathcal{C}$ and $\mathcal{C}'$ are* permutation-equivalent *if there exists a permutation σ such that*

$$\mathcal{C}' = \sigma(\mathcal{C}) = \{(x_{\sigma^{-1}(i)})_{i\in\mathcal{I}} \mid (x_i)_{i\in\mathcal{I}} \in \mathcal{C}\}.$$

The two definitions coincide in the binary case. Note also that the use of σ^{-1} in the index is consistent as we have $\pi(\sigma(\mathcal{C})) = \pi \circ \sigma(\mathcal{C})$.

Code equivalence relates with the ability of a code to correct errors. Two equivalent codes will have the same correcting capability.

Let $\mathcal{C}$ be a code equipped with a t-error correcting decoder $D_{\mathcal{C}}$. For any isometry f, the mapping $f \circ D_{\mathcal{C}} \circ f^{-1}$ is a t-error correcting decoder for $\mathcal{C}' = f(\mathcal{C})$.

4.2 The support splitting algorithm

The support splitting algorithm aims at solving the Code Equivalence problem:

Problem 6 (Code Equivalence).

Instance: Two matrices G_1 and G_2 defined over a finite field.
Question: Are the linear codes $\mathcal{C}_1$ and $\mathcal{C}_2$ spanned respectively by the rows of G_1 and G_2 permutation-equivalent?

This problem was introduced by Petrank and Roth [59], who proved that it was harder than the graph isomorphism problem but not $\mathcal{NP}$-complete unless $\mathcal{P} = \mathcal{NP}$.

Invariants and signatures

Let $\mathcal{L}_n$ denote the set of all linear codes of length n, and let $\mathcal{L} = \bigcup_{n>0} \mathcal{L}_n$ be the set of all linear codes.

Definition 4. *An* invariant *over a set E is defined to be a mapping $\mathcal{L} \to E$ such that any two permutation-equivalent codes take the same value.*

For instance the length, the cardinality or the minimum Hamming weight are invariants over the integers. The weight enumerator polynomial is an invariant over the polynomials with integer coefficients.

Applying an invariant, for instance the weight enumerator, may help us to decide whether two codes are equivalent or not. Two codes with different weight enumerators cannot be equivalent. Unfortunately we may have inequivalent codes with the same weight enumerator, though this only occurs with a small probability.

Any invariant is a global property of a code. We need to define a local property, that is a property of a code and of one of its positions.

Definition 5. *A* signature S *over a set* E *maps a code* $\mathcal{C}$ *of length* n *and an element* $i \in \mathcal{I}$ *into an element of* E *and is such that for all permutations* σ *on* $\mathcal{I}$, $S(\mathcal{C}, i) = S(\sigma(\mathcal{C}), \sigma(i))$.

A signature can be obtained, for instance, by applying an invariant on punctured codes. To an invariant V, we associate the signature $S_V : (\mathcal{C}, i) \mapsto V(\mathcal{C}_{\cdot\mathcal{I}\setminus\{i\}})$ ($\mathcal{C}_{\cdot\mathcal{J}}$ denotes the code restricted to $\mathcal{J} \subset \mathcal{I}$).

Now, if we have a signature S, and wish to answer the question: *"Are* $\mathcal{C}$ *and* $\mathcal{C}'$ *permutation-equivalent?"*, we can compute the sets $S(\mathcal{C}, \mathcal{I}) = \{S(\mathcal{C}, i), i \in \mathcal{I}\}$ and $S(\mathcal{C}', \mathcal{I}) = \{S(\mathcal{C}', i), i \in \mathcal{I}\}$. If $\mathcal{C}$ and $\mathcal{C}'$ are permutation-equivalent, then those sets must be equal (and for every signature value obtained more than once, the multiplicity must be the same). Moreover, for each distinct value in the sets $S(\mathcal{C}, \mathcal{I})$ and $S(\mathcal{C}', \mathcal{I})$, some information on the permutation between $\mathcal{C}$ and $\mathcal{C}'$ is revealed. The number of distinct values taken by a given signature for a given code $\mathcal{C}$ is thus of crucial importance to measure its efficiency.

Definition 6. *Let* $\mathcal{C}$ *be a code of length* n.

- *A signature* S *is said to be* discriminant *for* $\mathcal{C}$ *if there exist* i *and* j *in* $\mathcal{I}$ *such that* $S(\mathcal{C}, i) \neq S(\mathcal{C}, j)$.
- *A signature* S *is said to be* fully discriminant *for* $\mathcal{C}$ *if for all* i *and* j *distinct in* $\mathcal{I}$, $S(\mathcal{C}, i) \neq S(\mathcal{C}, j)$.

If $\mathcal{C}' = \sigma(\mathcal{C})$ and if S is fully discriminant for $\mathcal{C}$, then, for all i in $\mathcal{I}$, there exists a unique element j in $\mathcal{I}$ such that $S(\mathcal{C}, i) = S(\mathcal{C}', j)$, and we have $\sigma(i) = j$ and we thus obtain the permutation σ.

Description of the algorithm

If we assume the existence of a procedure `find_fd_signature` which returns for any generator matrix G a signature which is fully discriminant for $\mathcal{C} = \langle \mathsf{G} \rangle$, then Algorithm 4.1 will recover the permutation between permutation-equivalent codes. In fact it is easy to produce a procedure `find_fd_signature`, but the signature it returns has an exponential complexity.

Algorithm 4.1 The support splitting algorithm

Input: G_1 and G_2 two $k \times n$ matrices
Output: a permutation
$S \leftarrow$ `find_fd_signature`$(T[\,] = \emptyset)$
for $i \in \mathcal{I}$ **do**
  $T[S(\mathsf{G}_1, i)] \leftarrow i$
for $i \in \mathcal{I}$ **do**
  $\sigma[i] \leftarrow T[S(\mathsf{G}_2, i)]$

The difficulty is to obtain, for as many codes as possible, a fully discriminant signature which can be computed in polynomial time. The hull [2] of a linear code $\mathcal{C}$ is defined as its intersection with its dual $\mathcal{H}(\mathcal{C}) = \mathcal{C} \cap \mathcal{C}^\perp$. It has some very interesting features:

(i) It commutes with permutations: $\mathcal{H}(\sigma(\mathcal{C})) = \sigma(\mathcal{H}(\mathcal{C}))$
(ii) The hull of a random code is almost always of small dimension [69].
(iii) For all $i \in \mathcal{I}$, exactly one of the three sets $\mathcal{H}(\mathcal{C}_{.\mathcal{I}\setminus\{i\}})$, $\mathcal{H}(\mathcal{C}^\perp{}_{.\mathcal{I}\setminus\{i\}})$ and $\mathcal{H}(\mathcal{C})_{.\mathcal{I}\setminus\{i\}}$ is strictly greater than the other two, which are equal [68].

We consider the following signature

$$S(\mathcal{C}, i) = \left(W(\mathcal{H}(\mathcal{C}_{.\mathcal{I}\setminus\{i\}})), W(\mathcal{H}(\mathcal{C}^\perp{}_{.\mathcal{I}\setminus\{i\}}))\right)$$

where $W(\mathcal{C})$ denotes the weight enumerator polynomial of $\mathcal{C}$. Because of (i), the mapping $S()$ is a signature, because of (ii) it is almost always computable in polynomial time, and because of (iii), it is discriminant (but not always fully discriminant).

We apply $S()$ to all positions of a code and group those with the same value. We obtain a partition of the support. Using that partition we can refine the signature and eventually obtain a fully discriminant signature in a (conjectured) logarithmic number of refinements. When used on two codes of length n, the heuristic complexity for the whole procedure is

$$\mathcal{O}\left(n^3 + 2^h n^2 \log n\right)$$

where h is the dimension of the hull.

The first term is the cost of the Gaussian elimination needed to compute the hull. The second term is the (heuristic) number of refinements, $\log n$, multiplied by the cost of one refinement (n weight enumerator of codes of dimension h and length n). In practice, for random codes, the hull has a small dimension with overwhelming probability [69] and the dominant cost for the average case is $\mathcal{O}(n^3)$. The worst case happen when the hull's dimension is maximal: weakly-self dual codes ($\mathcal{C} \subset \mathcal{C}^\perp$) are equal to their hulls. The algorithm becomes intractable with a complexity equal to $\mathcal{O}(2^k n^2 \log n)$. This is the case in particular of the Reed-Muller codes used in Sidelnikov's system [70]. For more details on the support splitting algorithm, see [68].

4.3 Recognizing code structures

Only for alternant and algebraic geometric codes it is sufficient to publish a systematic generator matrix of a code permutation equivalent to the secret one in order to hide the private key from an attacker. In this section we want to give the reader an intuition, in which cases and how the structure of an algebraic code can be recognized or not.

GRS Codes

In 1992 V.M. Sidelnikov and S.O. Shestakov proposed an attack on the GRS Niederreiter PKC (compare §2.1) which reveals an alternative private key in polynomial time [71]. We consider this attack to be worth mentionable, as Goppa codes are subfield subcodes of GRS codes. Even though, the results from [71] do not affect the security of the original McEliece PKC.

In their attack, Sidelnikov and Shestakov take advantage of the fact, that the check matrix of GRS code is of the form (see §6.2)

$$\bar{\mathsf{H}} = \begin{pmatrix} z_1 a_1^0 & z_1 a_1^1 & \cdots & z_1 a_1^s \\ z_2 a_2^0 & z_2 a_2^1 & \cdots & z_2 a_2^s \\ \vdots & & \ddots & \vdots \\ z_n a_n^0 & z_n a_n^1 & \cdots & z_n a_n^s \end{pmatrix}^\top \in \mathbb{F}_q^{n\times(s+1)} \,. \tag{5}$$

A public key is of the form $\mathsf{H}' = \mathsf{M}\bar{\mathsf{H}}\mathsf{P}$, where M is a non-singular matrix and P a permutation matrix. The permutation matrix P does not change the structure of $\bar{\mathsf{H}}$, so we don't have to worry about P. Sidelnikov and Shestakov use the fact, that each entry of the row $\mathsf{H}'_{i\cdot}$ can be expressed by a polynomial f of degree $\leq s$ in a_i. From this observation one can derive a system of polynomial equations whose solution yields the private key.

To perform the attack, it is necessary to see, that we can assume that a_1, a_2, a_3 are distinguished elements, so we extend $\mathbb{F}_q$ by ∞: $\mathbb{F} := \mathbb{F}_q \cup \infty$ with $1/\infty = 0$ with $1/0 = \infty$ and $f(\infty) = f_s$ for every polynomial $f(x) = \sum_{j=0}^{s} f_j x^j$ of degree $\leq s$ over $\mathbb{F}_q$. Sidelnikov and Shestakov show that for every birational transformation, i.e. $\mathbb{F}$-automorphism

$$\phi(x) = \begin{cases} \frac{a}{c} & c \neq 0, x = \infty \\ \frac{ax+b}{cx+d} & \text{otherwise} \end{cases} \quad \text{with } a,b,c,d \in \mathbb{F}_q,\ ad - bc \neq 0$$

there exist $z'_1, \cdots, z'_1$ and a matrix M' such that

$$\mathsf{H}' = \mathsf{M}\left(\mathsf{M}'\right)^{-1} \cdot \begin{pmatrix} z'_1\phi(a_1)^0 & z'_1\phi(a_1)^1 & \cdots & z'_1\phi(a_1)^s \\ z'_2\phi(a_2)^0 & z'_2\phi(a_2)^1 & \cdots & z'_2\phi(a_2)^s \\ \vdots & & \ddots & \vdots \\ z'_n\phi(a_n)^0 & z'_n\phi(a_n)^1 & \cdots & z'_n\phi(a_n)^s \end{pmatrix}^\top .$$

Thus, without loss of generality, we can assume that H' defines the (dual) code with codewords

$$(z'_1 f(1), z'_2 f(0), z'_3 f(\infty), z'_4 f(a_4), z'_5 f(a_5), \cdots, z'_n f(a_n)),$$

where f varies over the polynomials of degree $\leq s$ over $\mathbb{F}_{2^m}$. This means, H' defines an extended GRS code (see Definition 9) with $a_1 = 1, a_2 = 0$ and $a_3 = \infty$. Note that because $a_3 = \infty$ we have $a_i \neq \infty$ for all $i \neq 3$.

The general idea of the attack is the following: If we take two codewords with $s-1$ common zeroes, then the corresponding polynomials π_1, π_2 have $s-1$ common factors, while each polynomial is of degree $\leq s$. As we have noted above, we can assume that $\pi_1(0) = 1 = \pi_2(1)$ and $\pi_1(1) = 0 = \pi_2(0)$, which leads to

$$\frac{\pi_1(x_j)}{\pi_2(x_j)} = \frac{\pi_1(\infty)}{\pi_2(\infty)} \cdot \frac{x_j - 1}{x_j} = \frac{\pi_1(a_3)}{\pi_2(a_3)} \cdot \frac{x_j - 1}{x_j},$$

and thus reveals a_j on all positions where neither π_1 nor π_2 are zero. We can repeat this procedure with other pairs of polynomials to obtain the whole vector $(a_1, a_2, \cdots, a_n)$. Taking a birational transform ϕ, such that $(\phi(a_1), \phi(a_2), \cdots, \phi(a_n))$ does not contain the ∞ element, we can recover $(z_1, \cdots, z_n)$ by setting $z_1 = 1$ and employing Gauss's algorithm afterwards. As pairs of codewords with $s-1$ common zeroes can be found by computing a systematic check matrix, the algorithm has a running time of $\mathcal{O}\left(s^4 + sn\right)$.

Remark 2. There were two proposals to modify the GRS Niederreiter cryptosystem: The first one is by E. Gabidulin and consists in adding artificial errors to the generator matrix [24] whereas the second by P. Loidreau uses a subcode of a GRS code (compare Table 1). While the first proposal did not receive much attention so far, the second one was cryptanalyzed by C. Wieschebrink [78], who showed how to attack that modification for small parameter sets by finding pairs of code words with $s-1-i$ common zeroes and guessing i elements from $(x_4, \cdots, x_n)$. This attack can be applied to the Niederreiter PKC variant proposed in [24] in certain cases, e.g., by puncturing the public code. Even if these attacks have exponential runtime, we are not sure if secure parameter sets have a better performance than McEliece's PKC with Goppa codes.

Remark 3. The attack on the GRS Niederreiter PKC can not be applied to McEliece/Niederreiter cryptosystems using Goppa codes. Even though for every Goppa code there is a check matrix H which has the same structure as the check matrix $\bar{\mathsf{H}}$ for GRS codes in equation (5) (see [47]), there is no analogous interpretation of H' for the Niederreiter cryptosystem using Goppa codes. We are able to view H as a matrix over $\mathbb{F}_2$ if we are using Goppa codes, whereas this doesn't work for GRS codes. Thus we have different matrices M: $\mathsf{M} \in \mathbb{F}_{2^m}^{(s+1)\times(s+1)}$ for the GRS case and $\mathsf{M} \in \mathbb{F}_2^{m(s+1)\times m(s+1)}$ for Goppa codes. Thus, in the latter case, H' has no obvious structure, as long as M is unknown.

Rank metric codes

So called Gabidulin codes are a subclass of Srivastava codes, which are MDS codes (i.e., they have a check matrix in form of Equation (5) and their minimum distance is $d = n-k+1$) [47], for which an efficient decoding algorithm exists [27]. These codes were introduced into cryptography together with the notion of rank metric (see Definition 11). The class of Gabidulin codes is the only class of codes for which an algorithm is known, which can correct errors in Hamming and rank metric. For now, however, we omit the interesting notion of rank metric, but give a general intuition, why one can recognize the structure of a Gabidulin code even better then the one of a GRS code and why the modifications proposed in Table 1 do not serve to hide their structure sufficiently for cryptographic purposes.

We will define Gabidulin codes by their generator matrix. For ease of notation we introduce the operator λ_f, which maps a matrix $\mathsf{M} = (m_{ij})$ to a blockmatrix:

$$\begin{aligned} \lambda_f : \mathbb{F}_{q^m}^{m\times n} &\to \mathbb{F}_{q^m}^{m(f+1)\times n} \\ \mathsf{M} &\mapsto \begin{bmatrix} \mathsf{M} \\ \mathsf{M}^{[q]} \\ \vdots \\ \mathsf{M}^{[q^f]} \end{bmatrix}, \end{aligned} \tag{6}$$

where $\mathsf{M}^{[x]} := (m_{ij}^x)$.

Definition 7. *Let $\mathbf{g} \in \mathbb{F}_{q^m}^n$ be a vector s.t. the components g_i, $i = 1, \cdots, n$ are linearly independent over $\mathbb{F}_q$. This implies that $n \leq m$. The $[n, k]$ Gabidulin code $\mathcal{G}$ is the rank distance code with generator matrix*

$$\mathsf{G} = \lambda_{k-1}(\mathbf{g}). \tag{7}$$

The vector $\mathbf{g}$ is said to be a *generator vector* of the Gabidulin code $\mathcal{G}$ (It is not unique, as all vectors $a\mathbf{g}$ with $0 \neq a \in \mathbb{F}_{q^m}$ are generator vectors of $\mathcal{G}$). Further, if $\mathsf{T} \in \mathbb{F}_q^{n\times n}$ is an invertible matrix, then $\mathsf{G}\cdot\mathsf{T}$ is the generator matrix of the Gabidulin code with generator vector $\mathbf{g}\mathsf{T}$. An error correction algorithm based on the "right Euclidian division algorithm" runs in $\mathcal{O}\left(d^3 + dn\right)$ operations over $\mathbb{F}_{q^m}$ for $[n, k, d]$ Gabidulin codes [27]. The property, that a matrix G generates a Gabidulin code is invariant under the operator $\Lambda_f(\mathsf{M})$:

Lemma 1. *If G is a generator matrix of an $[n, k]$ Gabidulin code $\mathcal{G}$ with $k < n$, then $\Lambda_f(\mathsf{G}^{\mathrm{pub}})$ is a generator matrix of the Gabidulin code with the same generator vector as $\mathcal{G}$ and dimension* $\min\{n, k + f\}$.

Another nice property of Gabidulin codes is, that the dual code of an $[n, k]$ Gabidulin code is an $[n, n-k]$ Gabidulin code (see [27]):

Lemma 2. *Let $\mathcal{G}$ be an $[n,k]$ Gabidulin code over $\mathbb{F}_{q^m}$ with generator vector g. Then $\mathcal{G}$ has a check matrix of the form*

$$\mathsf{H} = \lambda_{n-k-1}\left(\mathbf{h}^{[1/q^{n-k-1}]}\right) \in \mathbb{F}_{q^m}^{n-k\times n} .$$

Further, the vector $\mathbf{h}$ *is uniquely determined by* $\mathbf{g}$ *(independent from k) up to a scalar factor $\gamma \in \mathbb{F}_{q^m} \setminus \{0\}$. We will call* $\mathbf{h}$ *a check vector of $\mathcal{G}$.*

The major disadvantage of Gabidulin codes, is the fact, that one can easily distinguish a random $k \times n$ matrix M from an arbitrary generator matrix G of an $[n,k]$ Gabidulin code by a quite simple operation: The matrix $\lambda_1(\mathsf{G})$ defines an $[n,k+1]$ code, while the matrix $\lambda_1(\mathsf{M})$ will have rank $> k+1$ with overwhelming probability [45].

Remark 4. Unlike for GRS codes, it is not sufficient to take the generator matrix $\mathsf{G}_{\mathrm{SUB}}$ of an $[n,k-l]$ subcode of an secret $[n,k]$ Gabidulin code $\mathcal{G} = \langle \mathsf{G} \rangle$ to hide the structure as it was proposed in [5]. It is easy to verify, that $\lambda_1(\mathsf{G}_{\mathrm{SUB}})$ defines a subcode of $\langle \lambda_1(\mathsf{G}) \rangle$ and thus any full rank vector in the dual of $\lambda_{n-k-2}(\mathsf{G}_{\mathrm{SUB}})$ gives a Gabidulin check vector which allows to decode in $\mathsf{G}_{\mathrm{SUB}}$.

There were plenty of other proposals on how to use Gabidulin codes for cryptography, most with the notation of rank metric, see §6.3. However, as mentioned before, all these variants proved to be insecure [53, 57].

Reed-Muller Codes

Reed-Muller codes were considered for cryptographic use by Sidelnikov [70]. His basic proposal is to replace the Goppa code in McEliece's scheme by a Reed-Muller code, which can be defined as follows:

The Reed-Muller code in m variables of degree r consists of all codewords which can be obtained by evaluating some polynomial in $\mathbb{F}_2[x_1, \cdots, x_m]$ of degree at most r at all possible variable assignments, see [47]. Lexicographic ordering of the 2^m possible assignments leads to the following recursive description of the canonical generator matrix $\mathsf{R}(r,m)$, which is reducible:

$$\mathsf{R}(r,m) = \left[\begin{array}{c|c} \mathsf{R}(r,m-1) & \mathsf{R}(r,m-1) \\ \hline 0 & \mathsf{R}(r-1,m-1) \end{array}\right], \tag{8}$$

where $\mathsf{R}(r,m) = \mathsf{R}(m,m)$ for $r > m$ and $\mathsf{R}(0,m)$ is the codeword of length 2^m which is one at all positions. The code $\langle \mathsf{R}(r,m) \rangle$ is a $[2^m, \sum_{i=0}^{r} \binom{m}{i}, d]$ code with $d = 2^{m-r}$ and is a subcode of $\langle \mathsf{R}(r+1,m) \rangle$ [47].

From the construction of a Reed-Muller code it is easy to see, that each low weight codeword in $\mathsf{R}(r,m)$ can be represented as a product of $m-r$ pairwise different linear factors. Due to this large number of low weight codewords

(there exist about $2^{mr-r(r-1)}$ of them), Stern's algorithm [72] and its variants allow to find low weight codewords in Reed-Muller codes efficiently, compare §3.3.

Now, let $\mathsf{P} \in \mathbb{F}_2^{n\times n}$ be a permutation matrix. The main observation, which allows to recover P from some generator matrix $\mathsf{G}^{\mathrm{pub}}$ of $\langle \mathsf{R}\,(r,m)\,\mathsf{P}\rangle$ is that each low weight codeword of $\left\langle \mathsf{G}^{\mathrm{pub}}\right\rangle$ can be "factored". Indeed, each low weight codeword $\mathbf{v}$ in $\left\langle \mathsf{G}^{\mathrm{pub}}\right\rangle$ can be written as the "product" of a low weight codeword $\bar{\mathbf{v}}$ in $\mathsf{R}(r-1,m)\mathsf{P}$ and a low weight codeword $\hat{\mathbf{v}}$ in $\mathsf{R}(1,m)\mathsf{P}$:

$$\mathbf{v} := \bar{\mathbf{v}} \odot \hat{\mathbf{v}} := (\bar{\mathbf{v}}_1 \cdot \hat{\mathbf{v}}_1, \bar{\mathbf{v}}_2 \cdot \hat{\mathbf{v}}_2, \cdots, \bar{\mathbf{v}}_n \cdot \hat{\mathbf{v}}_n)$$

The goal is to find the factor $\bar{\mathbf{v}}$ of $\mathbf{v}$. If a sufficiently large number of low weight codewords of $\left\langle \mathsf{G}^{\mathrm{pub}}\right\rangle$ have been factored, the code $\mathsf{R}(r-1,m)\mathsf{P}$ can be reconstructed. Iteratively reducing the problem it remains to solve the problem to recover P from $\mathsf{R}(1,m)\mathsf{P}$, which is trivial [50].

Remark 5. The application of N. Sendrier's "Support Splitting Algorithm" (SSA, see §4.2) for finding the permutation between permutation equivalent codes is not efficient for Reed-Muller codes. The runtime of SSA is exponential in the dimension of the hull of a code $\mathcal{C}$, i.e. the dimension of $\mathcal{C} \cup \mathcal{C}^{\perp}$, which is large, if $\mathcal{C}$ is a Reed-Muller code. Thus, Sidelnikov's proposal can not be attacked via the SSA.

In [50] L. Minder gives an algorithm to deduce the factor $\bar{\mathbf{v}}$ of $\mathbf{v}$ efficiently. We will assume that $\mathsf{P} = \mathsf{Id}$ in the following, since the algorithm does not depend on P. Assume, that $\mathbf{v}$ is a low weight codeword in $\langle \mathsf{R}(r,m)\rangle$, then we may well assume, that the corresponding polynomial can be written as $v = v_1 \cdot v_2 \cdots v_r$ (after a change of basis). Now, the code $\mathcal{C}$ consisting of all codewords with support disjoint from $\mathbf{v}$ can be represented as a polynomial

$$f = f(v_1, v_2, \cdots, v_m) = \sum_{I\subseteq\{1,2,\cdots,r\}} f_I \cdot \prod_{i\in I} v_i$$

with $f_I \in \mathbb{F}_2[v_{r+1}, v_{r+2}, \cdots, v_m]$. Further, since f and v have disjoint support, we have $f(\underbrace{1,1,\cdots,1}_{r \text{ times}}, v_{r+1}, v_{r+2}, \cdots, v_m) = 0$ and thus

$$\sum_{I\subseteq\{1,2,\cdots,r\}} f_I = 0.$$

We can see, that restricting the codewords of disjoint support to the ones with a fixed value for $(v_1, \cdots, v_r) \neq (1,1,\cdots,1)$ we obtain an permuted version of $\mathsf{R}(r-1, m-r-1)$ (after puncturing). This shows, that the codewords with disjoint support from $\mathbf{v}$ form a code which is a permuted concatenated code build of $2^r - 1$ blocks, each a Reed-Muller code of degree $r-1$ in $m-r-1$ variables, i.e. there is a permutation Π such that

$$\mathcal{C}\Pi \subseteq \underbrace{(0,0,\cdots,0)}_{2^{m-r} \text{ times}} \otimes \left(\bigotimes_{i=1}^{2^r-1} \langle \mathsf{R}(r-1, m-r-1) \rangle \right).$$

Thus, each of this inner blocks together with the support of $\mathbf{v}$ gives a low weight codeword in $\langle \mathsf{R}(r-1,m)\mathsf{P} \rangle$.

Even if the identification of the inner blocks of a concatenated code has been studied in [64], Minder proposes to identify the different blocks by statistical analysis: For a low weight codeword $\mathbf{y}$, he states, that the probability, that $\mathbf{y}_i = 1$ and $\mathbf{y}_j = 1$ is independent if and only if i and j do not belong to the same inner block.

Remark 6. The code $\langle \mathsf{R}(r,m)\mathsf{P} \rangle$ is a permutation of a concatenated code $\subseteq \bigotimes_{i=1}^{2^r} \langle \mathsf{R}(r-1, m-r-1) \rangle$, too. Thus, one might think of applying the statistical analysis directly to $\mathsf{R}(r,m)\mathsf{P}$ in order to partition the code. However, Minder states that the support of the low weight code words of $\mathsf{R}(r,m)$ is too large (i.e. twice the length of each block) to allow sampling from the desired space.

Minder's runtime analysis shows, that the crucial point is to find the low weight codewords in $\mathcal{C}$, which however, due to the large number of low weight codewords is practical for reasonable parameter sets, turning Sidelnikov's cryptosystem inefficient. For $r = 3$ and $m = 11$ for example, his algorithm allows to recover the permutation P in less than one hour on a desktop PC.

Structural attacks on the McEliece cryptosystem

Binary Goppa codes were proposed by McEliece in the original version of his system. So far, all known structural attacks on Goppa codes have an exponential cost.

We assume t-error correcting binary irreducible Goppa codes of length $n = 2^m$ over $\mathbb{F}_{2^m}$ are used for the key generation. The secret key is the code $\Gamma(\mathbf{L}, g)$ which consists of

- a *generator*, a monic irreducible polynomial $g(z)$ of degree t over $\mathbb{F}_{2^m}$
- a *support*, a vector $\mathbf{L} \in \mathbb{F}_{2^m}^n$ with distinct coordinates (in fact, with $n = 2^m$, this defines a permutation).

If either the support or the generator is known, the other part of secret can be recovered in polynomial time from the public key G^{pub}.

1. If the support $\mathbf{L}$ is known, then a multiple of $g(z)$ can be obtained from any codeword by using equation (12) page 139. Codewords can easily be obtained from G^{pub}, and after a few gcds (usually one is enough) the generator polynomial is obtained.
2. If the generator polynomial $g(z)$ is known, we construct a generator matrix G of the Goppa code of generator $g(z)$ and support $\mathbf{L}_0$ (where $\mathbf{L}_0$ is fixed

and chosen arbitrarily), and we obtain the secret vector $\mathbf{L}$ by applying the support splitting algorithm to G and G^{pub} (the permutation between G and G^{pub} will also be the permutation between $\mathbf{L}_0$ and $\mathbf{L}$).

In both cases, we obtain an exhaustive search attack, either by enumerating the permutations (proposed by Gibson in [31]) or by enumerating the irreducible polynomials [46]. There are $\approx 2^{tm}/t = n^t/t$ irreducible polynomials compared to $n! = \mathcal{O}(\sqrt{n}(n/\mathrm{e})^n)$ permutations. The second attack is always more efficient. To evaluate the cost of this attack we consider

- the number of monic irreducible polynomials of degree t over $\mathbb{F}_{2^m}$ [43, p. 93], equal to $\approx 2^{tm}/t = n^t/t$.
- the cost of the support splitting algorithm, equal to $\mathcal{O}(n^3)$, because Goppa codes behave like random codes and have a small hull.
- the number of distinct pairs support/generator that produce the same Goppa code, which is almost always equal to $m2^m = n \log_2 n$ [31].

We multiply the first two numbers and divide by the third and we get $\mathcal{O}(n^{t+2}/t \log n)$. In fact, it is possible to do slightly better by considering extended codes (an overall parity check bit is appended). The number of distinct pairs support/generator that produce the same extended Goppa code is almost always equal to $m2^m(2^{2m} - 1)$ (see [47, p. 347]). The support splitting algorithm can be applied on extended code and the complexity of the attack is reduced to

$$\mathcal{O}\left(\frac{n^t}{t \log n}\right) = \mathcal{O}\left(\frac{2^{tm}}{tm}\right).$$

This is currently the best known structural attack on McEliece encryption scheme using Goppa codes. As the best decoding attack is upper bounded by $\mathcal{O}(2^{(n-k)/2}) = \mathcal{O}(2^{tm/2})$ (see [4] for instance), structural attacks are never better than decoding attacks.

Choosing the secret codes: general pitfalls

Beyond the existence of an efficient structural attack today, what kind of assumptions do we want to (or have to) make for arguing of McEliece's scheme security? First, obviously, the family of codes used to produce the keys is critical. Binary Goppa codes are safe (or seem to be), but not Reed-Solomon codes [71], concatenated codes [64], elliptic codes [49], Reed-Muller codes [50] (to some extend), and many other unpublished attempts.

Indistinguishability is the strongest security assumption related with structural attacks. Informally, it says that it is not computationally feasible to tell apart a generator matrix of a random code from a generator matrix of a particular family. When it holds, the security of the corresponding public-key system can be reduced to the hardness of decoding, for which very strong arguments exist.

Indistinguishability is conjectured for binary Goppa codes, and in practice, no property is known that can be computed from a generator matrix in polynomial time and which behaves differently for binary Goppa codes and for binary linear codes. To our knowledge, this is the only such family of codes with an efficient decoding algorithm.

Using other families of codes in public key cryptography should be considered with great care. There are at least two possible pitfalls

- Families with high performance decoding, like concatenated codes, turbo-codes or LDPC codes, have many low weight codewords in their duals. Those low weight codewords may be easy to find and are likely to leak some of the code structure.
- As we have seen previously in this section (§4.3 and §4.3), families with optimal or sub-optimal combinatorial properties are dangerous too. For instance, (generalized) Reed-Solomon codes are MDS (the highest possible minimum distance), elliptic codes are almost MDS (minimum distance is just one less), in both case minimum weight codewords are not hard to find and reveal a lot of information on the code structure. Reed-Muller codes are highly structured, and though they have an optimal resistance to the support splitting algorithm (they are weakly self-dual), Lorenz Minder has exhibited a structural attack which is more efficient than the decoding attack.

Finally, let us mention algebraic geometry codes, proposed for cryptography by Janwa and Moreno [35]. They are probably insecure for small genus (Minder's work) but otherwise, their security status is unknown.

5 Practical aspects

The practice of McEliece's PKC or more generally of a code-based PKC raises many questions. We address here a few of them in this section. The main advantage of McEliece's scheme is a low algorithmic complexity for encryption and decryption and its main drawback is a large public key size. We will stress the first point and examine what can be done for the second. Also, for practical purposes, the system suffers from many weaknesses, most of them related to malleability. We will examine the generic and ad-hoc semantically secure conversions that solve those issues.

5.1 Fast en- and decryption for the McEliece PKC

We describe here the implementation of the McEliece encryption scheme. The error correcting code will be a binary irreducible t error-correcting Goppa code $\mathcal{G}$ of length $n = 2^m$ and dimension $k = n - tm$. We denote $D_{\mathcal{G}} : \{0,1\}^n \to \mathcal{G}$ a t-error correcting procedure for $\mathcal{G}$ (see §6.1). The private key is the decoder $D_{\mathcal{G}}$ and the public key is a generator matrix G of $\mathcal{G}$.

We assume the existence of an injective mapping $\phi_{n,t} : \{0,1\}^\ell \to \mathcal{W}_{n,t}$ easy to compute and to invert (see §5.1). The key features of the implementation we describe are presented in Algorithm 5.1. The two main differences from the original proposal are:

1. The public key is chosen in systematic form $\mathsf{G}^{\text{syst}} = (\mathsf{Id} \mid \mathsf{R})$.
2. The mapping $\phi_{n,t}$ will be used to encrypt ℓ additional information bits.

Those modifications do not alter the security of the system as long as a semantically secure conversion is used (such a conversion is needed anyway). Moreover, those conversions (see §5.3) require the use of $\phi_{n,t}$, so, for practical purpose, that part of the computation has to be done anyway.

Algorithm 5.1 Modified McEliece encryption scheme

- **Public key:** a $k \times (n-k)$ binary matrix R
- **Private key:** a decoder $D_{\mathcal{G}}$ for the code $\mathcal{G}$ spanned by $(\mathsf{Id} \mid \mathsf{R})$
- **Encryption:** the plaintext is $(\mathbf{m}_1, \mathbf{m}_2) \in \{0,1\}^k \times \{0,1\}^\ell$
  the ciphertext is $\mathbf{y} = (\mathbf{m}_1, \mathbf{m}_1\mathsf{R}) + \phi_{n,t}(\mathbf{m}_2) \in \{0,1\}^n$
- **Decryption:** the ciphertext is $\mathbf{y} \in \{0,1\}^n$
  compute the codeword $\mathbf{x} = D_{\mathcal{G}}(\mathbf{y})$, with $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_1\mathsf{R})$
  the plaintext is $(\mathbf{m}_1, \mathbf{m}_2) = (\mathbf{x}_1, \phi_{n,t}^{-1}(\mathbf{y} - \mathbf{x}))$

The algorithmic complexity of the encryption and decryption procedures are relatively easy to analyse.

- The encryption complexity is dominated by the vector/matrix multiplication (k times $k \times (n-k)$) and the call to $\phi_{n,t}$. In practice those two costs are comparable.
- The decryption complexity is dominated by the decoding $D_{\mathcal{G}}(\mathbf{y})$ and the call to $\phi_{n,t}^{-1}$. In practice the decoding is much more expensive.

McEliece with a systematic public key

Let G be the public key of an instance of McEliece cryptosystem with parameters (n, k, t). Let $\mathsf{G}^{\text{syst}} = (\mathsf{Id} \mid \mathsf{R}) = \mathsf{UG}$ be a systematic generator matrix of the same code (w.l.o.g. the first k column of G are non-singular and U is a $k \times k$ matrix which can be computed from G in polynomial time).

For any G, we denote $\Psi_{\mathsf{G}}(\mathbf{m}, \mathbf{e}) = \mathbf{m}\mathsf{G} + \mathbf{e}$. Using $\Psi_{\mathsf{G}^{\text{syst}}}$ instead of Ψ_{G} for the encryption has many advantages:

- the public key is smaller, as it has a size of $k(n-k)$ bits instead of kn,
- the encryption is faster, as we multiply the plaintext by a smaller matrix,
- the decryption is faster, as the plaintext is a prefix of the ciphertext cleared of the errors.

The drawback is a "decrease" of the semantic security. The following example is taken from [65, p. 34], and is the beginning of a ciphertext for an instance of McEliece using a systematic public key:

```
Le{ cryptosystèmas0basés suv les code{‘corveãteurs soît-ils sýòs?
```

Obviously, there is a leak of information. However, since we have $\Psi_{\mathsf{G}}(\mathbf{m}, \mathbf{e}) = \Psi_{\mathsf{G}^{\text{syst}}}(\mathbf{m}\mathsf{U}^{-1}, \mathbf{e})$, any inversion oracle for $\Psi_{\mathsf{G}^{\text{syst}}}$ can be transformed in an inversion oracle for Ψ_{G}. Thus, if the plaintext $\mathbf{m}$ is uniformly distributed, both versions are equally secure. In practice, this means that a semantically secure conversion (see §5.3) will enable us to use G^{syst} without loss of security.

Encoding constant weight words

The problem here is to exhibit, for given n and t, an efficient injective mapping into the set of binary words of length n and weight t, $\phi_{n,t} : \{0,1\}^{\ell} \to \mathcal{W}_{n,t}$. This mapping is needed for implementing Niederreiter scheme and is also used in most semantically secure conversions. In practice we want ℓ to be close to $\lfloor \log_2 \binom{n}{t} \rfloor$. Else, we risk a loss of security.

Enumerative method.

This method is optimal in terms of information rate and can be traced back to [15, 62]. It is based on the following bijective mapping

$$\begin{aligned} \theta : \quad \mathcal{W}_{n,t} &\longrightarrow \left[0, \tbinom{n}{t}\right[\\ (i_1, \ldots, i_t) &\longmapsto \tbinom{i_1}{1} + \tbinom{i_2}{2} + \cdots + \tbinom{i_t}{t} \end{aligned}$$

where the element of $\mathcal{W}_{n,t}$ is represented by its non-zero positions in increasing order $0 \le i_1 < i_2 < \ldots < i_t < n$. Computing θ requires the computation of t binomial coefficients. When t is not too large, computing the inverse θ^{-1} is not significantly more expensive thanks to the following inversion formula

$$x = \binom{i}{t} \Leftrightarrow i = X + \frac{t-1}{2} + \frac{t^2-1}{24}\frac{1}{X} + \mathcal{O}\left(\frac{1}{X^3}\right), X = (t!x)^{1/t}. \quad (9)$$

We can define $\phi_{n,t}$ as the restriction of θ^{-1} to the interval $\left[0, 2^{\ell}\right[$ where $\ell = \lfloor \log_2 \binom{n}{t} \rfloor$. Both $\phi_{n,t}$ and $\phi_{n,t}^{-1}$ can be obtained by computing t binomial coefficients and have a cost of $\mathcal{O}(t\ell^2) = \mathcal{O}(t^3 m^2)$ binary operations.

The decoding procedure is described in Algorithm 5.2. It uses formula (9) for inverting the binomial coefficients. In fact, this inversion does not require a great precision as the result we seek is an integer, not a floating point number. In practice `invert_binomial` has a negligible cost compared with the computation of the binomial coefficients.

Algorithm 5.2 Enumerative decoding

Input: $x \in \left[0, \binom{n}{t}\right[$
Output: t integers $0 \leq i_1 < i_2 < \ldots < i_t < n$
$j \leftarrow t$
while $j > 0$ **do**
  $i_j \leftarrow \texttt{invert\_binomial}(x, j)$
  $x \leftarrow x - \binom{i_j}{j}$
  $j \leftarrow j - 1$

where $\texttt{invert\_binomial}(x, t)$ returns the integer i such that $\binom{i}{t} \leq x < \binom{i+1}{t}$

Recursive source coding methods.

Those methods consist, as for the enumerative method, in finding a binary encoder for the source $\mathcal{W}_{n,t}$ equipped with the uniform probability (i.e. a compression algorithm). The idea is to consider a simpler approximative source model which allows a faster encoding and decoding. Linear time methods were proposed in [63,67]. It consists in a (variable length) encoder $\mathcal{W}_{n,t} \to \{0,1\}^*$, with the additional requirement that any (long enough) binary sequence can be decoded into a sequence of words of $\mathcal{W}_{n,t}$. For instance in [67], an element of $\mathcal{W}_{n,t}$ is first represented by a t-tuple $(\delta_1, \ldots, \delta_t)$ of integers where δ_i is the number of '0's between the $(i-1)$-th and the i-th '1' (the 0-th '1' is the beginning of the word). The encoding is recursively defined as:

$$\Psi_{n,t}(\delta_1, \delta_2, \ldots, \delta_t) = (f_{n,t}(\delta_1), \Psi_{n-\delta_1-1,t-1}(\delta_2, \ldots, \delta_t))$$

where $f_{n,t}$ is a source encoder for the set of integers $\{0, 1, \ldots, n-t\}$ equipped with the probability distribution

$$P_{n,t}(i) = \frac{\binom{n-i-1}{t-1}}{\binom{n}{t}}, i = 0, \ldots, n-t.$$

The model is then simplified. We choose d an integer such that

$$\sum_{i<d} P_{n,t}(i) = 1 - \frac{\binom{n-d}{t}}{\binom{n}{t}} \approx \frac{1}{2} \;\Leftrightarrow\; d \approx \frac{2^{1/t} - 1}{2^{1/t}} \left(n - \frac{t-1}{2}\right)$$

and we define $f_{n,t}$ as

$$f_{n,t}(i) = \begin{cases} 0, B_2(i) & \text{if } 0 \leq i < d \\ 1, f_{n-d,t}(i-d) & \text{if } i \geq d \end{cases}$$

where $B_2()$ encodes the set $\{0, \ldots, d-1\}$ equipped with the uniform distribution (easily derived from the integers binary expansion). The best value of d depends of n and t (it is thus different for every recursive call). Choosing a different value of d is possible but suboptimal in terms of compression rate.

There is a good trade-off when one uses only powers of 2 for d, there is a small loss in average, but a significant advantage in speed.

The recursive method is significantly faster than the enumerative method: the computation time is linear in ℓ instead of quadratic. However the encoder $\Psi_{n,t} : \mathcal{W}_{n,t} \to \{0,1\}^*$ produces a variable length output.

Comments and implementation.

The enumerative method allows constant length encoding with a minimal loss ($\ell = \lfloor \binom{n}{t} \rfloor$). On the other hand, it is relatively slow, even when the binomial coefficients are precomputed. The recursive method can be much faster, however the encoder $\mathcal{W}_{n,t} \to \{0,1\}^*$ has an important length variation. This is unpractical and not recommendable, as it raises some security issues that need to be studied further. For instance if an adversary knows how many bits were used to produce the error, he might be able to use this information. The Table 3 gives the average running time for a $\phi_{n,t}$ (and for its inverse $\phi_{n,t}^{-1}$) build from both methods.

(n,t)	(2048,32)		(2048,40)		(4096,22)		(4096,45)	
	$\phi_{n,t}$	$\phi_{n,t}^{-1}$	$\phi_{n,t}$	$\phi_{n,t}^{-1}$	$\phi_{n,t}$	$\phi_{n,t}^{-1}$	$\phi_{n,t}$	$\phi_{n,t}^{-1}$
enumerative	1980	1550	2530	2090	1440	1080	3160	2750
enumerative[(1)]	560	200	580	210	490	200	620	290
recursive	240	250	250	250	240	230	230	240
recursive[(2)]	150	150	150	150	135	130	140	140

(1) enumerative method with precomputation of the binomial coefficients
(2) recursive method optimized for speed (*vs.* average length)

Table 3. Performance (cycles/byte, Intel Core 2) for various encoding methods

Remark 7. There is another proposal [61] which uses arithmetic coding. It is essentially the same as the enumerative method. It is not clear whether or not this algorithm allows a faster implementation.

Remark 8. A new approach has been considered very recently[8] which allows linear time encoding (around 300 cycles/byte on a processor Intel Core 2) with an optimal constant length. At the time of writing, this work was at a too early stage to be detailed here.

[8] see `http://www-rocq.inria.fr/secret/MCE`

Niederreiter's encryption scheme

Using a systematic public key for Niederreiter's scheme was already known to be harmless [13]. The decoder $D'_{\mathcal{G}} : \{0,1\}^k \to \mathcal{W}_{n,t}$ is slightly different, it takes as argument a syndrome (for $\mathsf{H} = (\mathsf{R}^\top \mid \mathsf{Id})$) and returns an error pattern. The implementation is presented in Algorithm 5.3.

Algorithm 5.3 Modified Niederreiter encryption scheme

- **Public key:** a $k \times (n-k)$ binary matrix R
- **Private key:** a decoder $D'_{\mathcal{G}}$ for the code $\mathcal{G}$ spanned by $(\mathsf{Id} \mid \mathsf{R})$
- **Encryption:** the plaintext is $\mathbf{m} \in \{0,1\}^\ell$
  compute the error $\mathbf{e} = \phi_{n,t}(\mathbf{m}) = (\mathbf{e}_1, \mathbf{e}_2) \in \{0,1\}^k \times \{0,1\}^{n-k}$
  the ciphertext is $\mathbf{s} = \mathbf{e}\left(\mathsf{R}^\top \middle| \mathsf{Id}\right)^\top = \mathbf{e}_1\mathsf{R} + \mathbf{e}_2 \in \{0,1\}^{n-k}$
- **Decryption:** the ciphertext is $\mathbf{s} \in \{0,1\}^{n-k}$
  the plaintext is $\mathbf{m} = \phi_{n,t}^{-1}(D'_{\mathcal{G}}(\mathbf{s}))$

Timings and sizes

In Table 4, numbers for McEliece and Niederreiter encryption schemes are given. They come from `http://www-rocq.inria.fr/secret/MCE`. Implementation uses a systematic public key and information is encoded in the error.

5.2 Reducing storage requirements

Reducing the key size for the McEliece PKC has a long history. Besides the approaches to use different codes than Goppa codes, there were two different attempts: The first uses the automorphism group of Goppa codes [44] and the second the quasi-cyclicity of codes [28]. While the first method was broken [38], the second reduces the number of possible secret keys. However, the quasi-cyclic approach has an interesting application in Stern's ID scheme, reducing the RAM requirements of the scheme. However, the proposal is too recent and further research is probably needed to establish secure parameter sets.

Definition 8. *An $[n,k,d]$ code $\mathcal{G}$ over $\mathbb{F}$ is called s-quasi cyclic if for all $\mathbf{c} \in \mathcal{G}$ the vector $\sigma_s(\mathbf{c})$ is in $\mathcal{G}$, where*

$$\begin{aligned} \sigma_s : \mathbb{F}^n &\to \mathbb{F}^n \\ (c_1, \cdots, c_n) &\mapsto (c_{n-s+1}, \cdots, c_n, c_1, \cdots, c_{n-s}) \end{aligned}$$

denotes a cyclic shift by s positions. If $s = 1$ or $s \times n$ the code is cyclic. A set of vectors G is called generating set if the vectors $\left\{\sigma_s^i(\mathbf{c}) \mid \mathbf{c} \in \mathsf{G}, i \in \mathbb{N}_+\right\}$ span $\mathcal{G}$, where $\sigma_s^i(\mathbf{c}) = \sigma_s(\sigma_s^{i-1}(\mathbf{c}))$.

	(n,t)	(2048,32)	(2048,40)	(4096,22)	(4096,45)
McEliece scheme	plaintext size[(1)]	1928	1888	4024	3904
	ciphertext size[(1)]	2048	2048	4096	4096
	encryption rate[(2)]	176	222	145	192
	decryption rate[(2)]	1780	2260	600	1650
Niederreiter scheme	plaintext size[(1)]	232	280	192	352
	ciphertext size[(1)]	352	440	264	540
	encryption rate[(2)]	360	370	320	340
	decryption rate[(2)]	13600	16700	9800	16900
	public key size[(3)]	73 KB	86 KB	123 KB	234 KB
	key generation[(3)]	$6.70\,10^7$	$9.55\,10^7$	$7.93\,10^7$	$23.1\,10^7$
	security bits[(3)(4)]	91	98	93	140

(1) plaintext and ciphertext sizes in bits
(2) in cycles per plaintext bytes, Intel Core 2
(3) common to both schemes (number of cycles on a processor Intel Core 2)
(4) $\log_2$ of the non-quantum binary workfactor

Table 4. McEliece and Niederreiter encryption scheme

Every cyclic code is s-quasi cyclic for all $s \in \mathbb{N}_+$ and the dual of a s-quasi cyclic code is s-quasi cyclic, too. Each cyclic code has a s-cyclic subcode that is not s'-cyclic for all $s' < s$ if $s|n$.

If one chooses to use a secret s-quasi cyclic $[n,k]$ code with $s|n$ for McEliece's scheme and restricts the possible choice of permutation matrices P to the ones which are of the form

$$\mathsf{P} = \begin{bmatrix} \pi & 0 & \cdots & 0 \\ 0 & \pi & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \pi \end{bmatrix},$$

where π is a $s \times s$ matrix. Then, a systematic generator matrix of $\mathcal{G}$ can be reconstructed from $\mathsf{G}^{\text{pub}} = \mathsf{G} \cdot \mathsf{P}$. Thus, the public key size can reduced by a certain factor.

However, this technique holds some risks, as the number of possible permutations is reduced and part of the structure is revealed (a first approach to attack such a system was reported by A. Otmani, J.P. Tillich and L. Dallot [16]). Second, general decoding algorithms could take advantage of the structure of the code, as it is e.g. the case for iterative or statistical decoding [20, 55]. Third, let $\mathbf{e}$ be an error vector of weight t, H be the generating set of the dual of $\mathcal{G}$ and $\mathbf{s} = \mathbf{e}\mathsf{H}$. Then a cyclic shift of $\mathbf{s}$ by one corresponds to the vector $\sigma_s(\mathbf{e})$.

For Stern's ID scheme, one could chose to use 2-quasi cyclic codes with a single generating vector as proposed in [29]. This reduces drastically the size of memory needed to execute the scheme (from kn to n). However, as for Stern's scheme only a random code is required, one could build the generating matrix G from a random string as well, if a cryptographic strong random number generator is used. This has the same effect of reducing the size of memory needed but does not come with the disadvantage of a quasi cyclic code.

5.3 Semantic security for the McEliece scheme

The McEliece PKC and the Niederreiter scheme are subject to several attacks if not completely random bit-strings are sent. Thus, the schemes as they are only serve for key-agreement protocols and not for encrypting messages. In this section we will point out the weaknesses of the McEliece scheme (and thus the Niederreiter version) against attacks on the semantic security and how to get a semantically secure cryptosystem.

A cryptosystem is called *secure against adaptive chosen ciphertext attacks* (CCA2 secure) if an attacker with access to a decryption oracle (which does not decrypt the ciphertext $\mathbf{c}$) has no advantage in deciphering a given ciphertext $\mathbf{c}$. A PKC is *indistinguishable in the CCA2-model* if the attacker has no advantage in determining for a given ciphertext and two plaintexts which of them was encrypted.

Weaknesses of the McEliece PKC

The main weakness of the McEliece PKC results from the malleability of its ciphertexts. Adding codewords, i.e. rows of $\mathsf{G}^{\mathrm{pub}}$ to a ciphertext yields another valid ciphertext. Therefore, the original McEliece cryptosystem does not satisfy non-malleability. A CCA2 attack can be derived immediately as the adversary can add a second message $\mathbf{m}'$ to $\mathbf{c}$ by computing $\mathbf{c}' = \mathbf{c} \oplus \mathbf{m}'\mathsf{G}^{\mathrm{pub}}$, which will be decrypted by the oracle. Note that malleability is not such a problem in the Niederreiter case, as we can not create new decodable syndromes from old ones with probability significantly larger than t/n.

As a consequence from the malleability, an adversary for the McEliece scheme may use the relation between two encrypted messages to determine error bits [8]. This attack can not be adapted to the Niederreiter cryptosystem. Let $\mathbf{m}_1, \mathbf{m}_2$ be two messages with a known relation Λ, e.g. $\Lambda(\mathbf{m}_1, \mathbf{m}_2) = \mathbf{m}_1 \oplus \mathbf{m}_2$ and $\mathbf{c}_1, \mathbf{c}_2$ the corresponding ciphertexts. Then $\mathbf{c}_1 \oplus \mathbf{c}_2 \oplus \Lambda(\mathbf{m}_1, \mathbf{m}_2)$ will be of weight $\leq 2t \leq n - k$ and at least k error-free positions of $\mathbf{m}_1 \oplus \mathbf{m}_2$ are revealed. This enables an adversary to efficiently guess error bits. A special case of related messages occurs in the *message-resend attack*, where the attacker can recover $\mathbf{z}_1 \oplus \mathbf{z}_2 = \mathbf{c}_1 \oplus \mathbf{c}_2$.

A reaction attack is a weaker version of an adaptively chosen ciphertext attack, in that the attacker does not have access to a full decryption oracle, but

can only observe the receiver's reaction on potential ciphertexts. An adversary may intercept ciphertexts, change a few bits, and watch the reaction of the designated receiver on these modified ciphertexts. Sending modifications of an authentic ciphertext amounts to adding further error bits. If the receiver cannot decode (reaction: repeat request), the corresponding bits were not in error originally. This enables the attacker to recover an error-free information set in at most k iterations. Observe, that such an attack is well possible for the Niederreiter PKC as it does not require the malleable property.

CCA2-secure versions of the McEliece scheme

In [37] Kobara and Imai review possible conversions to turn the McEliece PKC CCA2-secure. Not all generic conversions can be applied to the McEliece PKC, since the McEliece PKC encryption function is not a OWTP (one-way-trapdoor permutation) and it is vulnerable against message-resend attacks.

However, there are two generic conversions, which are applicable to the McEliece PKC: One presented by Pointcheval [60] and the other by Fujisaki and Okamoto [21]. These conversions are valid for all encryption schemes, which are *partially trapdoor one-way* (PTOWF), i.e., the encryption is a function $f : X \times Y \to Z,\ (x, y) \mapsto z$ where it is impossible to recover x or y from their image z alone, but the knowledge of secret enables a partial inversion, i.e. finding x from z. Pointcheval [60] demonstrated how any PTOWF can be converted to a public-key cryptosystem that is indistinguishable against CCA2, while the conversion of Fujisaki and Okamoto is applicable to those schemes which are one-way encryptions (OWE), which includes PTOWF and OWTP.

We omit giving details on generic conversions, since they add a large amount of redundancy to the cipher texts. Instead we focus on the McEliece-specific conversions presented by Kobara and Imai, whose main concern is to decrease data overhead. As an example we present the "γ-conversion" based on Algorithm 2.1. For the ease of presentation we introduce the notations given in Table 5. The γ-conversion is summarized in Algorithm 5.4. It is assumed that $\text{length}(\mathbf{m}) \geq \log_2 \lfloor \binom{n}{t} \rfloor + k - \text{length}(const) - \text{length}(r)$.

For large messages Kobara and Imai achieve a reduction in data redundancy even below the values for the original McEliece PKC for large parameters. For example, for $m = 11, t = 70$ the message size is expanded by 655 bits instead of 770 in the original McEliece scheme. The security of the γ-conversion can be reduced to the one of the original scheme [37]:

Theorem 2. *Breaking indistinguishability in the CCA2 model using any of the conversions presented above, is as hard as breaking the original McEliece public key system.*

Symbol	Function
ℓ	$\lfloor \log_2 \binom{n}{t} \rfloor$.
H	Cryptographic secure hashing to a ℓ-bit string
R	Cryptographically secure pseudo random number generator from fixed length seeds
$E_{(\mathsf{G}^{\mathrm{pub}},t)}$	McEliece encryption function, taking as first argument the message to be encrypted and as second one the error vector: $E_{(\mathsf{G}^{\mathrm{pub}},t)}(\mathbf{m},\mathbf{z}) = \mathbf{c}$
$D_{(S,D_{\mathcal{G}},P)}$	McEliece decryption function: $D_{(S,D_{\mathcal{G}},P)}(c) = (\mathbf{m},\mathbf{z})$
$MSB_n(\mathbf{m})$	The n rightmost bits of $\mathbf{m}$.
$LSB_n(\mathbf{m})$	The n leftmost bits of $\mathbf{m}$.

Table 5. Notation for Algorithm 5.4.

Algorithm 5.4 Kobara-Imai's γ Conversion

- **Additional System Parameters:** length($\mathbf{r}$), the length of the random seed and a constant *const*.
- **Encryption $E^{\gamma}_{(\mathsf{G}^{\mathrm{pub}},t)}$:**

  Generate a random seed $\mathbf{r}$ of length length($\mathbf{r}$).
  Set
  $\mathbf{c}_1 = \mathrm{PRG}(\mathbf{r}) \oplus (\mathbf{m}, const)$, $\mathbf{c}_2 = \mathbf{r} \oplus H(\mathbf{c}_1)$,
  $\mathbf{c}_3 = LSB_{\ell+k}(\mathbf{c}_2,\mathbf{c}_1)$, $\mathbf{c}_4 = LSB_k(\mathbf{c}_3)$,
  $\mathbf{c}_5 = MSB_{\ell}(\mathbf{c}_3)$, $\mathbf{z} = \phi_{n,t}(\mathbf{c}_5)$
  if $\mathrm{length}(\mathbf{c}_2,\mathbf{c}_1) - \ell - k > 0$ **then**
    $\mathbf{c}_6 = MSB_{\mathrm{length}(\mathbf{c}_2,\mathbf{c}_1)-\ell-k}(\mathbf{c}_2,\mathbf{c}_1)$
    $\mathbf{c} = (\mathbf{c}_6, E_{(\mathsf{G}^{\mathrm{pub}},t)}(\mathbf{c}_4,\mathbf{z}))$
  else
    $\mathbf{c} = E_{(\mathsf{G}^{\mathrm{pub}},t)}(\mathbf{c}_4,\mathbf{z})$
- **Decryption $D^{\gamma}_{(S,\boldsymbol{D}_{\mathcal{G}},P)}$:**

  Set
  $\mathbf{c}_6 = MSB_{Len(\mathbf{c})-n}(\mathbf{c})$, $(\mathbf{c}_4,\mathbf{z}) = D_{(S,D_{\mathcal{G}},P)}(LSB_n(\mathbf{c}))$,
  $\mathbf{c}_5 = \phi^{-1}_{n,t}(\mathbf{z})$, $\mathbf{c}_2 = MSB_{\mathrm{length}(\mathbf{r})}(\mathbf{c}_6,\mathbf{c}_5,\mathbf{c}_4)$,
  $\mathbf{c}_1 = LSB_{\mathrm{length}(\mathbf{c})-\mathrm{length}(\mathbf{r})}(\mathbf{c}_6,\mathbf{c}_5,\mathbf{c}_4)$,
  $(\mathbf{m}, const') = (\mathbf{c}_1) \oplus \mathrm{PRG}(\mathbf{c}_2 \oplus H(\mathbf{c}_1))$
  if $const' = const$ **then**
    return m
  else
    reject c

Furthermore, all adaptive attacks become impossible, since relations among plaintexts do no longer result in relations among ciphertexts. Already the simple hashing of messages before encryption prevents this.

6 Annex

6.1 Algebraic coding theory

Hamming distance and linear codes.

Let $\mathbb{F}_q$ be a finite field. The Hamming distance between two words $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{F}_q^n$ is defined to be the number of coordinates in which $\mathbf{x}$ and $\mathbf{y}$ differ. The Hamming weight $\mathrm{wt}(\mathbf{x})$ of $\mathbf{x}$ is the number of non-zero coordinates of $\mathbf{x}$. A *code* is a non-empty subset of the Hamming space $\mathbb{F}_q^n$. A k-dimensional subspace of $\mathbb{F}_q^n$ is called a $[n, k]$ *linear code* over $\mathbb{F}_q$.

Generator and parity check matrices.

Let $\mathcal{C}$ denote an $[n, k]$ linear code over $\mathbb{F}_q$.

- A *generator matrix* G for $\mathcal{C}$ is a matrix over $\mathbb{F}_q$ such that $\mathcal{C} = \langle \mathsf{G} \rangle$, where $\langle \mathsf{G} \rangle$ denotes the vector space spanned by the rows of G. Usually, the rows of G are independent and the matrix is $k \times n$. A generator matrix G is said to be in *systematic form*, if its first k columns form the identity matrix.
- The *dual code* $\mathcal{C}^\perp$ of $\mathcal{C}$ is the orthogonal of $\mathcal{C}$ for the usual scalar product over $\mathbb{F}_q$. It is a $[n, n-k]$ linear code over $\mathbb{F}_q$.
- A *parity check matrix* H of $\mathcal{C}$ is a generator matrix of $\mathcal{C}^\perp$.

Minimum distance and weight.

Let $\mathcal{C}$ denote an $[n, k]$ linear code over $\mathbb{F}_q$. The *minimum distance* $d = \mathrm{dmin}(\mathcal{C})$ of $\mathcal{C}$ is the smallest Hamming distance between distinct codewords. For a linear code, it is equal to the *minimum weight*, the smallest non-zero weight of a codeword. We will speak of an $[n, k, d]$ code.

Decoder.

A decoder for $\mathcal{C}$ is a mapping $D_\mathcal{C} : \mathbb{F}_q^n \to \mathcal{C}$. It is t-error correcting if for all $\mathbf{e} \in \mathbb{F}_q^n$ and all $\mathbf{x} \in \mathcal{C}$

$$\mathrm{wt}(\mathbf{e}) \leq t \Rightarrow D_\mathcal{C}(\mathbf{x} + \mathbf{e}) = \mathbf{x}$$

For any $[n, k, d]$ linear code, there exist a t-error correcting decoder if and only if $t < d/2$.

Weight enumerator polynomial.

For a linear code $\mathcal{C}$, it is defined as

$$W(\mathcal{C})(X) = \sum_{\mathbf{c} \in \mathcal{C}} X^{\mathrm{wt}(\mathbf{c})} = \sum_{i=0}^{n} A_i X^i$$

where A_i is the number of codewords of Hamming weight i.

Support.

The *support* $\mathcal{I}$ of a code of length n is an ordered set of cardinality n used to index the coordinates. Typically $\mathcal{I} = \{1, \ldots, n\}$, but it is sometimes convenient to index the coordinates with another ordered set (in Goppa codes for instance). The support of a codeword is the subset of $\mathcal{I}$ containing its non-zero coordinates.

Puncturing.

Let $\mathcal{C}$ be an $[n, k]$ linear code of support $\mathcal{I}$, let G be a generator matrix of $\mathcal{C}$, and let $\mathcal{J}$ be a subset of $\mathcal{I}$.

- Punctured matrix: we denote by $\mathsf{G}_{.\mathcal{J}}$ the $k \times |\mathcal{J}|$ matrix obtained from G by keeping the columns indexed by $\mathcal{J}$.
- Punctured code: We denote by $\mathcal{C}_{.\mathcal{J}}$ the code obtained by retaining in all codeword of $\mathcal{C}$ the coordinates indexed by $\mathcal{J}$.

Note that $\mathcal{C}_{.\mathcal{J}} = \langle \mathsf{G}_{.\mathcal{J}} \rangle$ (i.e. the punctured matrix spans the corresponding punctured code).

Subcodes.

Any linear subspace of $\mathcal{C}$ is said to be a subcode of $\mathcal{C}$. If $\mathcal{C}$ is a code over $\mathbb{F}$ and $\mathbb{F}_{\mathrm{SUB}}$ is a subfield of $\mathbb{F}$, then the $\mathbb{F}_{\mathrm{SUB}}$-(subfield) subcode of $\mathcal{C}$ is the code consisting of all words of $\mathcal{C}$, which have only entries in $\mathbb{F}_{\mathrm{SUB}}$. A $\mathbb{F}_{\mathrm{SUB}}$-subfield subcode is a $\mathbb{F}_{\mathrm{SUB}}$-linear code. As codes are treated as vector spaces, we will often define them by the matrices related to the code.

6.2 GRS and Goppa codes

An important class of codes are the GRS codes, which are strongly related to the class of Goppa codes used by McEliece to define his cryptosystem. Thus, we briefly introduce them:

Definition 9. *A GRS code over $\mathbb{F}_{q^m}$ of length n with designed minimum Hamming distance $t+1$ is defined by two vectors $\mathbf{a}, \mathbf{z} \in \mathbb{F}_{q^m}^n$, where $a_i \neq a_j$ for $i \neq j$ and all $z_i \neq 0$. The canonical check matrix of the GRS code is of the form*

$$\mathsf{H}^\top = \begin{pmatrix} z_1 a_1^0 & z_1 a_1^1 & \cdots & z_1 a_1^{t-1} \\ z_2 a_2^0 & z_2 a_2^1 & \cdots & z_2 a_2^{t-1} \\ \vdots & & \ddots & \vdots \\ z_n a_n^0 & z_n a_n^1 & \cdots & z_n a_n^{t-1} \end{pmatrix} \in \mathbb{F}_{q^m}^{n \times t}. \tag{10}$$

The code with check matrix

$$\left[\begin{array}{c} \mathsf{H}^\top \\ \hline 0 \cdots 0\ 1 \end{array}\right]^\top$$

is called an extended GRS code.

The $\mathbb{F}_q$-subfield subcode of a GRS code is called an *alternant code* and has dimension $k \geq n - mt$. If for a GRS code, there exists a polynomial $g \in \mathbb{F}_{q^m}[X]$ of degree t, for which $g(a_i) = 1/z_i$, the polynomial is called *Goppa polynomial* and the $\mathbb{F}_q$-subfield subcode is called *Goppa code* (see e.g. [47] or [17]). An equivalent definition is the following:

Definition 10. *A* binary Goppa code $\mathcal{G}$ *over* $\mathbb{F}_{2^m}$ *is defined by a vector* $\mathbf{a} \in \mathbb{F}_{2^m}^n$*, where* $a_i \neq a_j$ *and the Goppa polynomial* $g(X) = \sum_{i=0}^{t} g_i X^i \in \mathbb{F}_{2^m}[X]$*.* $\mathcal{G}$ *is the set of all* $\mathbf{c} = (\mathbf{c}_0, \ldots, \mathbf{c}_{n-1}) \in \mathbb{F}_2^n$ *such that the identity*

$$S_{\mathbf{c}}(X) = -\sum_{i=0}^{n-1} \frac{\mathbf{c}_i}{g(a_i)} \frac{g(X) - g(a_i)}{X - a_i} \mod g(X) = 0 \tag{11}$$

holds in the polynomial ring $\mathbb{F}_{2^m}[X]$ *or equivalently if*

$$S_{\mathbf{c}}(X) \equiv \sum_{i=0}^{n-1} \frac{\mathbf{c}_i}{X - a_i} \equiv 0 \mod g(X). \tag{12}$$

Oftentimes, the vector $\mathbf{a}$ *is called* γ *or* $\mathbf{L}$ *and since* $\mathcal{G}$ *is defined in function of* $\mathbf{L}$ *and the Goppa polynomial we write:* $\mathcal{G} = \Gamma(\mathbf{L}, g)$*.*

If the Goppa polynomial is irreducible, then the Goppa code has minimum distance $2 \cdot t + 1$ and is called an *irreducible Goppa code*.

The coefficients of the syndrome polynomial $S_{\mathbf{c}}(X) = \sum_{i=0}^{t-1} s_i X^i$ of a vector $\mathbf{c}$ in a Goppa code may be computed via equation (13), where H is given in equation (10) with $z_i = 1/g(a_i)$.

$$\begin{pmatrix} s_0 & s_1 & \cdots & s_{t-1} \end{pmatrix} = \mathbf{c}\mathsf{H}^\top \begin{pmatrix} g_t & 0 & \cdots & 0 \\ g_{t-1} & g_t & \ddots & 0 \\ \vdots & & \ddots & \vdots \\ g_1 & g_2 & \cdots & g_t \end{pmatrix} \tag{13}$$

For GRS codes, as well as for Goppa codes, there exist algorithms for correcting errors of Hamming weight up to half of the minimum distance. Such algorithms take $\mathcal{O}(n^2)$ respectively $\mathcal{O}(n \cdot t \cdot m^2)$ binary operations, see e.g. [7, 58]. Here we present Patterson's algorithm for correcting errors in irreducible binary Goppa codes, where we follow the presentation in [17]: Let $\mathbf{m}$ be a codeword, $\mathbf{e} \in \mathbb{F}_2^n$ with $\mathrm{wt}(\mathbf{e}) \leq t$ an error vector, and $\mathbf{c} = \mathbf{m} \oplus \mathbf{e}$. Since $S_{\mathbf{m}}(X) \equiv 0 \mod g(X)$, we have

$$0 \neq S_{\mathbf{c}}(X) \equiv S_{\mathbf{e}}(X) \mod g(X).$$

We introduce the *error locator polynomial* $\sigma_{\mathbf{e}}(X)$ of $\mathbf{e}$ as

$$\sigma_{\mathbf{e}}(X) := \prod_{j \in \mathcal{T}_{\mathbf{e}}} (X - \gamma_j) \in \mathbb{F}_{2^m}[X],$$

where $\mathcal{T}_{\mathbf{e}}$ is the support of $\mathbf{e}$. From (12), it follows that

$$\sigma_{\mathbf{e}}(X) S_{\mathbf{e}}(X) \equiv \sigma'_{\mathbf{e}}(X) \mod g(X). \tag{14}$$

We split $\sigma_{\mathbf{e}}(X)$ in squares and non-squares:

$$\sigma_{\mathbf{e}}(X) = \alpha^2(X) + X\beta^2(X).$$

Since the characteristic of the field is 2, we have $\sigma'_{\mathbf{e}}(X) = \beta^2(X)$. Setting $T(X) = S_{\mathbf{e}}^{-1}(X)$ and multiply equation (14) by $T(X)$ we obtain

$$\beta^2(X)(X + T(X)) \equiv \alpha^2(X) \mod g(X) \tag{15}$$

Each element of $\mathbb{F}_{2^{mt}}$ has a unique square root. Let $\tau(X) \in \mathbb{F}_{2^m}[X]$ be the square root of $T(X) + X$, then

$$\beta(X)\tau(X) \equiv \alpha(X) \mod g(X).$$

The equation above can be solved: We have to determine $\alpha(X)$ and $\beta(X)$ of least degree, i.e. with $\deg(\alpha(X)) \leq \lfloor t/2 \rfloor$ and $\deg(\beta(X)) \leq \lfloor (t-1)/2 \rfloor$. Computing the inverse of $\tau(X)$ modulo $g(X)$ via the extended Euclidean algorithm and stopping it in mid-time gives the (unique) solution [33, 43, 47]. Finally, the zeroes of $\sigma_{\mathbf{e}}(X) = \alpha^2(X) + X\beta^2(X)$ can be determined, which reveals $\mathbf{e}$.

The runtime of the presented error correction algorithm may be estimated as follows. To compute the syndrome $S_{\mathbf{c}}(X)$ employing the check matrix H, we need at most $(n-k)n$ binary operations. To compute $T(X)$, we employ the extended Euclidean algorithm. This takes $\mathcal{O}\left(t^2m^2\right)$ binary operations, as the computations are modulo $g(X)$, a polynomial of degree t and coefficients of size m. Computing the square root of $T(X)+X$ takes $\mathcal{O}\left(t^2m^2\right)$ operation since it is a linear mapping on $\mathbb{F}_{2^m}[X]/g(X)$. The subsequently employed variant of the extended Euclidean algorithm takes $\mathcal{O}\left(t^2m^2\right)$ binary operations, too. These steps are fast in comparison to the last step to find all roots of the error locator polynomial. The latter can be performed in $n(tm^2 + tm)$ binary operations. Since $mt \geq (n-k)$, the error correction algorithm needs

$$\mathcal{O}\left(n \cdot t \cdot m^2\right)$$

binary operations. However, verifying, that an unique error locator polynomial exists requires only

$$\mathcal{O}\left(m^3t^2\right)$$

if the syndrome is already known.

6.3 Rank Distance

Not all codes are used with the Hamming metric. Here, we introduce a metric, which allows to correct "crisscross" errors in memory chip arrays or in magnetic tape recording, see [9, 41]:

Definition 11. *Let* $\mathbf{x} = (x_1, \cdots, x_n) \in \mathbb{F}_{q^m}^n$ *and* $b_1, \cdots, b_m$ *a basis of* $\mathbb{F}_{q^m}$ *over* $\mathbb{F}_q$. *We can write* $x_i = \sum_{j=1}^{m} x_{ij} b_j$ *for each* $i = 1, \cdots, n$ *with* $x_{ij} \in \mathbb{F}_q$. *The rank norm* $\|\cdot\|_q$ *is defined as follows:*

$$\|\mathbf{x}\|_q := \operatorname{rank}\left((x_{ij})_{1 \le i \le n,\ 1 \le j \le m}\right) .$$

There are more isometries preserving rank distance than Hamming distance since all invertible matrices over the base field are isometries for the rank metric. The Syndrome Decoding Problem seems to be much harder in rank metric than in Hamming metric. In [36] Ourivski and Johansson presented two algorithms which solve the general decoding problem in $\mathcal{O}\left((k + \frac{d-1}{2})^3 (\frac{d-1}{2})^3 \times q^{(d-3)(m-(d-1)/2)/2}\right)$, respectively $\mathcal{O}\left((m\frac{d-1}{2})^3 q^{(d-3)(k+1)/2}\right)$ operations over $\mathbb{F}_q$ for $[n, k, d]$ rank distance codes over $\mathbb{F}_{q^m}$.

Even if rank distance codes can not be used to build a PKC (compare §4.3), the introduction of the rank metric into cryptography is interesting and might be useful, as it could, e.g., allow to reduce the key sizes for Stern's identification scheme or strengthen the FSB hash. The interested reader may find more information on the aspects of rank metric in [23, 25, 36, 45].

References

1. Alabbadi, M. and Wicker, S.: A digital signature scheme based on linear error-correcting block codes. In *ASIACRYPT '94*, volume LNCS 917, pages 238–248 (Springer 1995).
2. Assmus, Jr, E.F. and Key, J.D.: Affine and projective planes. *Discrete Mathematics*, 83:161–187 (1990).
3. Augot, D., Finiasz, M., and N.Sendrier: A family of fast syndrome based cryptographic hash functions. In *Proc. of Mycrypt 2005*, volume 3715 of *LNCS*, pages 64–83 (2005).
4. Barg, A.: Complexity issues in coding theory. In V.S. Pless and W.C. Huffman, editors, *Handbook of Coding theory*, volume I, chapter 7, pages 649–754. North-Holland (1998).
5. Berger, T. and Loidreau, P.: Security of the Niederreiter form of the GPT public-key cryptosystem. In *IEEE International Symposium on Information Theory, Lausanne, Suisse*. IEEE (July 2002).
6. Berlekamp, E., McEliece, R., and van Tilborg, H.: On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386 (1978).
7. Berlekamp, E.: *Algebraic coding theory*. McGraw-Hill, New York (1968).

8. Berson, T.: Failure of the McEliece public-key cryptosystem under message-resend and related-message attack. In *Proceedings of CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 213–220. Springer Verlag (1997).
9. Blaum, M. and McEliece, R.J.: Coding protection for magnetic tapes: A generalization of the Patel - Hong code. *IEEE Transactions on Information Theory*, 31(5):690– (1985).
10. Camion, P. and Patarin, J.: The knapsack hash function proposed at Crypto'89 can be broken. In D.W. Davies, editor, *Advances in Cryptology - EUROCRYPT'91*, number 547 in LNCS, pages 39–53. Springer-Verlag (1991).
11. Canteaut, A. and Chabaud, F.: Improvements of the attacks on cryptosystems based on error-correcting codes. *Rapport interne du Departement Mathematiques et Informatique*, LIENS-95-21 (1995).
12. Canteaut, A. and Chabaud, F.: 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. *IEEETIT: IEEE Transactions on Information Theory*, 44 (1998).
13. Canteaut, A. and Sendrier, N.: Cryptanalysis of the original McEliece cryptosystem. In *Advances in Cryptology - ASIACRYPT '98 Proceedings*, pages 187–199. Springer-Verlag (1998).
14. Courtois, N., Finiasz, M., and N.Sendrier: How to achieve a McEliece-based digital signature scheme. In *Advances in Cryptology - ASIACRYPT 2001*, volume 2248, pages 157–174. Springer-Verlag (2001).
15. Cover, T.: Enumerative source encoding. *IEEE Transactions on Information Theory*, 19(1):73–77 (1973).
16. Dallot, L., Tillich, J., Otmani, A.: Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes (2008). CoRR, abs/0804.0409, available at http://arxiv.org/abs/0804.0409 (2008).
17. Engelbert, D., Overbeck, R., and Schmidt, A.: A summary of McEliece-type cryptosystems and their security. *Journal of Mathematical Cryptology*, 1(2):151–199 (2007).
18. Finiasz, M.: *Nouvelles constructions utilisant des codes correcteurs d'erreurs en cryptographie à clef publique.* Thèse de doctorat, École Polytechnique (2004).
19. Fischer, J.B. and Stern, J.: An eficient pseudo-random generator provably as secure as syndrome decoding. In U.M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *LNCS*, pages 245–255. Springer-Verlag (1996).
20. Fossorier, M., Imai, H., and Kobara, K.: Modeling bit flipping decoding based on non orthogonal check sums and application to iterative decoding attack of McEliece cryptosystem. In *Proc. of 2004 International Symposium on Information Theory and its Applications, Parma, Italy (ISITA'04)* (October 2004).
21. Fujisaki, E. and Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In *Proc. of CRYPTO*, volume 547 of *LNCS*, pages 535–554. Springer Verlag (1999).
22. Gabidulin, E.M. and Ourivski, A.V.: Column scrambler for the GPT cryptosystem. *Discrete Applied Mathematics*, 128(1):207–221 (2003).
23. Gabidulin, E.: Theory of codes with maximum rank distance. *Problems of Information Transmission*, 21, No. 1 (1985).
24. Gabidulin, E.: On public-key cryptosystems based on linear codes. In *Proc. of 4th IMA Conference on Cryptography and Coding 1993*, Codes and Ciphers. IMA Press (1995).

25. Gabidulin, E. and Loidreau, P.: Subfield subcodes of maximum-rank distance codes. In *Seventh International Workshop on Algebraic and Combinatorial Coding Theory*, volume 7 of *ACCT*, pages 151–156 (2000).
26. Gabidulin, E., Ourivski, A., Honary, B., and Ammar, B.: Reducible rank codes and their applications to cryptography. *IEEE Transactions on Information Theory*, 49(12):3289–3293 (2003).
27. Gabidulin, E., Paramonov, A., and Tretjakov, O.: Ideals over a non-commutative ring and their applications to cryptography. In *Proc. Eurocrypt '91*, volume 547 of *LNCS*. Springer Verlag (1991).
28. Gaborit, P.: Shorter keys for code based cryptography. In *Proc. of WCC 2005*, pages 81–90 (2005).
29. Gaborit, P. and Girault, M.: Lightweight code-based authentication and signature. In *Proc. of ISIT 2007* (2007).
30. Gaborit, P., Laudaroux, C., and Sendrier, N.: Synd: a very fast code-based cipher stream with a security reduction. In *IEEE Conference, ISIT'07*, pages 186–190. Nice, France (2007).
31. Gibson, K.: Equivalent Goppa codes and trapdoors to McEliece's public key cryptosystem. In D.W. Davies, editor, *Advances in Cryptology - Eurocrypt'91*, volume 547 of *LNCS*, pages 517–521. Springer Verlag (1991).
32. Harn, L. and Wang, D.C.: Cryptanalysis and modification of digital signature scheme based on error-correcting codes. *Electronics Letters*, 28(2):157–159 (1992).
33. Heise and Quattrocchi: *Informations- und Codierungstheorie*. Springer Berlin Heidelberg, 3 edition (1995).
34. Jabri, A.K.A.: A statistical decoding algorithm for general linear block codes. In *Cryptography and Coding 2001*, volume 2260 of *LNCS*, pages 1–8. Springer Verlag (2001).
35. Janwa, H. and Moreno, O.: McEliece public key cryptosystems using algebraic-geometric codes. *Designes, Codes and Cryptography*, 8:293–307 (1996).
36. Johansson, T. and Ourivski, A.: New technique for decoding codes in the rank metric and its cryptography applications. *Problems of Information Transmission*, 38, No. 3:237–246 (2002).
37. Kobara, K. and Imai, H.: Semantically secure McEliece public-key cryptosystems - conversions for McEliece PKC. In *Practice and Theory in Public Key Cryptography - PKC '01 Proceedings*. Springer Verlag (2001).
38. Kobara, K. and Imai, H.: On the one-wayness against chosen-plaintext attacks of the Loidreau's modified McEliece PKC. *IEEE Transactions on Information Theory*, 49, No. 12:3160–3168 (2003).
39. Lee, P. and Brickell, E.: An observation on the security of McEliece's public key cryptosystem. In *Advances in Cryptology-EUROCRYPT'88*, volume 330 of *LNCS*, pages 275–280. Springer Verlag (1989).
40. Leon, J.: A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, 34(5):1354–1359 (1988).
41. Levine, L. and Myers, W.: Semiconductor memory reliability with error detecting and correcting codes. *COMPUTER*, 9(10):43–50 (1976). ISSN 0018-9162.
42. Li, Y., Deng, R., and Wang, X.: the equivalence of McEliece's and Niederreiter's public-key cryptosystems. *IEEE Transactions on Information Theory*, Vol. 40, pp. 271-273 (1994).

43. Lidl, R. and Niederreiter, H.: *Introduction to finite fields and their applications.* Cambridge University Press, 2 edition (1986).
44. Loidreau, P.: Strengthening McEliece cryptosystem. In *Advances in Cryptology - ASIACRYPT '00 Proceedings*, pages 585–598. Springer Verlag (2000).
45. Loidreau, P. and Overbeck, R.: Decoding rank errors beyond the error-correction capability. In *Proc. of ACCT-10, Zvenigorod*, pages 168–190 (2006).
46. Loidreau, P. and Sendrier, N.: Weak keys in the McEliece public-key cryptosystem. *IEEE Transactions on Information Theory*, 47, No. 3:1207 –1211 (March 2001).
47. MacWilliams, F. and Sloane, N.: *The Theory of Error-Correctiong Codes.* North-Holland Amsterdam, 7 edition (1992).
48. McEliece, R.: A public key cryptosystem based on algebraic coding theory. *DSN progress report*, 42-44:114–116 (1978).
49. Minder, L.: *Cryptography based on error correcting codes.* Phd thesis, EPFL (2007).
50. Minder, L. and Shokrollahi, A.: Cryptanalysis of the Sidelnikov cryptosystem. In M. Naor, editor, *Advances in Cryptology - EUROCRYPT 2007*, number 4515 in LNCS, pages 347–360. Springer (2007).
51. Montpetit, A.: Note sur la notion d'équivalence entre deux codes linéaires. *Discrete Mathematics*, 65:177–185 (1987).
52. Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. *Probl. Control and Inform. Theory*, 15:19–34 (1986).
53. Overbeck, R.: Public key cryptography based on coding theory. Ph.D. Thesis, Available at http://elib.tu-darmstadt.de/diss/000823/.
54. Overbeck, R.: A new structural attack for GPT and variants. In *Proc. of Mycrypt 2005*, volume 3715 of *LNCS*, pages 50–63. Springer Verlag (2005).
55. Overbeck, R.: Statistical decoding revisited. In *Proc. of ACISP 2006*, volume 4058 of *LNCS*, pages 283–294. Springer Verlag (2006).
56. Overbeck, R.: Recognizing the structure of permuted reducible codes. In *Proc. of WCC 2007*, pages 269–276 (2007).
57. Overbeck, R.: Structural attacks for public key cryptosystems based on Gabidulin codes. *Journal of Cryptology*, 21(2):280–301 (2008).
58. Patterson, N.: Algebraic decoding of Goppa codes. *IEEE Trans. Info.Theory*, 21:203–207 (1975).
59. Petrank, E. and Roth, R.M.: Is code equivalence easy to decide? *IEEE Trans. on IT*, 43(5):1602–1604 (1997).
60. Pointcheval, D.: Chosen-ciphertext security for any one-way cryptosystem. In *Proc. of PKC*, volume 1751 of *LNCS*, pages 129–146. Springer Verlag (2000).
61. Ramabadran, T.V.: A coding scheme for m-out-of-n codes. *IEEE Transactions on Communications*, 38(8):1156–1163 (1990).
62. Schalkwijk, J.P.M.: An algorithm for source coding. *IEEE Transactions on Information Theory*, 18(3):395–399 (1972).
63. Sendrier, N.: Efficient generation of binary words of given weight. In C. Boyd, editor, *Cryptography and Coding ; proceedings of the 5th IMA conference*, number 1025 in LNCS, pages 184–187. Springer-Verlag (1995).
64. Sendrier, N.: On the concatenated structure of a linear code. *AAECC*, 9(3):221–242 (1998).
65. Sendrier, N.: *Cryptosystèmes à clé publique basés sur les codes correcteurs d'erreurs.* Mémoire d'habilitation à diriger des recherches, Université Paris 6 (2002).

66. Sendrier, N.: On the security of the McEliece public-key cryptosystem. In M. Blaum, P. Farrell, and H. van Tilborg, editors, *Proceedings of Workshop honoring Prof. Bob McEliece on his 60th birthday*, pages 141–163. Kluwer (2002).
67. Sendrier, N.: Encoding information into constant weight words. In *IEEE Conference, ISIT'2005*, pages 435–438. Adelaide, Australia (2005).
68. Sendrier, N.: Finding the permutation between equivalent linear codes: the support splitting algorithm. *IEEE Transactions on Information Theory*, 46:1193–1203 (Jul 2000).
69. Sendrier, N.: On the dimension of the hull. *SIAM Journal on Discrete Mathematics*, 10(2):282–293 (May 1997).
70. Sidelnikov, V.: A public-key cryptosystem based on binary Reed-Muller codes. *Discrete Mathematics and Applications*, 4 No. 3 (1994).
71. Sidelnikov, V. and Shestakov, S.: On insecurity of cryptosystems based on generalized Reed-Solomon codes. *Discrete Mathematics and Applications*, 2, No. 4:439–444 (1992).
72. Stern, J.: A method for finding codewords of small weight. *Coding Theory and Applications*, 388:106–133 (1989).
73. Stern, J.: A new identification scheme based on syndrome decoding. In *Advances in Cryptology - CRYPTO'93*, volume 773 of *LNCS*. Springer Verlag (1994).
74. Stern, J.: Can one design a signature scheme based on error-correcting codes. In *ASIACRYPT '94*, volume 917 of *LNCS*, pages 424–426 (1995).
75. van Tilburg, J.: On the McEliece cryptosystem. In S. Goldwasser, editor, *Advances in Cryptology - CRYPTO'88*, number 403 in LNCS, pages 119–131. Springer-Verlag (1990).
76. Véron, P.: Improved identification schemes based on error-correcting codes. *Appl. Algebra Eng. Commun. Comput.*, 8(1):57–69 (1996).
77. Wagner, D.: A generalized birthday problem. In M. Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer (2002). ISBN 3-540-44050-X.
78. Wieschebrink, C.: An attack on a modified Niederreiter encryption scheme. In *Public Key Cryptography*, volume 3958 of *LNCS*, pages 14–26 (2006).
79. Xinmei, W.: Digital signature scheme based on error-correcting codes. *Electronics Letters*, 26(13):898–899 (1990).