Signing with Codes

by

Zuzana Masárová

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2014

AUTHOR'S DECLARATION

Abstract

Code-based cryptography is an area of classical cryptography in which cryptographic primitives rely on hard problems and trapdoor functions related to linear error-correcting codes. Since its inception in 1978, the area has produced the McEliece and the Niederreiter cryptosystems, multiple digital signature schemes, identification schemes and code-based hash functions. All of these are believed to be resistant to attacks by quantum computers. Hence, code-based cryptography represents a post-quantum alternative to the widespread number-theoretic systems.

This thesis summarises recent developments in the field of code-based cryptography, with a particular emphasis on code-based signature schemes. After a brief introduction and analysis of the McEliece and the Niederreiter cryptosystems, we discuss the currently unresolved issue of constructing a practical, yet provably secure signature scheme. A detailed analysis is provided for the Courtois, Finiasz and Sendrier signature scheme, along with the mCFS and parallel CFS variations. Finally, we discuss a recent proposal by Preetha et al. that attempts to solve the issue of provable security, currently failing in the CFS scheme case, by randomizing the public key construct. We conclude that, while the proposal is not yet practical, it represents an important advancement in the search for an ideal code-based signature scheme.

Acknowledgements

I would like to thank my academic advisor, Prof. Edlyn Teske-Wilson, for a continuing support during my graduate studies and, in particular, for choosing the topic of code-based cryptography, reading through my early drafts and giving me useful hints and comments while writing. I would also like to thank Prof. Alfred Menezes and Prof. David Jao for reading my thesis and giving me very prompt and useful comments and suggestions. Finally, I would like to thank Dalimil Mazáč for being able to occasionally fight LaTeX on my behalf.

Table of Contents

Chapter 1

Introduction

Most of the cryptographic schemes deployed nowadays in practice are number-theoretic in nature. RSA, ECC, ElGamal, and most of the other popular choices are based on factorization or the discrete logarithm problem. These schemes have been carefully designed and working well for the last three decades. However, with the development of quantum computers the situation may soon change, as the important problems behind the schemes are vulnerable to quantum attacks. In 1994, Shor published a quantum algorithm capable of solving both factorization and the discrete logarithm problem in polynomial time. Such an achievement put into practice could, potentially, ruin information security as we know it today. Hence, in parallel with the research in quantum engineering, there runs a cryptographic thread of research focusing on the development of new, practical and quantum-resistant cryptographic schemes.

Code-based cryptography represents one such alternative. As a subfield of classical cryptography to which none of the known quantum attacks applies, it is viewed as a secure possibility for the post-quantum world.

The field was established in 1978 when McEliece [30] designed the first code-based public-key cryptosystem and envisioned a wide use of coding theory in cryptography. Since then, many code-based cryptographic primitives have been proposed - another cryptosystem, digital signatures, an identification scheme and hash functions (see [34, 16, 1, 11] for surveys). A particularly active research period has been witnessed over the last 15 years, when the known schemes were modified for better efficiency and security, new variants were proposed and a variety of new code families tried out (see, e.g. [27, 34] for surveys).

Now, we start with a basic but frequent question: how does one apply coding theory, which is otherwise independent, to cryptography? As always with public-key cryptography, one identifies a hard problem and a suitable trapdoor function allowing the entitled persons to solve the problem. The trapdoor role is played by the linear error-correcting codes. These are specially designed families of codes that, after transmission of a message, can in polynomial time identify which bits have been transmitted incorrectly, correct them and recover the original message (see, e.g. [29]). On the other hand, there are random codes in which error correction turns out to be NP-hard [5]. Then, roughly speaking, one builds a cryptographic scheme by linking an error-correcting code and a random(-looking) code through a secret function. A ciphertext is an element of the random(-looking) code with some purposely added errors so that an eavesdropper cannot correct them. A legitimate receiver, however, maps the ciphertext into the error-correcting code and easily recovers the plaintext there. Specifying the exact details of a scheme, the choice of a suitable error-correcting code, the mapping function, etc. so that the resulting schemes are both efficient and secure is a subject of ongoing research in code-based cryptography.

In general, code-based cryptography is not only post-quantum, but, as compared to mainstream number-theoretic systems, it also offers other advantages. Unlike in the case of factorization and the discrete logarithm, the hard problems behind code-based schemes are provably hard. The problem of correcting errors in a random code mentioned above has been formalized via the General decoding problem 2.3.2, or, equivalently, the Syndrome decoding problem 2.3.3. Both of these problems have been proven NP-hard [5, 44]. Hence, if NP$\neq$P, then the code-based systems promise good security to start with.

Another advantage of the code-based schemes is the efficiency of encryption (and equivalent operations, such as signature verification). These usually consist of a simple matrix-vector multiplication and are, thus, several orders of magnitude faster than the widespread RSA or ECC schemes.

On the other hand, there are some reasons for why code-based cryptography is not nearly as popular as the number-theoretic schemes. At the time of its inception in 1978, the public keys of the proposed cryptosystems were too large to be practical. With the development of computing platforms and fast networks over the years, this is a more manageable problem nowadays, although the keys are still relatively big: a 100-bit security level typically corresponds to public keys of size $\sim$ 100KB [43]. This is easily manageable by desktop computers and there do exist some implementations of code-based schemes on small devices, too (see [37, 41]). However, there is still ongoing research trying to make the code-based public key parameters comparable in size with their RSA and ECC counterparts.

Another issue has been the question of establishing an efficient and provably secure code-based signature scheme. Already McEliece [30] predicted the difficulties with turning his public-key cryptosystem into a signature scheme. The problem, in general, is that code-based cryptosystems are not easily invertible: if one picks a random element of the random code, this element is, highly probably, not decryptable. It took over two decades until Courtois et al. [14] realised that one needs to modify the parameters of the codes used in the cryptosystems to make them at least "practically invertible." Still, the signing time of the resulting signature scheme is rather large and even small increases in the code parameters may render the scheme impractical.

The aim of this thesis is to summarise recent developments in the field of code-based cryptography, with a particular emphasis on code-based signature schemes.

In Section 2, we review some coding theory. We prepare the background for code-based cryptography by introducing error-correcting codes, and, in particular, the binary irreducible Goppa codes with their polynomial-time error correcting algorithm. We also discuss some hard coding theory problems to which the security of the code-based schemes is usually reduced. These are the Goppa code indistinguishability assumption 2.3.7, the General decoding problem 2.3.2, the Syndrome decoding problem 2.3.3 and their variations.

In Chapter 3, we give a brief overview of code-based cryptography in general. We then introduce the two main cryptosystems, namely the McEliece and the Niederreiter cryptosystem, both using the binary Goppa codes. We discuss in detail their efficiency and security. In particular, cryptosystems' time and memory requirements are outlined and suitable parameter choices cited. On the security side, we explain the reduction of cryptosystems' security to hard problems from Chapter 2 and discuss the most efficient attacks against the cryptosystems - the information set decoding techniques. Finally, we review the conversions for McEliece and Niederreiter that are needed if the cryptosystems are to achieve IND-CCA2 security.

Chapter 4 addresses the main topic of the thesis: code-based signatures. After giving a brief overview of the area and formally defining a digital signature scheme with the desired security notions, we turn to the first practical signature scheme, namely, the CFS scheme by Courtouis et al. [14]. We explain the clever approach by which the authors managed to "practically invert" the Niederreiter cryptosystem, as well as the reasons why signing is not possible by inverting the McEliece cryptosystem. We discuss the security of the CFS scheme by identifying the relevant hard problems behind the scheme. Further, the main attacks on the scheme are discussed, in particular, the Bleichenbacher attack (described in [20]) based on the generalized birthday paradox. This attack neccesitated a slight increase in the CFS originally proposed parameters [20], making the signing time of the CFS scheme almost impractical. We discuss a remedy proposed by Finiasz [19] who modifies the CFS scheme into a so-called CFS-Parallel scheme, capable of securely using smaller parameters.

The memory and time requirements of the CFS signatures are discussed and possible trade-offs in the performance outlined. We also compare individual parameters of the CFS scheme to their counterparts in the Niederreiter cryptosystem. In general, we find that verification and encryption is equally fast and the size of public key is equally large in the signature scheme as in the cryptosystem. However, the CFS scheme's signing times are much longer than are the corresponding decryption times of the ciphertexts in the Niederreiter cryptosystem.

An open question of the field, namely, provable security of code-based signatures is discussed. We describe a modification mCFS of the CFS scheme by Dallot [15] that enables an EUF-CMA security proof in the random oracle model, assuming the hardness of the Bounded syndrome decoding problem 2.3.6 and Goppa code indistinguishability 2.3.7. Unfortunately, due to recent developments in coding theory, namely, showing that the Goppa code indistinguishability assumption does not hold for the code parameters used in the signature scheme [17], the above security proof has been invalidated. We discuss a very recent proposal of yet another modification of the CFS scheme by Preetha et al. [38]. The authors of [38] formulate a weaker indistinguishability assumption and modify the public code of the signature scheme so that the distinguishing methods of [17] no longer apply. The resulting signature scheme is claimed to be as efficient as the original CFS scheme, while also provably secure [38].

Finally, we give a short survey of other schemes related to code-based signing. In particular, we mention the KKS scheme, signatures obtained from identification schemes, ring signatures, threshold ring signatures, blind signatures and identity-based signatures.

Lastly, we round up the thesis with a brief conclusion in Chapter 5.

Chapter 2

Some coding theory

The main ingredients needed in code-based cryptography are linear error-correcting codes and some code-related hard problems, based on which code-based cryptosystems and signature schemes can be created.

This chapter therefore gives an overview of the relevant parts of coding theory. In particular, we start by defining and discussing linear codes and some related basic concepts. We then focus on a particular class of linear codes, namely, binary Goppa codes, as these codes play an important role in the schemes discussed in Chapters 3 and 4. Finally, we list some code-related problems that are believed to be hard.

2.1 Basic concepts

In what follows, let $\mathbb{F}_q$ be a finite field with q elements, where q is a prime power.

Definition 2.1.1. *A q-*ary linear code *$\mathcal{C}$ is a subspace of a finite-dimensional vector space V over $\mathbb{F}_q$. If the dimension of V is n, then $\mathcal{C} \subseteq \mathbb{F}_q^n$ and we say that the code $\mathcal{C}$ has length n. $\mathcal{C}$ is a k-*dimensional *code if the dimension of $\mathcal{C}$ is k. The elements of $\mathcal{C}$ are called* codewords*.*

Definition 2.1.2. *A* generator matrix *G for the code $\mathcal{C}$ with dimension k and length n is a $k \times n$ matrix whose rows form a basis for $\mathcal{C}$. G is said to be in a* systematic form *if $G = (I_k|Q)$ where I_k is the $k \times k$ identity matrix.*

Let $\mathcal{C}^\perp$ denote the $(n-k)$-dimensional subspace of V that is dual to $\mathcal{C}$. Then $\mathcal{C}^\perp$ is said to be the dual code *to $\mathcal{C}$. A* parity-check matrix *H for the code $\mathcal{C}$ is an $(n-k) \times n$ matrix whose rows form a basis for $\mathcal{C}^\perp$. Finally, given any vector $v \in \mathbb{F}_q^n$, the* syndrome *of v is the vector $vH^T \in \mathbb{F}_q^{n-k}$ and $\mathbb{F}_q^{n-k}$ is called the* syndrome space*.*

Note that a parity-check matrix for the code $\mathcal{C}$ is a generator matrix for $\mathcal{C}^\perp$ and vice versa. Also note that an important property of the parity-check matrix H for $\mathcal{C}$ is that, for any $v \in \mathbb{F}_q^n$,

$$v \in \mathcal{C} \text{ iff } Hv^T = 0,$$

and so the syndrome of $v \in \mathbb{F}_q^n$ is zero if and only if $v \in \mathcal{C}$. The point is that with a parity-check matrix one can easily check whether a given vector in $\mathbb{F}_q^n$ is a codeword in the corresponding code.

Definition 2.1.3. *The* weight *of a vector $v \in \mathbb{F}_q^n$, denoted $wt(v)$, is the number of non-zero components in v. A* (Hamming) distance *$d(v, w)$ between two vectors $v, w \in \mathbb{F}_q^n$ is the number of components at which v and w differ. The* minimum distance *$d_{\min}^{\mathcal{C}}$ of the code $\mathcal{C}$ is the smallest distance that there is between two codewords in $\mathcal{C}$. We denote a linear code $\mathcal{C}$ with length n, dimension k and minimum distance d an $[n, k, d]$-code.*

A nice property of linear codes is that the codewords are 'evenly spaced' in the vector space, meaning, that the minimum distance of the code is the distance between 0 and a codeword with the smallest non-zero weight in $\mathcal{C}$, i.e.

$$d_{\min}^{\mathcal{C}} = \min_{\substack{a \in \mathcal{C} \\ a \neq 0}} \{wt(a)\}.$$

Minimum distance greater than one gives code error-detecting and error-correcting capabilities. The main idea is as follows. There is a word space $\mathbb{F}_q^k$ in which each $x \in \mathbb{F}_q^k$ represents a valid word. A sender A wishes to transmit a particular word x to a receiver B. If the transmission channel is noisy, i.e. the components of $x \in \mathbb{F}_q^k$ may not all be transmitted correctly, B may receive a word $y \in \mathbb{F}_q^k$ different from x. However, since y is also a valid word, B has no means of finding out whether y is the word that was originally sent or whether the corruption occured.

The problem is solved by bijectively mapping the word space into a linear $[n, k, d]$-code $\mathcal{C}$ with $d > 1$. Let $g : \mathbb{F}_q^k \to \mathbb{F}_q^n$ be defined by $g(a) = aG$, $\forall a \in \mathbb{F}_q^k$, where G is the generator matrix of $\mathcal{C}$. Now, instead of sending x, A first *encodes* the word x to obtain a codeword xG and then sends xG along the noisy channel. Assume that B receives a vector $xG + e \in \mathbb{F}_q^n$, where $e \in \mathbb{F}_q^n$ is the corresponding error vector, i.e. the component e_i of e, for $1 \leq i \leq n$, is zero if the ith component of xG was transmitted correctly, and contains the respective error if $(xG)_i$ was corrupted. The received vector $xG + e$ is distance $\mathrm{wt}(e)$ from the codeword xG. Notice that if $\mathrm{wt}(e) < d$ then $xG + e \notin \mathcal{C}$ and the receiver B immediately *detects* the corruption.

If, moreover, $\mathrm{wt}(e) < \lceil \frac{d}{2} \rceil$ and there exists an efficient way of finding the closest codeword in $\mathcal{C}$, then B is able to *correct* the error and recover the originally sent codeword xG by the principle below. Finally, B recovers the word $x \in \mathbb{F}_q^k$ by calculating $g^{-1}(xG)$.

The process of relating a codeword to a received $\mathbb{F}_q^n$-vector is referred to as *decoding*.

The maximum likelihood decoding principle 2.1.4. *For a code $\mathcal{C} \subset \mathbb{F}_q^n$ and a received vector $v \in \mathbb{F}_q^n$, decode v as a codeword $c \in \mathcal{C}$, where*

$$d(v, c) = \min_{a \in \mathcal{C}} \{d(v, a)\}.$$

In other words, c is a closest codeword to v. We write $c = \mathrm{Dec}_{\mathcal{C}}^{\mathcal{D}}(v)$, where $\mathcal{D}$ is a specific decoding algorithm used.

The principle always yields a correct unique decoding of a vector $v \in \mathbb{F}_q^n$ if, for the number t of incorrectly transmitted components of a sent codeword $c \in \mathcal{C}$, we have

$$t < \left\lceil \frac{d_{\min}^{\mathcal{C}}}{2} \right\rceil.$$

On the other hand, if $t > \lfloor \frac{d_{\min}^{\mathcal{C}}}{2} \rfloor$, the principle, in general, fails to decode correctly (although it may still work for vectors that are $> \lfloor \frac{d_{\min}^{\mathcal{C}}}{2} \rfloor$ away from each codeword) and if $t = \frac{d_{\min}^{\mathcal{C}}}{2}$, then the decoding is not unique. In what follows, we always assume to use the Maximum likelihood decoding principle and decode a received vector as a closest codeword in a given code.

Now, the main issue that matters in the decoding process is the following: given a vector $v \in \mathbb{F}_q^n$, how hard is it to find a codeword closest to v? It turns out, and we will see in Section 2.3, that, in general, this is a hard problem. But there do exist specifically designed families of error-correcting codes that can find the closest codeword in polynomial time. In particular, we say:

Definition 2.1.5. *A code $\mathcal{C} \subset \mathbb{F}_q^n$ is* t-error-correcting, *if there exists a polynomial-time decoding algorithm $\mathcal{D}$ such that for all pairs of a sent codeword $c \in \mathcal{C}$ and received vector $v \in \mathbb{F}_q^n$, whenever $d(c, v) \leq t$, then $\mathrm{Dec}_{\mathcal{C}}^{\mathcal{D}}(v) = c$.*

Note that a t-error-correcting code $\mathcal{C}$ must have the minimum distance $d_{\min}^{\mathcal{C}} > 2t$.

Known examples of error-correcting codes are the Hamming, Reed-Muller, BCH, Reed-Solomon, algebraic geometry, LDPC, convolutional and alternant codes with a subclass of binary Goppa codes, among others. For a review of error-correcting codes see [29].

In Chapters 3 and 4 we study code-based cryptography in detail; for now we only remark that the fact that the decoding process is hard in some codes, but easy in others, lies at the very centre of code-based cryptography. It is exactly this property of decoding that makes it possible to construct cryptosystems and signature schemes.

We conclude the section by definitions needed later on.

Definition 2.1.6. *Two $[n, k, d]$-codes $\mathcal{C}$ and $\mathcal{C}'$ are* permutation equivalent *if there exists a permutation π of n elements such that*

$$\mathcal{C}' = \{(c_{\pi(1)}, \ldots, c_{\pi(n)}) | (c_1, \ldots, c_n) \in \mathcal{C}\}.$$

Definition 2.1.7. *The* (information) rate *of an $[n, k, d]$-code $\mathcal{C}$ is the ratio $\frac{k}{n}$; k is the number of* information characters *and $n-k$ is the number of* parity check characters.

The rate says that in a codeword of length n, only k components carry information. The rest of the components were added to enable the error-correcting capabilities of the code. In general, there is a trade-off between the rate and the number of errors that a code is able to correct.

2.2 Irreducible binary Goppa codes

We now introduce a specific class of linear error-correcting codes - the Goppa codes. These codes were first defined by V. D. Goppa [22].

Definition 2.2.1. *Fix a field $\mathbb{F}_{q^m}$. Pick a polynomial $g(X) = g_0 + g_1X + ... + g_tX^t \in \mathbb{F}_{q^m}[X]$ of degree t and a set of n pairwise distinct elements $L = \{\alpha_1, ..., \alpha_n\} \subseteq \mathbb{F}_{q^m}$ such that $g(\alpha_i) \neq 0$ for all $1 \leq i \leq n$. The* degree-t (classical) Goppa code $\Gamma_q(L, g)$ *is defined to be the set of all $\mathbf{c} = (c_1, ..., c_n) \in \mathbb{F}_q^n$ such that*

$$\sum_{i=1}^{n} \frac{c_i}{X - \alpha_i} \equiv 0 \bmod g(X),$$

or, equivalently, $\sum_{i=1}^{n} \frac{c_i}{X-\alpha_i} = 0$ in $\mathbb{F}_{q^m}[X]/g(X)$.

Note that in the ring $\mathbb{F}_{q^m}[X]/g(X)$, the polynomial $X-\alpha_i$ has an inverse for all $1 \leq i \leq n$, since $X-\alpha_i$ is, by definition, coprime with $g(X)$. Hence, the Goppa code is well-defined.

In the above, the polynomial $g(X)$ is called the *Goppa polynomial* and the set L a *support*. In cryptographic applications, the support often consists of all elements of $\mathbb{F}_{q^m}$ that are not roots of $g(X)$.

The code $\Gamma_q(L, g)$ is q-ary, while L and the coefficients of $g(X)$ are from $\mathbb{F}_{q^m}$. The code has length n and it can also be shown to have dimension $k \geq n - mt$ and minimum distance $d_{\min}^{\Gamma_q(L,g)} \geq t+1$. Notice that in order to obtain the above length and to guarantee nonzero dimension, the parameters of the code must satisfy $n \leq q^m$ and $n \geq mt$.

We now derive the parity-check matrix, mimicking the approach in [29] and [16] with the notation above. A generator matrix for $\Gamma_q(L, g)$ can then be found by computing the orthogonal subspace by linear algebra. Notice that in the ring $\mathbb{F}_{q^m}[X]/g(X)$, for all i we have

$$-\frac{g(X)-g(\alpha_i)}{g(\alpha_i)} \equiv -g^{-1}(\alpha_i) \cdot g(X) + 1 \equiv 1 \bmod g(X).$$

Then for any vector $c \in \mathbb{F}_q^n$:

$$c \in \Gamma_q(L, g)$$

if and only if

$$\sum_{i=1}^{n} \frac{c_i}{X-\alpha_i} \equiv 0 \bmod g(X)$$

if and only if

$$-\sum_{i=1}^{n} \frac{c_i}{g(\alpha_i)} \cdot \frac{g(X)-g(\alpha_i)}{X-\alpha_i} \equiv 0 \bmod g(X).$$

Definition 2.2.2. *The expression* $-\sum_{i=1}^{n} \frac{c_i}{g(\alpha_i)} \cdot \frac{g(X)-g(\alpha_i)}{X-\alpha_i} \in \mathbb{F}_{q^m}[X]$ *is known as the* (Goppa) syndrome *of a vector* $c \in \mathbb{F}_q^n$*, denoted* $S_c(X)$.

The terminology becomes clear once we see how the parity-check matrix is created from the syndrome.

Proposition 2.2.3. *For a Goppa code* $\Gamma_q(L, g)$ *and any* $c \in \mathbb{F}_q^n$, $c \in \Gamma_q(L, g)$ *if and only if* $S_c(X) = 0$ *as a polynomial in* $\mathbb{F}_{q^m}[X]$.

Proof. For all $1 \leq i \leq n$, α_i is a root of both $g(X) - g(\alpha_i)$ and $X - \alpha_i$. Then, since the degree of $g(X)$ is t, the degree of each term in $S_c(X)$ is $t-1$. Hence, the degree of $S_c(X)$ in $\mathbb{F}_{q^m}[X]$ is $\leq t-1$. We saw above that $c \in \Gamma_q(L, g)$ iff $S_c(X) \equiv 0 \bmod g(X)$. The latter holds if and only if $S_c(X)$ is a zero polynomial in $\mathbb{F}_{q^m}[X]$.

□

Thus, a vector c is a codeword iff the coefficient of each of $X^0, X^1, \ldots, X^{t-1}$ in $S_c(X)$ is zero. After expanding

$$S_c(X) = \sum_{i=1}^{n} \frac{c_i}{g(\alpha_i)} \frac{g(X) - g(\alpha_i)}{X - \alpha_i} =$$

$$= \sum_{i=1}^{n} \frac{c_i}{g(\alpha_i)} \cdot \{g_1 + g_2(X + \alpha_i) + g_3(X^2 + X\alpha_i + \alpha_i^2) + \cdots + g_t(X^{t-1} + \cdots + \alpha_i^{t-1})\},$$

we get the following conditions for the coefficients of X^j, $0 \leq j \leq t-1$:

$$\sum_{i=1}^{n} \frac{c_i}{g(\alpha_i)} \cdot (g_{j+1} + g_{j+2}\alpha_i + g_{j+3}\alpha_i^2 + \cdots + g_t \alpha_i^{t-1-j}) = 0.$$

Reformulating the conditions, it can easily be seen that $c \in \Gamma_q(L, g)$ if and only if $Hc^T = 0$, where

$$H = \begin{pmatrix} \frac{g_t}{g(\alpha_1)} & \frac{g_t}{g(\alpha_2)} & \cdots & \frac{g_t}{g(\alpha_n)} \\ \frac{g_{t-1}+g_t\alpha_1}{g(\alpha_1)} & \frac{g_{t-1}+g_t\alpha_2}{g(\alpha_2)} & \cdots & \frac{g_{t-1}+g_t\alpha_n}{g(\alpha_n)} \\ \vdots & \vdots & \ddots & \cdots \\ \frac{g_1+g_2\alpha_1+\cdots+g_t\alpha_1^{t-1}}{g(\alpha_1)} & \frac{g_1+g_2\alpha_2+\cdots+g_t\alpha_2^{t-1}}{g(\alpha_2)} & \cdots & \frac{g_1+g_2\alpha_n+\cdots+g_t\alpha_n^{t-1}}{g(\alpha_n)} \end{pmatrix} \tag{2.1}$$

$$= \begin{pmatrix} g_t & 0 & 0 & \cdots & 0 \\ g_{t-1} & g_t & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & g_3 & \cdots & g_t \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{t-1} & \alpha_2^{t-1} & \cdots & \alpha_n^{t-1} \end{pmatrix} \cdot$$

$$\cdot \begin{pmatrix} g^{-1}(\alpha_1) & 0 & \cdots & 0 \\ 0 & g^{-1}(\alpha_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & g^{-1}(\alpha_n) \end{pmatrix} =: XYZ.$$

Since X, Y and Z each have full rank, also H has full rank, and so, H is a parity-check matrix for the code $\Gamma_q(L, g)$. If we wish to write the parity-check matrix in terms of elements of $\mathbb{F}_q$, we express $\mathbb{F}_{q^m}$ as an m-dimensional vector space over $\mathbb{F}_q$, i.e. each element of $\mathbb{F}_{q^m}$ is an $\mathbb{F}_q^m$-vector, and the parity-check matrix has dimension $mt \times n$.

As further mentioned in [29], since X is non-singular, the rows of the matrices H and YZ generate, in fact, the same subspace of $\mathbb{F}_q^n$. Hence, YZ is another parity-check matrix for $\Gamma_q(L, g)$. As a product of a Vandermonde and a diagonal matrix, YZ has the form of a parity-check matrix for alternant codes (see [29] for the definition), showing that the Goppa codes are a subclass of the alternant codes and that any decoding algorithm for alternant codes can also be applied to Goppa codes. In practice, however, all alternant code decoders perform on Goppa codes poorer than (the general version of) the so-called Patterson algorithm devised especially for Goppa codes.

Finally, let us remark that an important property of the Goppa codes, as we see in Chapters 3 and 4, is that they resemble random linear codes in many characteristics. For example, Faugere et. al. [17] state that both Goppa and random codes have trivial permutation group, meet the so-called Gilbert-Varshamov bound and, for most Goppa codes there is currently no way of distinguishing their parity-check matrices from the matrices of random codes. The general theory of Goppa codes is explained in detail in [29, 35, 4].

We now restrict our attention to a subclass of Goppa codes that is of the most interest to code-based cryptography. These are the *irreducible binary Goppa codes*, i.e. the Goppa codes with $q = 2$ and with polynomial $g(X)$ irreducible in $\mathbb{F}_{2^m}[X]$.

While inheriting the properties discussed above, this subclass has some notable advantages over the general family of Goppa codes (and other kinds of codes as well). Not only it is better suited for computing, but, more importantly, it has a very good error-correction capability while maintaining a relatively high information rate [6].

The main aim of this section, is to prove that the irreducible binary Goppa codes have minimum distance $\mathrm{d}_{\min}^{\Gamma_2(L,g)} \geq 2t+1$ and to show that there exists an easily implementable version of Patterson decoding algorithm for binary Goppa codes correcting the full t errors in polynomial time. The proofs are adapted from [6, 16]. Finally, note also that the recent work of Bernstein [6] develops a method that enables one to correct even more errors than what the correcting capacity of the Goppa code is. This method, on top of the Patterson algorithm, also necessitates introduction of decoding lists containing multiple candidates for a decoding of a given vector and a delicate handling of these candidates. Therefore, we do not describe this method below. An interested reader is directed to [6].

Facts 2.2.4. *In a field $\mathbb{F}_{2^m}$, there is the Frobenius automorphism $\mathbb{F}_{2^m} \to \mathbb{F}_{2^m}$: $x \mapsto x^2$. Hence, every element of $\mathbb{F}_{2^m}$ has a unique square root.*

Also, since $\mathbb{F}_{2^m}$ has characteristic 2, a polynomial $f(X) \in \mathbb{F}_{2^m}[X]$ is a perfect square if and only if $f(X)$ contains powers X^i with i even only.

Proposition 2.2.5. *For an irreducible binary Goppa code $\Gamma_2(L, g)$ with $\deg(g(X)) = t$, the minimum distance $d_{\min}^{\Gamma_2(L,g)}$ of the code satisfies $d_{\min}^{\Gamma_2(L,g)} \geq 2t + 1$.*

Proof. Let $c \in \Gamma_2(L, g)$ be a nonzero codeword. Then

$$0 \equiv \sum_{i=1}^{n} \frac{c_i}{X - \alpha_i} = \sum_{i:c_i=1} \frac{1}{X - \alpha_i} = \frac{\sigma'}{\sigma} \bmod g(X)$$

where σ' is a formal derivative of $\sigma = \prod_{i:c_i=1}(X - \alpha_i)$. Then $\sigma' \equiv 0 \bmod g(X)$, i.e. $g|\sigma'$. Notice that σ', being a derivative in characteristic two, consists only of even powers of X, so by Fact 2.2.4, σ' is a perfect square. Since $g(X)$ is an irreducible polynomial in a separable extension (finite extension of a finite field), $g(X)$ has no multiple roots. Thus, $g|\sigma'$ implies $g^2|\sigma'$. Hence, $wt(c) = \deg(\sigma) \geq \deg(\sigma') + 1 \geq 2\deg(g) + 1 = 2t + 1$. □

We now review an instance of the Euclidean algorithm for polynomials in $\mathbb{F}_{2^m}[X]$, as this is needed in the Patterson error-correction algorithm for irreducible binary Goppa codes. Consider the following problem.

Problem 2.2.6. *Given polynomials $g(X), \tau(X) \in \mathbb{F}_{2^m}[X]$ with $\deg(g(X)) = t$ and $\deg(\tau(X)) < t$, find coprime polynomials $\gamma(X), \beta(X) \in \mathbb{F}_{2^m}[X]$ such that*

$$\deg(\gamma(X)) \leq t/2,$$

$$\deg(\beta(X)) \leq (t-1)/2,$$

and

$$\gamma(X) \equiv \beta(X)\tau(X) \bmod g(X).$$

Show that the solution is unique (up to multiplication of both polynomials by the same unit in $\mathbb{F}_{2^m}$).

Euclidean algorithm 2.2.7. *Since $\mathbb{F}_{2^m}[X]$ is a Euclidean domain, we can obtain finite sequences of polynomials $r_{-1}(X), r_0(X), r_1(X), r_2(X), \ldots, r_k(X)$ with*

$$r_{-1}(X) = g(X),$$
$$r_0 = \tau(X),$$
$$r_k(X) = 0,$$

and $q_1(X), q_2(X), \ldots, q_k(X)$, such that for all $1 \leq i \leq k$,

$$r_i(X) = r_{i-2}(X) - q_i(X)r_{i-1}(X)$$

and $\deg(r_i(X)) < \deg(r_{i-1}(X))$.

Let $\beta_{-1} = 0$, $\beta_0 = 1$, and for $1 \leq i \leq k$, define $\beta_i(X) = \beta_{i-2}(X) + q_i(X)\beta_{i-1}(X)$. Let $\gamma_i(X) = r_i(X)$ for $-1 \leq i \leq k$.

Lemma 2.2.8. *For $-1 \leq i \leq k$ and the notation above,*

$$\gamma_i(X) \equiv \beta_i(X)\tau(X) \bmod g(X).$$

Proof. By induction on i. We have $\gamma_{-1} = r_{-1} = g \equiv 0 \cdot \tau = \beta_{-1}\tau \bmod g(X)$ and $\gamma_0 = r_0 = \tau = 1 \cdot \tau = \beta_0\tau$. When $i \geq 1$,

$$\gamma_i = r_i = r_{i-2} - q_i r_{i-1} \equiv \beta_{i-2}\tau - q_i\beta_{i-1}\tau = (\beta_{i-2} + q_i\beta_{i-1})\tau = \beta_i\tau \bmod g$$

by definition of r_i, β_i and using the induction hypothesis and properties of characteristic two.

□

Lemma 2.2.9. *For $-1 \leq i \leq k-1$ and the notation above, $g(X) = \gamma_i(X)\beta_{i+1}(X) + \gamma_{i+1}(X)\beta_i(X)$. Hence, in particular, since $g(X)$ is irreducible, $\gamma_i(X)$ and $\beta_i(X)$ are coprime.*

Proof. By induction on i. We have $g(X) = g(X) \cdot 1 + \tau(X) \cdot 0 = \gamma_{-1}(X)\beta_0(X) + \gamma_0(X)\beta_{-1}(X)$. By using the definition of $\beta_{i+1}(X)$, the $(i+1)$th step in the Euclidean algorithm and the inductive hypothesis for $i \geq 0$, we obtain

$$\gamma_i(X)\beta_{i+1}(X) + \gamma_{i+1}(X)\beta_i(X)$$
$$= \gamma_i(X)(\beta_{i-1}(X) + q_{i+1}(X)\beta_i(X)) + (\gamma_{i-1}(X) - q_{i+1}(X)\gamma_i(X))\beta_i(X)$$
$$= \gamma_i(X)\beta_{i-1}(X) + \gamma_{i-1}(X)\beta_i(X) = g(X).$$

□

Lemma 2.2.10. *For $0 \leq i \leq k$ and the notation above,* $\deg(g(X)) = \deg(\gamma_{i-1}(X)) + \deg(\beta_i(X))$.

Proof. By induction on i. We have $\deg(\gamma_{-1}) + \deg(\beta_0) = \deg(g)$. For $1 \leq i \leq k$, notice that $\deg(\beta_i) = \deg(q_1 \cdot \dots \cdot q_i)$ and consider the ith step of the Euclidean algorithm:

$$r_i = r_{i-2} - q_i r_{i-1}.$$

Since $\deg(r_i) < \deg(r_{i-2})$, it follows that $\deg(q_i) + \deg(r_{i-1}) = \deg(r_{i-2})$. By induction hypothesis, $\deg(r_{i-2}) = \deg(\gamma_{i-2}) = \deg(g) - \deg(\beta_{i-1})$. Hence,

$$\deg(g) = \deg(q_i) + \deg(r_{i-1}) + \deg(\beta_{i-1}) = \deg(r_{i-1}) + \deg(q_1 \cdot \dots \cdot q_i)$$

$$= \deg(\gamma_{i-1}) + \deg(\beta_i).$$

□

Lemma 2.2.11. *There is $I \geq -1$ such that the polynomials $\gamma_I(X)$, $\beta_I(X)$ produced by the Euclidean algorithm above satisfies* $\deg(\gamma_I(X)) \leq t/2$ *and* $\deg(\beta_I(X)) \leq (t-1)/2$.

Proof. In the Euclidean algorithm, the degree of r_i, hence also of γ_i, decreases from t to 1 (since $g(X)$ is irreducible) as i increases. Hence, by the previous lemma, the degree of β_i increases as i increases. Let I be such that $\deg(\gamma_I) < (t+1)/2 \leq \deg(\gamma_{I-1})$. Then $\deg(\gamma_I) \leq t/2$ and $\deg(\beta_I) = \deg(g) - \deg(\gamma_{I-1}) \leq t - (t+1)/2 = (t-1)/2$.

□

Theorem 2.2.12. *The pair of polynomials $\gamma_I(X)$, $\beta_I(X)$ found in Lemma 2.2.11 is the only solution to Problem 2.2.6 (up to multiplication by a unit).*

Proof. Since $\gamma_I(X)$ and $\beta_I(X)$ were found by the Euclidean algorithm, they are coprime and $\gamma_I(X) \equiv \beta_I(X)\tau(X) \bmod g(X)$ as guaranteed by Lemmas 2.2.9 and 2.2.8. By Lemma 2.2.11, they also satisfy the degree requirements and hence, represent a solution to Problem 2.2.6.

Notice that any solution $\gamma(X), \beta(X) \in \mathbb{F}_{2^m}[X]$ to Problem 2.2.6 must have $\beta(X) \neq 0$. Indeed, if $\beta(X) = 0$, then $\gamma(X) \equiv 0 \cdot \tau(X) \bmod g(X)$. Then, $\deg(\gamma(X)) \leq t/2$ forces $\gamma(X) = 0$, and so $\gamma(X)$ and $\beta(X)$ are not coprime, which is a contradiction.

Assume that there exists a pair of polynomials $\gamma'(X), \beta'(X) \in \mathbb{F}_{2^m}[X]$, possibly different from $\gamma_I(X)$, $\beta_I(X)$, that represent a solution to Problem 2.2.6. Then,

$$\frac{\gamma'(X)}{\beta'(X)} \equiv \tau(X) \equiv \frac{\gamma_I(X)}{\beta_I(X)} \bmod g(X),$$

or, equivalently,

$$\gamma'(X)\beta_I(X) \equiv \gamma_I(X)\beta'(X) \bmod g(X).$$

Notice that $\deg(\gamma'(X)\beta_I(X)) \leq t/2 + (t-1)/2 < t = \deg(g(X))$, and, similarly, $\deg(\gamma_I(X)\beta'(X)) < t$, and so the above holds as an equality

$$\gamma'(X)\beta_I(X) = \gamma_I(X)\beta'(X)$$

in $\mathbb{F}_{2^m}[X]$. Since $\gamma_I(X)$ and $\beta_I(X)$ are coprime, it follows that $\gamma'(X) = \gamma_I(X) \cdot \eta(X)$ and $\beta'(X) = \beta_I(X) \cdot \mu(X)$ for some $\eta(X), \mu(X) \in \mathbb{F}_{2^m}[X]$. If $\gamma_I(X) = 0$, then $\gamma'(X) = 0 = \gamma_I(X) \cdot \mu(X)$, i.e. $\gamma'(X)$ and $\beta'(X)$ are a unit multiple of $\gamma_I(X)$ and $\beta_I(X)$, hence represent the same solution.

Assume $\gamma_I(X) \neq 0$. We have

$$\frac{\gamma_I(X)}{\beta_I(X)} = \frac{\gamma'(X)}{\beta'(X)} = \frac{\gamma_I(X) \cdot \eta(X)}{\beta_I(X) \cdot \mu(X)}$$

and so $\eta(X) = \mu(X)$. Since we assumed that $\gamma'(X)$ and $\beta'(X)$ are coprime, $\mu(X)$ must, in fact, be a unit. Hence, $\gamma'(X)$, $\beta'(X)$ represent the same solution as $\gamma_I(X)$, $\beta_I(X)$.

□

We now describe the most efficient known decoding algorithm for the irreducible binary Goppa codes that is due to Patterson [35].

Assume that a codeword $c \in \Gamma_2(L, g)$ is sent and a vector $v = c + e$ is received, where $e \in \mathbb{F}_2^n$ is an error vector with $wt(e) \leq t$. To recover c, use the following algorithm (adapted from [16]).

Patterson algorithm for irreducible binary Goppa codes 2.2.13.
Input: *irreducible binary Goppa code $\Gamma_2(L, g)$ of degree t with support $L = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ and a received vector $v \in \mathbb{F}_2^n$ such that*

$$\exists c \in \Gamma_2(L, g) : d_{\Gamma_2(L,g)}(v, c) \leq t.$$

Output: *codeword $c \in \Gamma_2(L, g)$.*

Algorithm:

1. *Calculate* $\sqrt{(\sum_{i=1}^n \frac{v_i}{X-\alpha_i})^{-1} - X} \bmod g(X)$*, and denote its lift to* $\mathbb{F}_{2^m}[X]$ *by* $\tau(X)$*. If the calculation fails, output* v*.*

2. *Find coprime polynomials* $\gamma(X)$*,* $\beta(X) \in \mathbb{F}_{2^m}[X]$ *with* $\deg(\gamma(X)) \leq t/2$ *and* $\deg(\beta(X)) \leq (t-1)/2$ *such that* $\gamma(X) \equiv \beta(X)\tau(X) \bmod g(X)$*.*

3. *Let* $\sigma(X) = \gamma^2(X) + X\beta^2(X)$*. Find the roots of* $\sigma(X)$ *in* $\mathbb{F}_{2^m}$*.*

4. *Let* $e_i = 1$ *if* α_i *is a root of* $\sigma(X)$*. Otherwise,* $e_i = 0$*.*

5. *Let* $c = v + e$*.*

6. *Output* c*.*

Why the algorithm works. Since $g(X)$ is irreducible, $\mathbb{F}_{2^m}[X]/(g(X))$ is a field. Hence, every nonzero element has an inverse and, by Fact 2.2.4, every element has a unique square root. Thus, the calculation in 1. fails if and only if $\sum \frac{v_i}{X-\alpha_i} \equiv 0 \bmod g(X)$ which happens if and only if v is a codeword, i.e. no error correction is needed and the algorithm outputs v.

Assume that $v = c + e \in \mathbb{F}_2^n$ with $e \neq 0$. Define the *error locator polynomial* as

$$\sigma(X) = \prod_{i:e_i=1} (X - \alpha_i).$$

By assumption, $wt(e) \leq t$, therefore $\deg(\sigma(X)) \leq t$.

Using the definition $\sum_{i=1}^n \frac{c_i}{X-\alpha_i} \equiv 0 \bmod g(X)$ of a codeword c, we have

$$0 \neq \sum_{i=1}^n \frac{v_i}{X-\alpha_i} = \sum_{i=1}^n \frac{c_i}{X-\alpha_i} + \sum_{i=1}^n \frac{e_i}{X-\alpha_i} \equiv$$

$$\equiv \sum_{i:e_i=1} \frac{1}{X-\alpha_i} = \frac{\sigma'(X)}{\sigma(X)} \bmod g(X)$$

where $\sigma'(X)$ is a formal derivative of $\sigma(X)$. Notice that $\sigma'(X)$ is non-zero.

Express the error locator polynomial as a sum of squares and non-squares, i.e. $\sigma(X) = \gamma^2(X) + X \cdot \beta^2(X)$ and $\sigma'(X) = \beta^2(X)$ for polynomials $\gamma(X), \beta(X) \in \mathbb{F}_{2^m}[X]$. Notice that $\beta(X)$ is non-zero and, since $\sigma(X)$ has no multiple roots, $\gamma(X)$ and $\beta(X)$ must be coprime.

The above identity becomes

$$\left(\sum_{i=1}^{n} \frac{v_i}{X-\alpha_i}\right)^{-1} \equiv \frac{\gamma^2(X) + X \cdot \beta^2(X)}{\beta^2(X)} = \left(\frac{\gamma(X)}{\beta(X)}\right)^2 + X \bmod g(X).$$

Now, $\left(\sum_{i=1}^{n} \frac{v_i}{X-\alpha_i}\right)^{-1} - X$ has a unique square root modulo $g(X)$. Denote its lift to $\mathbb{F}_{2^m}[X]$ by $\tau(X)$. We have

$$\tau(X)\beta(X) \equiv \gamma(X) \bmod g(X).$$

Here $\deg(g(X)) = t$, $\deg(\tau(X)) < t$ and, since $\deg(\sigma(X)) \leq t$, we must have $\deg(\gamma(X)) \leq t/2$ and $\deg(\beta(X)) \leq (t-1)/2$. Hence, the unknown polynomials $\gamma(X)$ and $\beta(X)$ represent a solution to Problem 2.2.6 which, by Theorem 2.2.12, is guaranteed to be unique (up to multiplication by a unit, but this does not change the roots of $\sigma(X)$). Finding this solution, using the formula for $\sigma(X)$ and recovering its roots then determines the error vector e and hence, also the codeword c.

The expression $\sum_{i=1}^{n} \frac{v_i}{X-\alpha_i}$ is usually computed as $Hv \bmod g(X)$ where H is the parity-check matrix (2.1) for $\Gamma_2(L, g)$. To calculate its inverse, one uses the extended Euclidean algorithm. The square root is found by applying the inverse Frobenius map and to recover $\gamma(X)$, $\beta(X)$, one follows the version of the Euclidean algorithm described above.

In terms of complexity, the hardest part of the algorithm is to determine the roots of $\sigma(X)$. Strenzke [42] discussed several known approaches to root finding, namely, a naive polynomial evaluation method, an evaluation with division by known roots, the Berlekamp trace algorithm [3], a method based on polynomial decomposition into so-called linearized polynomials [18] and some combinations thereof. The algorithms were compared in terms of their performance and security that they offer when implemented within a code-based cryptosystem. It turns out that the first two methods are too slow and the Berlekamp trace algorithm (that has already been used in some implementations of McEliece cryptosystem, e.g. [8]) allows a substantial message-aimed timing attack on the underlying cryptosystem. All algorithms were, in fact, found to exhibit timing-attack vulnerabilities to some extent; the one where the problems were the subtlest and only related to recovering the support and not the message, is the decomposition into linearized polynomials [42]. This method also turns out to be the fastest of all algorithms when performed on Goppa codes with parameters corresponding to cryptosystems with the smallest public keys – an issue of paramount importance in code-based cryptography, as we see in later chapters. Yet, as Strenzke concludes, there may be contexts in which the other algorithms are more appropriate and so currently there is no definitive answer as to which root-finding algorithm is the most suitable.

The performance of a root-finding algorithm depends on many factors, such as the particular codes in use and the exact implementation of field arithmetic. A rough estimate provided in [16] states that the complexity of such an algorithm is $O(ntm^2)$ operations where t is the degree of $g(X)$ and m is the size of its coefficients.

Going back to the Patterson algorithm, we conclude that its overall complexity is $O(ntm^2)$. Thus, we see that given a Goppa polynomial $g(X)$ of degree t with coefficients $g_i \in \mathbb{F}_{2^m}$ and a corresponding support L of cardinality n, there exists a fast and easily implementable algorithm correcting the full t errors. In fact, notice that it's enough to only know some non-zero constant multiple of $g(X)$ to be able to decode, since $g(X)$ and $\alpha \cdot g(X)$, where $\alpha \in \mathbb{F}_{2^m} \backslash \{0\}$ generate the same Goppa code.

On the other hand, if (a constant multiple of) the polynomial $g(X)$ is not known, there is currently no known efficient way of decoding [16]. A question may be, from what information can one learn $g(X)$? For example, Heyse et al. [24] show that $g(X)$ can usually be reconstructed from the knowledge of the support L and a generator matrix for $\Gamma_2(g, L)$ in systematic form. Or, as pointed out in [16], knowing the support L and the parity-check matrix H in the form (2.1) enables one to compute $g_t^{-1} g(X)$, and therefore, also efficiently decode.

These issues ultimately determine when a code-based cryptosystem using Goppa codes can be regarded as secure and when not. Due to their properties, the irreducible binary Goppa codes were the first codes used in a code-based cryptosystem devised in 1978. After 30+ years, when other code families have been tried out for this purpose, too, the binary Goppa codes still remain pretty much the only unbroken and efficient type of code used in code-based cryptography.

2.3 Hard problems

We now discuss some hard coding theory problems that are frequently used in constructing, but also attacking, code-based cryptographic schemes. Consider the following problems.

Problem 2.3.1 (The problem of finding weights)**.** *Given a binary linear code $\mathcal{C} \subset \mathbb{F}_2^n$, find a non-zero codeword $c \in \mathcal{C}$ with the minimum weight.*

Decisional version: *Given a binary linear code $\mathcal{C} \subset \mathbb{F}_2^n$ and $w \in \mathbb{N}$, is there a non-zero $c \in \mathcal{C}$ such that $wt(c) \leq w$?*

After being a conjecture of Berlekamp, McEliece and van Tilborg [5] for almost 20 years, Problem 2.3.1 was shown to be NP-hard (and the corresponding decision problem NP-complete) by Vardy [44].

Problem 2.3.2 (The general decoding problem)**.** *Given a binary linear code $\mathcal{C} \subset \mathbb{F}_2^n$ and a vector $v \in \mathbb{F}_2^n$, find $c \in \mathcal{C}$ such that $d(v, c) = \min_{a \in \mathcal{C}}\{d(v, a)\}$.*

Decisional version: *Given a binary linear code $\mathcal{C} \subset \mathbb{F}_2^n$, a vector $v \in \mathbb{F}_2^n$ and $w \in \mathbb{N}$, is there $c \in \mathcal{C}$ such that $d(v, c) \leq w$?*

Problem 2.3.3 (The syndrome decoding problem)**.** *Given a binary linear code $\mathcal{C} \subset \mathbb{F}_2^n$ with an $(n-k) \times n$ parity-check matrix H and a syndrome $s \in \mathbb{F}_2^{n-k}$, find $e \in \mathbb{F}_2^n$ such that $He^T = s^T$ with $wt(e)$ minimal.*

Decisional version: *Given a binary linear code $\mathcal{C} \subset \mathbb{F}_2^n$ with an $(n-k) \times n$ parity-check matrix H, a syndrome $s \in \mathbb{F}_2^{n-k}$ and a $w \in \mathbb{N}$, is there $e \in \mathbb{F}_2^n$ with $wt(e) \leq w$ such that $He^T = s^T$?*

Proposition 2.3.4. *Problems 2.3.2 and 2.3.3 are computationally equivalent.*

Proof. We prove the claim for the computational versions of the problems; the proof for decisional versions is similar.

Assume that we have an oracle for solving Problem 2.3.3 and are given $v \in \mathbb{F}_2^n$ and want to find a closest $c \in \mathcal{C}$. Query the oracle with any parity-check matrix H for $\mathcal{C}$ and $s^T = Hv^T$. For the returned vector e, $He^T = Hv^T$ such that $wt(e)$ is minimal, i.e. $v + e$ is a codeword with $d(v, v + e)$ minimal.

Conversely, assume that we have an oracle for solving Problem 2.3.2 and are given a parity-check matrix H and a syndrome $s \in \mathbb{F}_2^{n-k}$. By solving the system of $n - k$ equations in n unknowns, we compute some $z \in \mathbb{F}_2^n$ satisfying $Hz^T = s^T$. Then, for some $e \in \mathbb{F}_2^n$, $0 = Hz^T + He^T = H(z + e)^T$, i.e. $z + e$ is a codeword if and only if $He^T = s^T$. Moreover, $z + e$ is a closest codeword to z if and only if $He^T = s^T$ and $wt(e)$ is minimal. Thus, by querying the oracle with z one obtains $z + e$, where e is a desired solution. $\square$

The decisional version of Problem 2.3.3 was shown to be NP-complete [5]. Since it can be reduced to the computational problem, it follows that the computational version of Problem 2.3.3 is NP-hard. Similarly, by the claim, the versions of Problem 2.3.2 are NP-complete, and NP-hard respectively.

Then, if NP≠P, these coding problems are provably hard and therefore building cryptographic schemes with security resting on these problems would be a promising idea. Moreover, unlike in the case of factoring, there is no known quantum algorithm that can solve these problems faster than classical algorithms and schemes based on these problems are, therefore, referred to as 'post-quantum'.

We, however, remark that one still needs to be careful about judging the exact level of security, since in reality most of the cryptographic schemes rely on slight modifications of the above problems, such as the following.

Problem 2.3.5 (The bounded decoding problem)**.** *Given a binary linear code* $\mathcal{C} \subset \mathbb{F}_2^n$ *and a vector* $v \in \mathbb{F}_2^n$*, find all* $c \in \mathcal{C}$ *such that* $d(v,c) \leq \lfloor \frac{d_{\min}^{\mathcal{C}}-1}{2} \rfloor$.

Decisional version: *Given a binary linear code* $\mathcal{C} \subset \mathbb{F}_2^n$*, a vector* $v \in \mathbb{F}_2^n$ *and* $w \leq \lfloor \frac{d_{\min}^{\mathcal{C}}-1}{2} \rfloor$*, is there* $c \in \mathcal{C}$ *such that* $d(v,c) \leq w$*?*

Problem 2.3.6 (The bounded syndrome decoding problem)**.** *Given a binary linear code* $\mathcal{C} \subset \mathbb{F}_2^n$*, an* $(n-k) \times n$ *parity-check matrix* H *in which every* $d_{\min}^{\mathcal{C}} - 1$ *columns are linearly independent and a syndrome* $s \in \mathbb{F}_2^{n-k}$*, find all* $e \in \mathbb{F}_2^n$ *such that* $He^T = s^T$ *and* $wt(e) \leq \lfloor \frac{d_{\min}^{\mathcal{C}}-1}{2} \rfloor$.

Decisional version: *Given a binary linear code* $\mathcal{C} \subset \mathbb{F}_2^n$ *with an* $(n-k) \times n$ *parity-check matrix* H *in which every* $d_{\min}^{\mathcal{C}} - 1$ *columns are linearly independent, a syndrome* $s \in \mathbb{F}_2^{n-k}$ *and a* $w \leq \lfloor \frac{d_{\min}^{\mathcal{C}}-1}{2} \rfloor$*, is there* $e \in \mathbb{F}_2^n$ *with* $wt(e) \leq w$ *such that* $He^T = s^T$*?*

Note that the linear independence condition on columns of H in Problem 2.3.6 is equivalent to the requirement that the minimum distance of the given code is at least $d_{\min}^{\mathcal{C}}$.

Similarly as in Problem 2.3.4, the respective versions of Problems 2.3.5 and 2.3.6 can be shown to be computationally equivalent. It is conjectured [44] that these problems are NP-hard, however, there is currently no actual proof to confirm this.

For the purposes of later sections, notice the following: if the solution set to Problem 2.3.6 is non-empty, then, in fact, it consists of exactly one solution. Indeed, assume that there are two solutions e', $e'' \in \mathbb{F}_2^n$. Then,

$$0 = H(e')^T - H(e'')^T = H(e' - e'')^T,$$

and so $e' - e''$ is a codeword with weight $wt(e' - e'') \leq wt(e') + wt(e'') \leq 2 \cdot \lfloor \frac{d_{\min}^{\mathcal{C}}-1}{2} \rfloor < d_{\min}^{\mathcal{C}}$. Therefore, $e' - e'' = 0$, i.e. $e' = e''$.

Finally, we discuss one last assumption needed in security considerations of cryptosystems and signature schemes in Chapters 3 and 4.

Assumption 2.3.7 (The Goppa code indistinguishability)**.** *Let* $\texttt{Goppa}(k, n)$ *be the set of all* $k \times n$ *generator matrices of irreducible binary Goppa codes and* $\texttt{Random}(k, n)$ *the set of all* $k \times n$ *random matrices of full rank, i.e. generator matrices for random linear codes. Pick a value* $b \in \{0, 1\}$ *uniformly at random. If* $b = 0$*, let* G *be a matrix randomly chosen from* $\texttt{Goppa}(k, n)$*, otherwise, pick* G *randomly from* $\texttt{Random}(k, n)$*. Then, there is no algorithm polynomial in* n *that can correctly decide whether* $G \in \texttt{Goppa}(k, n)$ *with probability non-negligably larger than* $1/2$*.*

In other words, Assumption 2.3.7 asserts that there is no property computable in polynomial time that behaves differently in the Goppa codes and random codes, and so, these two classes of codes are, in polynomial time, indistinguishable from each other.

This assumption was first brought up in [14]. It is needed if one wishes to use Goppa codes in the cryptographic schemes, while at the same time, one wants to base the schemes' security on hardness of decoding a random code (Problem 2.3.2 or 2.3.5).

However, little is actually known about the indistinguishability assumption. It sounds plausible, since, as explained in Section 2.2, Goppa codes resemble random codes in many aspects, but there does not exist any proof of Assumption 2.3.7. In fact, Faugere et al. [17] disproved the indistinguishability assumption for codes with large information rate by showing that in such codes a certain easily computable algebraic quantity in Goppa codes differs from random codes.

For the remaining codes, however, Assumption 2.3.7 is generally assumed to be true.

Chapter 3

Code-based encryption

Code-based cryptography is an area of active cryptographic research, encompassing cryptosystems, digital signatures, code-based hash functions and an identification scheme (see, for example, the surveys [34, 11, 16]). It represents an alternative to today's most widespread cryptographic schemes that are based on factorization and discrete logarithm problem. Unlike them, the code-based schemes do not seem to be vulnerable to Shor's algorithm, nor to any other quantum algorithm [36]. Hence, they are considered to be classical schemes that are secure and suitable for the post-quantum world.

Another significant advantage of code-based systems over the widely used schemes is a very efficient encryption and decryption that is several orders of magnitude faster than in RSA or elliptic curves.

The first code-based cryptosystem was published in 1978 by R. McEliece [30] and was designed to use the irreducible binary Goppa codes. In 1986, Niederreiter published another code-based cryptosystem using generalised Reed-Solomon codes [16]. Although this version of Niederreiter cryptosystem turned out to be insecure [16], the Niederreiter system works well with the Goppa codes and, in fact, the security of both McEliece and Niederreiter systems using the irreducible binary Goppa codes was later shown to be equivalent [28].

In 1994 Stern (see [16, 34]) proposed an identification scheme inspired by the Niederreiter system and based on the difficulty of syndrome decoding. In 1996, Fisher and Stern used the identification scheme and syndrome decoding to construct a generator of pseudorandom numbers (see [34]) and, in 2005, based on the same ideas, Augot et al. [1] came up with a family of code-based cryptographic hash-functions. Finally, in 2001 Courtois et al. [14] managed to construct the first practical code-based signature scheme. Known as a CFS scheme, this signature scheme is based on inverting the

Niederreiter system and modifying the parameters of the underlying Goppa code in a special way.

All in all, code-based cryptography has a long tradition. But, despite the fact that it was established around the same time as RSA, it did not become as popular. Why? The main two reasons are as follows.

Signature schemes. Originally, it was believed that construction of a code-based signature scheme would not be possible from the known cryptosystems. It took over two decades before the CFS scheme was proposed and even today, there is no complete security proof for the scheme. We describe the difficulties in this area in Chapter 4. The main part of the thesis is concerned with the discussion of the amendments of the CFS scheme as well as the new candidates for the signature schemes that have recently appeared.

Key sizes. Code-based cryptosystems tend to have relatively large public key sizes. As estimated in [43], for a 100-bit security level, the corresponding McEliece public key has about 100KB. At the time of appearance of the McEliece cryptosystem, this was a significant problem. Since then, considerable efforts have been invested into research to make the keys smaller.

One possibility is to try to use different codes in the McEliece/Niederreiter scheme, especially ones that are over larger finite fields. Tillich et al. [27] sums up that all the following codes have been used: quasi-cyclic and quasi-dyadic Goppa codes, subcodes of generalised Reed-Soloman codes, Reed-Muller codes, algebraic geometry codes, LDPC, MDPC and convolutional codes. However, most of the new constructions were immediately broken, since it was possible to find some specific characteristic of the underlying secret code that distinguished the code from a random one and enabled decoding [27].

Another branch of research concentrates on the original McEliece scheme with irreducible binary Goppa codes. It has withstood all cryptographic scrutiny over the years and is still unbroken today [16] (apart from when used with Goppa codes with large rates that are distinguishable from random code as shown above – but this danger is only relevant in case of code-based signatures as we see in Chapter 4). Also, the key sizes are no longer a substantial problem as they used to be 30 years ago. In large part this is due to an increased memory space available on platforms nowadays, but also due to clever implementation methods. For example, in [43] the relevant information from the public key is processed straight as it is being received, and so there is no need to keep the entire public key in memory all at once.

We concentrate on the latter area of research. In this chapter, we introduce and briefly analyze the McEliece and Niederreiter cryptosystems using the irreducible binary Goppa codes. This prepares the background for the main discussion of signature schemes in Chapter 4.

3.1 The McEliece and Niederreiter cryptosystems

Code-based cryptography takes advantage of the existence of codes for which an efficient error-correcting algorithm is known and of those in which error-correction is known to be a hard problem.

Roughly speaking, one constructs a secret correspondence between a private t-error-correcting code $\mathcal{C}$ and a public code $\mathcal{C}'$ with no efficient error-correction (usually, a random linear code or a code indistinguishable from random code). Alice, the sender, incorporates her message into a codeword in $\mathcal{C}'$, purposely adds t errors to it and sends it to Bob. Since decoding in $\mathcal{C}'$ is hard, Eve, if intercepting the transmission, cannot deduce any information about the underlying message. Bob, on the other hand, uses the secret correspondence between codes $\mathcal{C}$ and $\mathcal{C}'$, maps the received vector to $\mathcal{C}$ and corrects it and extracts the message efficiently there.

The exact details may vary, for example, one can equally well take the message to be an error vector and transmit the corresponding syndrome in the public code $\mathcal{C}'$, but in all cases the trapdoor is the secret correspondence between the codes $\mathcal{C}$ and $\mathcal{C}'$.

We now introduce the two most important code-based cryptosystems formally.

Definition 3.1.1 (The McEliece cryptosystem)**.**
Public parameters: *Integers n, k, t such that there exists a t-error-correcting, k-dimensional irreducible binary Goppa code of length n.*

Setup: *Select a generator matrix G in systematic form for a random $[n, k]$ irreducible binary Goppa code $\Gamma_2(L, g)$ of degree t, a random $k \times k$ invertible matrix S and a random $n \times n$ permutation matrix P.*

Private key: *Goppa polynomial $g(X)$ and support L, the parity-check matrix H for $\Gamma_2(L, g)$ of the form (2.1), matrices S and P.*

Public key: *The $k \times n$ matrix $\hat{G} := SGP$.*

Plaintext space: *Message $m \in \mathbb{F}_2^k$.*

Ciphertext space: *Ciphertext $c \in \mathbb{F}_2^n$.*

Encryption: *Given a plaintext $m \in \mathbb{F}_2^k$, the corresponding ciphertext c is*

$$c = m\hat{G} + e,$$

where e is an error vector of weight $wt(e) = t$ chosen at random.

Decryption: *Given a ciphertext c, compute $cP^{-1} = mSG + eP^{-1}$. Note that $wt(eP^{-1}) = t$ and since S is invertible, SG is just another generator matrix for the secret code $\Gamma_2(L, g)$. Hence, $cP^{-1} \in \mathbb{F}_2^n$ is distance t from a codeword d in $\Gamma_2(L, g)$. Recover d by using the Patterson decoding algorithm 2.2.13. Finally, from $d = mSG$ recover the plaintext m by linear algebra.*

Definition 3.1.2 (The Niederreiter cryptosystem)**.**
Public parameters: *Integers n, k, t such that there exists a t-error-correcting, k-dimensional irreducible binary Goppa code of length n.*

Setup: *Select a parity-check matrix H in systematic form for a random $[n, k]$ irreducible binary Goppa code $\Gamma_2(L, g)$ of degree t, a random $(n-k) \times (n-k)$ invertible matrix M and a random $n \times n$ permutation matrix P.*

Private key: *Goppa polynomial $g(X)$ and support L, the parity-check matrix H for $\Gamma_2(L, g)$ of the form (2.1), matrices M and P.*

Public key: *The $(n-k) \times n$ matrix $\hat{H} := MHP$.*

Plaintext space: *Message $m \in \mathbb{F}_2^n$ and $wt(m) = t$.*

Ciphertext space: *Ciphertext $c \in \mathbb{F}_2^{n-k}$.*

Encryption: *Given a plaintext $m \in \mathbb{F}_2^n$ with $wt(m) = t$, the corresponding ciphertext c is*

$$c^T = \hat{H} m^T.$$

Decryption: *Given a ciphertext $c \in \mathbb{F}_2^{n-k}$, by using linear algebra, find any $z \in \mathbb{F}_2^n$ such that $\hat{H} z^T = c^T$. Note that since M is invertible, MH is just another parity-check matrix for the secret code $\Gamma_2(L, g)$. Then $0 = c^T - c^T = \hat{H} z^T - \hat{H} m^T = MH(Pz^T - Pm^T)$, i.e. $(Pz^T - Pm^T)^T$ is a codeword in $\Gamma_2(L, g)$. Compute zP^T and note that $wt(mP^T) = t$. Hence, $zP^T \in \mathbb{F}_2^n$ is a vector at distance t from the codeword $zP^T - mP^T$. Patterson's algorithm 2.2.13 applied to zP^T recovers $zP^T - mP^T$. Hence, we obtain mP^T, and, subsequently, also m.*

Remark. In the Niederreiter system, the condition $wt(m) = t$ guarantees an unambiguous decryption. Without it, there are multiple vectors $z \in \mathbb{F}_2^n$ satisfying the relation $c^T = \hat{H} z^T$, for a given ciphertext $c \in \mathbb{F}_2^{n-k}$, all of which can easily be found by linear algebra.

Similarly, in the McEliece scheme, recovering the message m from a known ciphertext c is equivalent to determining the error vector e: when $c - e$ is known, one easily computes m from $c - e = m\hat{G}$ by linear algebra.

Thus, in both cryptosystems, breaking the scheme essentially means finding a low-weight error vector.

Li et al. [28] showed that the securities of the McEliece and Niederreiter systems are equivalent. More precisely:

Theorem 3.1.3. *Suppose we are given instances of McEliece and Niederreiter cryptosystems with the same secret code $\Gamma_2(L, g)$ and matrix P. Then, an attacker can break the McEliece cryptosystem if and only if she can break the Niederreiter cryptosystem.*

Proof. Notice first that, since P is the same permutation in both cryptosystems, $GH^T = 0$ if and only if $\hat{G}\hat{H}^T = 0$. This means that whenever the underlying secret Goppa code is the same, then also the public codes in the two cryptosystems are the same, i.e. $\hat{G} = SGP$ is a generator matrix and $\hat{H} = MHP$ is a parity-check matrix for the same linear code.

Assume that the attacker can break the McEliece cryptosystem and is given a ciphertext $c = \hat{H} m^T$ of the Niederreiter system. The attacker finds any $z \in \mathbb{F}_2^n$ such that $c = \hat{H} z^T$. Then $\hat{H}(z - m)^T = 0$, i.e. $z - m$ is a codeword in the public code with the parity-check matrix $\hat{H}$ and

$d(z, z-m) = wt(m) = t$. Hence, z is a valid ciphertext in the McEliece cryptosystem: $z = m'\hat{G} + m$ for some $m' \in \mathbb{F}_2^k$. By breaking the McEliece cryptosystem, one recovers m' and m.

Conversely, assume that the attacker is given a McEliece ciphertext $c = m\hat{G} + e$. Multiplying by $\hat{H}^T$, one obtains $c\hat{H}^T = m\hat{G}\hat{H}^T + e\hat{H}^T = e\hat{H}^T$ which is a valid Niederreiter ciphertext. By breaking the Niederreiter scheme, the attacker recovers e, and hence, from $c - e = m\hat{G}$ she also obtains m by linear algebra.

□

3.2 Analysis of the cryptosystems

We now analyze the McEliece and the Niederreiter cryptosystems in greater detail. Notice that although the security of the McEliece and Niederreiter systems is equivalent and both cryptosystems work with the same private and public codes, there are differences between the two systems. For example, encryption is randomized in the McEliece, but not in the Niederreiter cryptosystem. Similarly, subtle differences appear below. In all cases we refer to cryptosystems using a t-error-correcting $[n, k]$ Goppa code with m the degree of underlying field extension $\mathbb{F}_{2^m}$. The summary of the performance estimates for the McEliece and the Niederreiter cryptosystems can be found in Table 3.1.

Key sizes. Engelbert et al. [16] compute the key sizes for both the McEliece and the Niederreiter cryptosystems. The authors state that the McEliece private key, as described in Definition 3.1.1, has size $(n-k)n + (n-k+1+2\log_2 n) + k^2 + n\log_2 n$ bits. This is $O((n-k)^2 + kn + n\log_2 n)$ bits. The Niederreiter private key, as given in Definition 3.1.2, has size $(n-k+1+2\log_2 n)+(n-k)^2+n\log_2 n$ bits [16], which is $O((n-k)^2+n\log_2 n)$ bits.

The public key in the McEliece cryptosystem requires publishing of all $k \cdot n$ entries of matrix $\hat{G}$, while in the Niederreiter it is the $(n-k) \cdot n$ entries of matrix $\hat{H}$. If one uses conversions (see below), the public matrices may stay in the systematic form, which results in $k \cdot (n-k)$ bits needed to store the public key in both the McEliece and Niederreiter systems [16].

The exact comparison of the key sizes presented here depends on the choice of n and k. For the parameters recommended in [7] (see below), the Niederreiter public key size is considerably larger than its private key size. In the McEliece system the situation is less obvious but the private key size is at least of the same order of magnitude as the public key size. If no conversions are used, then with the recommended parameters the McEliece public key is larger than the Niederreiter public key. Finally, the McEliece private key size is of at least the same order of magnitude as the Niederreiter private key size.

In general, more problematic are the public key sizes in the above cryptosystems. The reasons for this are twofold. Firstly, the private key estimates given here are rather generous and, if needed, may be decreased. This can be done by omitting the parity-check matrix H from the private keys, as H is only used to speed up the computation of the syndrome in the Patterson decoding algorithm needed in the decryption process. On the other hand, no further reduction in public key sizes is possible. Secondly, public keys usually need to be transmitted to a sender via network, while private keys are stored by the receiver. Hence, having small public keys is more important.

Running time. For the cryptosystems as described in Section 3.1, Engelbert et al. [16] compute the following running times. Generating an instance of McEliece and Niederreiter cryptosystem takes $O(k^2n + n^2 + t^3(n-k) + (n-k)^3)$ and $O((n-k)^2n + n^2 + t^3(n-k))$ binary operations, respectively [16]. Encryption in McEliece takes $O(k \cdot n + t + w)$ operations, where w is the time to generate the error vector [16]. In Niederreiter, the plaintext m is of length n with $wt(m) = t \ll n$, and so may be compressed in different ways. This affects the encryption times in the cryptosystem accordingly [16]. A rough estimate for the Niederreiter encryption time is $O((n-k) \cdot t)$ operations, since the encryption basically consists of summing up t columns of the public matrix $\hat{H}$. Decryption in both cryptosystems takes $O(ntm^2)$; see [16] for the proof.

	McEliece system	Niederreiter system
public key size	kn	$(n-k)n$
private key size	$(n-k)^2 + kn + n\log_2 n$	$(n-k)^2 + n\log_2 n$
encryption operations	$kn + t + w$	$(n-k)t$
decryption operations	ntm^2	ntm^2

Table 3.1: McEliece and Niederreiter cryptosystems summary. All data represent the big-O values.

Proposed parameters. Suggestions for suitable parameters for code-based cryptosystems resisting the best current attacks are given by, for example, Bernstein et al. [7]. For 80-bit security level the authors propose to use degree-t $[n, k]$ irreducible binary Goppa codes where $n = 2048$, $k = 1751$ and $t = 27$. Apart from this "traditional" set in which n is a power of 2, and no list decoding is performed, the authors also consider other possibilities that minimize the size of the public key for a given security level. In particular, they give new values of n and introduce more errors than what the error-correcting capacity of the used Goppa codes is. The presence of more errors increases the security level, or, equivalently, enables smaller key sizes for the same security level [7]. The decoding algorithm is then forced to contain list decoding techniques. The recommended parameter sets are given in Table 3.2. These are suitable for both the McEliece as well as the Niederreiter system [7].

security level	n	k	t	no. of errors
80-bit	1632	1269	33	34
128-bit	2960	2288	56	57
256-bit	6624	5129	115	117

Table 3.2: Recommended Goppa code parameters for the McEliece and Niederreiter systems (taken from [7]).

Security considerations. We now look closer at the algorithm in 3.1.1 (similar considerations apply to algorithm 3.1.2). The private code is a t-error-correcting $[n, k]$ irreducible binary Goppa code $\Gamma_2(L, g)$ with generator matrix G in systematic form. Multiplying by the invertible random matrix S on the left does not change the code and only produces a random generator matrix SG for $\Gamma_2(L, g)$. Multiplying by the random permutation matrix P on the right, we see that the public code in the McEliece cryptosystem is permutation equivalent to the private code. More precisely, one can easily show that the public code is also a t-error-correcting irreducible binary Goppa code, denote it $\Gamma'_2(L', g')$, with the same minimum distance and rate as the secret code and $\hat{G}$ is a random generator matrix for $\Gamma'_2(L', g')$.

Hence, the system can actually be reduced to the following requirements: publishing a random generator matrix for a Goppa code for which the Goppa polynomial and support is (secretly) known to enable the decoding by a legitimate recipient. If a conversion is used on top of the algorithm described in 3.1.1, then, in fact, the public generator matrix can even be in systematic form and the keys in the McEliece cryptosystem become

- private key: support L and a Goppa polynomial $g(X)$ defining a degree-t $[n, k]$ irreducible binary Goppa code $\Gamma_2(L, g)$
- public key: matrix $\hat{G} = (I_k|Q)$, where I_k is a $k \times k$ identity matrix, Q is a $k \times (n-k)$ matrix and G is a generator matrix for $\Gamma_2(L, g)$

as described in [43].

Now, what are the security assumptions in McEliece's scheme? One can define the following problem.

Problem 3.2.1 (The McEliece problem)**.** *Given a McEliece ciphertext $c = m\hat{G} + e$, find $m \in \mathbb{F}_2^k$ such that $d(c, m\hat{G}) = wt(e)$.*

Since the public code in the McEliece system is not a random, but a Goppa code, hardness of the McEliece problem relies on two factors: on hardness of the problem of decoding a random code (Problem 2.3.5) and on indistinguishability of the Goppa codes from the random ones (Assumption 2.3.7).

The latter condition is exactly the reason why the originally proposed Niederreiter scheme with the generalised Reed-Solomon codes, as well as many new suggested constructions of McEliece cryptosystem using other than the Goppa codes, have been broken. In these alternative codes, one was able to find a characteristic property of the code that enabled decoding much more efficiently than any of the methods for decoding a generic random linear code. Thus, in this respect, Goppa codes seem to indeed be a very special class of codes, with some authors, such as in [42], leaning towards the conviction that they represent the only known secure codes for use in code-based schemes.

The general decoding problem can be rephrased into the problem of finding weights as follows. Given a code $\Gamma_2(L, g)$ and a ciphertext $c = m\hat{G}+e$ in the McEliece system, then instead of looking for the closest codeword $m\hat{G} \in \Gamma_2(L, g)$, one can search for a codeword with minimum weight in a new code $\mathcal{C}'$ with generator matrix $\begin{pmatrix} \hat{G} \\ c \end{pmatrix}$ (matrix $\hat{G}$ with appended vector c in the last row). Indeed, by construction, the error vector e must be a codeword with minimum weight in $\mathcal{C}'$ and so solving Problem 2.3.1 of finding weights in $\mathcal{C}'$ is equivalent to solving the Bounded general decoding problem 2.3.5 in $\Gamma_2(L, g)$.

Attacks. Attacks on the McEliece and Niederreiter systems have been traditionally (starting in the original paper by McEliece [30]) divided into structural attacks and direct attacks. The structural attacks try to recover the secret code $\Gamma_2(L, g)$ of the cryptosystem, given a generator matrix $\hat{G}$ of the public code $\Gamma_2'(L', g')$. For example, one may try all possibilities for a generator matrix G of $\Gamma_2(L, g)$ and test G and $\hat{G}$ for permutation equivalence. Direct attacks, on the other hand, seek to decode the public code $\Gamma_2'(L', g')$ directly. Of course, if we take a more abstract view of the McEliece (or Niederreiter) cryptosystem such as presented in the Security Considerations subsection, where $\Gamma_2(L, g) = \Gamma_2'(L', g')$, then all attacks may be regarded as direct.

Assuming the indistinguishability of Goppa codes, direct attacks basically mean direct decoding attacks on a random binary linear code.

McEliece [30] described a basic brute force direct attack: given a generator matrix $\hat{G}$ for a code $\mathcal{C}$ and a vector $c = m\hat{G} + e$, one searches through all the codewords $c' \in \mathcal{C}$, checking whether $d(c', c) = wt(e)$.

Another possible method is statistical decoding due to Al Jabri (see [16]). As summarized by Engelbert et al. [16], statistical decoding is based on the observation that vectors belonging to a code dual to $\mathcal{C}$ which are not orthogonal to the vector $c = m\hat{G} + e$ reveal information about the error positions in e. This method is problematic because not much can, in general, be derived about the properties of a code given just its dual. In particular, finding dual vectors fitting the above description and having small weights turns out to be tricky.

The best methods known for decoding random-looking linear codes are based on information-set decoding.

As explained in [36, 1], given $c = m\hat{G} + e$, the main idea is to guess a set of coordinates in c that are error-free and correspond to an invertible submatrix of $\hat{G}$. If the guess is correct, the attacker can multiply c by the inverse of the submatrix and reveal the corresponding parts of the plaintext.

First described by Prange in 1962 (see, e.g. [36]), this method was also mentioned in McEliece's original paper [30]. Information-set decoding subsequently received a lot of attention. Lee and Brickell offered a systematized version from the general-decoding-problem point of view and Leon and Stern and more recently Bernstein, Lange and Peters and Finiasz and Sendrier substantially improved the method by providing algorithms to solve the problem of finding weights (see, e.g. [16, 36, 34]). The attacks in the latter thread of research searching for minimum weight codewords all have the following pattern [16]: given a code $\mathcal{C}$, one first searches for codewords of a small weight within a restriction of $\mathcal{C}$, generated by a submatrix of a generator matrix for $\mathcal{C}$. One then identifies the corresponding codewords belonging to $\mathcal{C}$ and checks their weight. The algorithm is repeated until a suitable codeword is found. For further details and information about the information-set decoding attacks, see [36, 16, 34, 1].

Although the information-set decoding algorithms represent a substantial improvement over exhaustive search or any other alternative method for decoding a random code, the running time of these algorithms remains exponential in the code length [1]. Hence, there are no known attacks on McEliece or Niederreiter with subexponential complexity [16].

Conversions. Similarly as a naive implementation of RSA, neither the implementations of McEliece and Niederreiter schemes as described in 3.1.1 and 3.1.2, respectively, attain the IND-CCA2 security. For example, when resending the same plaintext m in McEliece twice, one obtains two ciphertexts

$$c_1 = m\hat{G} + e_1$$

and

$$c_2 = m\hat{G} + e_2$$

from which one recovers much information about the low-weight error vectors, namely, $c_1 + c_2 = e_1 + e_2$ [16]. As further discussed in [16], such an implementation is also prone to malleability, since adding a linear combination of rows from the generator matrix $\hat{G}$ to any ciphertext c produces another valid ciphertext; and, prone to reaction attacks. In these, a man in the middle finds error-free coordinates by swapping bits in a captured ciphertext and observing whether the recipient is able to decrypt the modified ciphertext (i.e. the changed position was erroneous) or asks the ciphertext to be resent (i.e. the swapped bit added an extra error, and hence made the ciphertext undecodable). To prevent the above kinds of attacks, we need to destroy dependencies between the plaintexts and the corresponding ciphertexts.

Conversions providing the IND-CCA2 security for the code-based cryptosystems were reviewed by Kobara and Imai [26]. They discuss the, so-called, Pointcheval and Fujisaki and Okamoto conversions as well as three other new conversions for the use with the McEliece cryptosystems (see [16, 26] for conversion reviews). Engelbert et al. [16] summarize that "breaking indistinguishability in the CCA2 model using any of the conversions presented [by Kobara and Imai], is as hard as breaking the original McEliece public key system". Also, the conversions do not cause significant increase in the running time of the cryptosystem, and even reduce the public key size.

Chapter 4

Code-based signing

Already in 1978, when publishing the cryptosystem (3.1.1), McEliece [30] noted that the scheme may not be suitable for creating digital signatures. His claim turned out to be reasonably justified. It took over a decade before the first code-based signing scheme was proposed by Xinmei (see [16]). However, the scheme was attacked and proven insecure just two years later in 1992 ([16]). Other code-based constructions, such as an identification scheme and a pseudorandom number generator were successfully developed (see, e.g. [16, 34]), but there did not seem to be any advance in the field of signatures. All proposals made in the 90's were either broken or impractical (see [16, 11, 34]).

Finally, in 2001, Courtois, Finiasz and Sendrier [14] designed the first working code-based signature scheme known as CFS. The scheme is derived from the Niederreiter cryptosystem and based on the syndrome decoding problem. Its signature size is exceptionally small and verification time very fast. On the other hand, the signature time and size of the public key in CFS are relatively large and, even worse, exponential in the scheme's parameters. Bleichenbacher came up with an innovative attack on the CFS scheme that was later published by Finiasz and Sendrier [20] in 2009. The attack necessitated a moderate increase in parameters, but, due to the exponential scaling, it made the CFS scheme almost impractical. In 2011, Finiasz [19] suggested a remedy for this problem: a new version of the signature scheme, called Parallel-CFS, which has similar costs and sizes as the original CFS but which is resistant to the Bleichenbacher attack, so that the original smaller parameters could securely be used again.

The main unresolved shortcoming of the CFS family of signatures has been the lack of a formal security proof. In 2007, Dallot [15] modified the original CFS scheme into a so-called mCFS scheme and proved that in the random oracle model, the mCFS signature scheme is existentially unforgeable under chosen message attack. Dallot's proof rests on two assumptions, namely, on the Goppa code indistinguishability problem 2.3.7 and the Bounded syndrome decoding problem 2.3.6 both being hard. In 2010, Faugere et al. [17] built a distinguisher capable of distinguishing random matrices from matrices corresponding to a Goppa code, provided that the rate of the code is sufficiently high. Although no specific distinguishing attack has been mounted against the original CFS, mCFS or Parallel-CFS scheme versions, the existence of the distinguisher invalidated the security proof and suggested that any provably secure CFS-like signature scheme must use Goppa codes of lower rates. This, however, would dramatically change the range of the allowed CFS parameter values and would render the scheme impractical.

In this chapter, we describe in detail the above developments and complications regarding the CFS family of signatures and add a discussion of a very new CFS-altered construct published by Preetha et al. [38]. The authors of [38] claim that their signature scheme is practical while at the same time provably secure in the random oracle model.

Finally, we briefly review the non-CFS branches of code-based signatures' research and proposed signature schemes with additional properties, such as ring signatures or blind signatures.

4.1 Basic definitions and requirements

Definition 4.1.1. *A* (digital) signature scheme *is a collection of algorithms* $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ *where*

$$\mathcal{G}(1^l) = (PubKey, PrivKey),$$

$$\mathcal{S}(m, PrivKey) = \tau,$$

$$\mathcal{V}(m, \tau, PubKey) = \begin{cases} true \\ false \end{cases} .$$

More precisely,

- $\mathcal{G}$ *is a randomised key-generation algorithm that, given a security parameter* l*, outputs a pair of a public key and a corresponding private key,*

- *$\mathcal{S}$ is a (possibly randomised) signing algorithm that, given the message m to sign and a private key, outputs the corresponding signature τ, and*

- *$\mathcal{V}$ is a verification algorithm that, given the public key, message and a signature to verify, outputs either* true *and accepts the signature as valid, or $false$ and declines the signature.*

Moreover, every signature scheme must satisfy the following correctness requirement: for all messages m and key pairs $(PubKey, PrivKey)$ generated by $\mathcal{G}$, if $\mathcal{S}(m, PrivKey) = \tau$ then $\mathcal{V}(m, \tau, PubKey) = true$.

Definition 4.1.2. *A signature scheme is said to be* ϵ-existentially unforgeable under an adaptive chosen message attack*, denoted ϵ-EUF-CMA, if every polynomial-time algorithm $\mathcal{A}$ has a chance $\leq \epsilon$ of winning in the following game:*

1. *The challenger generates the key pair $\mathcal{G}(1^l) = (PubKey, PrivKey)$.*

2. *$PubKey$ is given to $\mathcal{A}$, while $PrivKey$ is kept secret.*

3. *$\mathcal{A}$ may finitely many times adaptively ask for a signature of a message m of $\mathcal{A}$'s choice and obtains a corresponding valid signature τ from the challenger.*

4. *$\mathcal{A}$ produces a message-signature pair (m', τ').*

$\mathcal{A}$ wins the game if the pair (m', τ') is different from all the pairs (m, τ) obtained in Step 3 from the challenger and if $\mathcal{V}(m, \tau, PubKey) = true$.

Definition 4.1.3. *A signature scheme is called ϵ-secure if it is ϵ-existentially unforgeable under an adaptive chosen message attack.*

4.2 Difficulties with code-based signatures

A usual way of creating a signature from a known public-key cryptosystem is by using the so-called “hash-then-decrypt” paradigm [15, 14]. The idea is as follows.

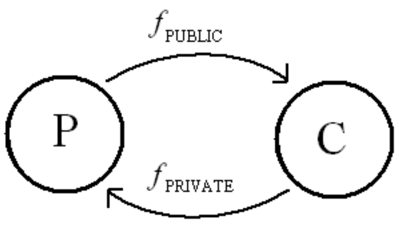
The given public-key cryptosystem may be seen as consisting of two sets, P of plaintexts and C of ciphertexts and two functions, one private and one public, where

$$f_{\text{PUBLIC}} : P \to C$$

and

$$f_{\text{PRIVATE}} : C \to P.$$

The public function f_{PUBLIC} may be used by anyone, while f_{PRIVATE} is kept secret. In the cryptosystem, the (possibly hashed) messages are elements of the set P, f_{PUBLIC} carries out the encryption and f_{PRIVATE} corresponds to the decryption. The situation is depicted in Figure 4.1.



.

Figure 4.1: Hash-then-decrypt paradigm.

If, for all elements $c \in C$, we have

$$f_{\text{PUBLIC}}(f_{\text{PRIVATE}}(c)) = c,$$

we may reverse the application of the two functions and create a signature scheme. In this case, the (hashed) message space is the set C. The signer applies the secret f_{PRIVATE} to the message, thus obtaining its signature, an element $\tau := f_{\text{PRIVATE}}(m) \in P$. Finally, anyone can verify the validity of the signature by applying f_{PUBLIC} and accept the signature as valid if $f_{\text{PUBLIC}}(\tau) = m$.

A textbook example of the process just described is the creation of FDH RSA signature scheme from the RSA cryptosystem.

In the case of the Niederreiter cryptosystem with the parameters n, k, t, and the public parity-check matrix $\hat{H}$, we have

$$P = \{m \in \mathbb{F}_2^n | wt(m) = t\},$$

$$C = \{s \in \mathbb{F}_2^{n-k} | s \text{ is a decodable syndrome}\},$$

f_{PUBLIC} is the multiplication by the matrix $\hat{H}$ and f_{PRIVATE} is the secret process enabling syndrome decoding.

And here is the problem: for the Goppa codes, there does not exist an efficient way of describing the set C [19]. The syndromes clearly are binary vectors of length $n-k$, but given an $s \in \mathbb{F}_2^{n-k}$ at random, we do not know whether it is decodable, i.e. whether there exists $e \in \mathbb{F}_2^n$ with $wt(e) \leq t$ such that $s^T = \hat{H}e^T$, unless we actually try to decode s and the process fails.

This does not cause any problem in the cryptosystem, since, any element of $\mathbb{F}_2^{n-k}$ hit by encryption is, by construction, decodable. However, if we want to reverse the process to create a signature scheme and we pick a syndrome $s \in \mathbb{F}_2^{n-k}$ at random, what is the probability that it is decodable, i.e. that it lies in the domain of f_{PRIVATE}?

Courtois et al. [14] compute the density of the decodable syndromes in a t-error correcting binary irreducible Goppa code $\Gamma_q(L,g)$ of length $n = 2^m$ and dimension $k = n - tm$. The number $\mathcal{N}_{dec}$ of decodable syndromes is equal to the number of words $e \in \mathbb{F}_2^n$ with $wt(e) \leq t$ and so we have

$$\mathcal{N}_{dec} = \sum_{i=0}^{t} \binom{n}{i} \approx \binom{n}{t} \approx \frac{n^t}{t!} = \frac{2^{tm}}{t!}$$

since $n \gg t$. The number of all syndromes, on the other hand, is $\mathcal{N}_{tot} = 2^{n-k} = 2^{tm}$. Hence, the probability that a randomly chosen syndrome is decodable is

$$\frac{\mathcal{N}_{dec}}{\mathcal{N}_{tot}} \approx \frac{1}{t!}.$$

A similar situation occurs when one starts with the McEliece cryptosystem and considers the probability of a randomly chosen word $w \in \mathbb{F}_2^n$ to be decodable to a codeword in $\Gamma_q(L,g)$.

Now, with the proposed parameters for the McEliece/Niederreiter cryptosystems (see Section 3.2) in which $t \geq 50$, the average number of trials until one hits a decodable syndrome/word at random would be $\geq 50! \simeq 3 \cdot 10^{64}$. This makes the signing time of a signature scheme infeasible.

4.3 CFS signature scheme

The authors of the first practical signature scheme, Courtois et al. [14], notice the above problem and reconsider the parameter choices for the underlying Goppa codes. They decide to pick a class of t-error correcting binary irreducible Goppa codes with length $n = 2^m$ and dimension $k = n - tm$[1]. A signature scheme is then parametrised by m and t. For a given

[1]This used to be a typical choice; Goppa codes of length other than 2^m have started to be used only recently.

pair (m, t), there exist about $2^{mt}/t$ different Goppa codes satisfying the above relations to choose from ([14]).

Courtois et al. [14] accept the fact that, on average, $t!$ decoding attempts must be made in order to successfully decode a random syndrome in $\mathbb{F}_2^{n-k}$. In order for the signing time to be practical, then, t must be chosen very small. On the other hand, if the code is to achieve the same level of security as the codes used in the McEliece/Niederreiter cryptosystems (where, e.g. $n = 1024$ and $t = 50$), the code's length n must be taken very large. Courtois et al. propose the values $(n, t) = (2^{16}, 9)$ or $(n, t) = (2^{15}, 10)$. Luckily, as we see later on, such a large n turns out to be compatible with a practical signature scheme, and, although it affects the size of the signatures public key, this size stays reasonable.

We now give a description of two basic versions of the CFS signature scheme [14, 19].

Definition 4.3.1 (CFS signature scheme – counter version)**.**
Public parameters: *A public cryptographic hash function* $h : \{0,1\}^* \to \mathbb{F}_2^{n-k}$*, from the message space* $\{0,1\}^*$ *to the syndrome space* $\mathbb{F}_2^{n-k}$*. Integers* m, t *such that there is a* t*-error correcting binary irreducible Goppa code of length* $n = 2^m$ *and dimension* $k = n - tm$*. Denote the set of all such codes by* S*.*

Setup: *Select an* $(n-k) \times k$ *parity-check matrix* H *in systematic form for a random* $\Gamma_2(L, g) \in S$*. Select a random* $(n-k) \times (n-k)$ *invertible matrix* M *and a random* $n \times n$ *permutation matrix* P*.*

Private key: *Goppa polynomial* $g(X) \in \mathbb{F}_{2^m}[X]$ *and support* $L \in \mathbb{F}_{2^m}^n$*, the parity-check matrix* $H \in \mathbb{F}_2^{(n-k)\times n}$ *for* $\Gamma_2(L, g)$*, matrices* $M \in \mathbb{F}_2^{(n-k)\times(n-k)}$ *and* $P \in \mathbb{F}_2^{n\times n}$*.*

Public key: *The* $(n-k) \times n$ *matrix* $\hat{H} := MHP$*.*

Signing:

1. *Given a message* $\mathrm{m} \in \{0,1\}^*$*, create a counter* i*, set* $i = 0$ *and, by concatenating, create* $(\mathrm{m}|i)$*.*

2. *Compute* $h(\mathrm{m}|i) \in \mathbb{F}_2^{n-k}$*.*

3. *Apply the Niederreiter's decryption algorithm from Definition 3.1.2 to* $h(\mathrm{m}|i)$*.*

4. *If Step 3 fails, set $i = i + 1$ and go to Step 2. Otherwise, output the signature $\tau = (i, e)$, where $e \in \mathbb{F}_2^n$ is the unique vector of weight $\leq t$ with $\hat{H}e^T = h(\mathrm{m}|i)^T$ that was found in Step 3.*

Verification: *Given a message* $\mathrm{m} \in \{0,1\}^*$ *and a signature* $\tau = (i, e)$, *compute* $s_1 = h(\mathrm{m}|i)$ *and* $s_2^T = \hat{H}e^T$. *If* $s_1 = s_2$, *return true, otherwise, return false.*

In the given CFS description, the successive random choice of syndromes in $\mathbb{F}_2^{n-k}$ is ensured by introducing a hash function h and appending an increasing counter to the message so that the input to h changes with every signing attempt and a new syndrome in $\mathbb{F}_2^{n-k}$ is picked uniformly at random. The rest of the signature may be seen as the reversed Niederreiter system, exactly following the "hash-then-decrypt" paradigm explained in Section 4.2. We refer to the above version of the signature scheme as the "CFS-counter version".

Apart from this, Courtois et al. [14] also vaguely mention, and Finiasz [19] explicitly discusses another version of the CFS, namely the "CFS-complete decoding version". As the name suggests, the idea is to extend the decoding algorithm from Step 3 above so that (almost) any element of $\mathbb{F}_2^{n-k}$ is decodable. One may then omit the counter and only hash a given message. The authors find the smallest $\delta > 0$ such that $\binom{n}{t+\delta} > 2^{n-k}$ and note that most of the syndromes in $\mathbb{F}_2^{n-k}$ must be decodable into vectors in $\mathbb{F}_2^n$ of weight at most $t + \delta$. The original decoding algorithm is then combined with an exhaustive search for the extra δ non-zero positions. The signature scheme becomes as follows (taken from [19]).

Definition 4.3.2 (CFS signature scheme – complete decoding version)**.**
Public parameters: *All parameters as in Scheme 4.3.1 and a* $\delta > 0$ *such that* $\binom{n}{t+\delta} > 2^{n-k}$.

Setup: *Same as in Scheme 4.3.1.*

Signing:

1. *Given a message* $\mathrm{m} \in \{0,1\}^*$, *compute* $h(\mathrm{m}) \in \mathbb{F}_2^{n-k}$.
2. *Pick a vector* $w \in \mathbb{F}_2^n$ *with* $wt(w) = \delta$.
3. *Apply the Niederreiter's decryption algorithm from 3.1.2 to* $h(\mathrm{m})^T + \hat{H}w^T$.

4. *If Step 3 fails, go to Step 2. Otherwise, output the signature* $\tau = (w+e)$*, where* $e \in \mathbb{F}_2^n$ *is the unique vector of weight* $\leq t$ *with* $\hat{H}e^T = h(\mathrm{m})^T + \hat{H}w^T$ *that was found in Step 3.*

Verification: *Given a message* $\mathrm{m} \in \{0,1\}^*$ *and a signature* $\tau = (e+w)$*, compute* $s_1 = h(\mathrm{m})$ *and* $s_2^T = \hat{H}(e+w)^T$*. If* $s_1 = s_2$*, return true, otherwise, return* $false$*.*

Notice that, similarly to the counter version, the complete decoding CFS requires, on average, $t!$ decoding attempts. This is because in Step 3 we basically search through different syndromes in $\mathbb{F}_2^{n-k}$ looking for one that is decodable into a vector of weight t.

The complete decoding CFS version has a slight disadvantage of not being able to sign some small number of messages at all, as there is always a very small probability that a given syndrome $h(\mathrm{m})$ can only be decoded into a vector of weight greater than $t+\delta$. Finiasz [19] then suggests to modify the message and try to sign it again.

To conclude this section, we sum up the main ideas of [14] making the CFS the first successful signature scheme. The authors

- picked a new set of parameters for the underlying Goppa codes,
- came up with a way of sampling random syndromes in $\mathbb{F}_2^{n-k}$ (either by introducing a counter or the complete decoding), and
- applied the "hash-then-decrypt" paradigm to the Niederreiter cryptosystem.

Finally, Curtois et al. [14] remark that the CFS scheme is purposely based on the Niederreiter cryptosystem, rather than on the McEliece variant. The reason for this is that in the Niederreiter scheme an efficient signature size compression is possible, while this is not the case in the McEliece system. Notice that in the Schemes 4.3.1 and 4.3.2 the vector e has length n and $wt(e) \leq t$ where $n \gg t$. Thus, as explained in [14], by indexing all the $\binom{n}{t}$ possibilities for e, one only needs about $\log_2 \binom{n}{t} \ll n$ bits to store e. For the proposed values $(n,t) = (2^{16}, 9)$, this translates into $\log_2 \binom{n}{t} \approx 126$ bits. On the other hand, a hypothetical signature scheme created from the McEliece system would have as part of the signature a binary word e of length $k = n - tm$ (so that for a public McEliece matrix $\hat{G}$, $e\hat{G}$ would be a codeword in the underlying Goppa code). Such a word contains no redundancy and cannot be compressed. For the above parameters, e would thus require about $2^{16} - 9 \cdot 16 = 65392$ bits.

4.4 CFS parameters and performance

We now analyze the parameters and performance of the CFS scheme in greater detail. We use the notation from Section 4.3. In particular, $n = 2^m$ and $k = n - tm$.

Key sizes. The CFS scheme needs the same kind of information for its public/private key as the Niederreiter cryptosystem. By using the expressions for the keys given in Section 3.2 and substituting $n = 2^m$ and $k = n - tm$, we obtain that the CFS private key has size $O(t^2m^2 + 2^m m) = O((n-k)^2 + n\log_2 n)$ bits. Note that this is a rather generous estimate, since, similarly as in the Niederreiter's scheme, the decoding process would work also without the $(mt \times 2^m)$ parity-check matrix H.

CFS's public key consists of the $(mt \times 2^m)$ matrix $\hat{H}$ and so $O(2^m mt) = O((n-k)n)$ bits are needed to store the public key.

Signature cost. As already explained above, $t!$ decoding attempts are needed on average before one obtains a valid decoding of a syndrome in $\mathbb{F}_2^{n-k}$. The hardest part of the decoding algorithm, namely, testing whether the error locator polynomial splits into linear factors, can be done in about t^2m^3 binary operations [14], and so the overall signing time is $O(t!t^2m^3) = O(t!(n-k)^2\log_2 n)$ operations.

Signature length. Because of the average $t!$ attempts needed for signing, the CFS counter version with the signature $\tau = (i, e)$ typically has $i \approx t!$. As discussed in Section 4.3, $\log_2 \binom{n}{t}$ bits are needed to store e. Altogether, therefore, the CFS counter version signature is about $O\left(\log_2 \binom{n}{t} + \log_2 t!\right) = O(\log_2(n^t))$ bits long.

The signature $\tau = (w + e)$ in the CFS with complete decoding consists of a single vector of length n and weight $t + \delta$. Hence, the signature has $O\left(\log_2 \binom{n}{t+\delta}\right)$ bits.

Verification cost. Verification consists of summing up t columns of the $(mt \times 2^m)$ matrix $\hat{H}$ in the CFS counter version (or $t + \delta$ columns in the CFS complete decoding version), i.e. it takes about $O(mt^2) = O((n-k)t)$ binary operations.

Table 4.1 summarizes the above performance estimates for the CFS scheme and compares them to their counterparts in the Niederreiter cryptosystem. Since the CFS scheme is, basically, an inverted Niederreiter system, the respective values for the two schemes are almost identical. The only difference occurs in the case of the decryption cost in the Niederreiter system and the signing times in the CFS scheme. The latter contains a multiple of $t!$ since $t!$ decoding attempts must be, on average, made in the signing process. Moreover, the individual decoding attempts in the CFS scheme also take less than in the case of the Niederreiter scheme ($O((n-k)^2\log_2 n)$ as opposed to $O(n(n-k)\log_2 n)$). Courtois et al. [14] explain that this is because in these attempts one does not have to explicitly find the roots of the error locator polynomial, but only to check whether the polynomial splits into linear factors.

	Niederreiter		CFS
public key size	$(n-k)n$	public key size	$(n-k)n$
private key size	$(n-k)^2+n\log_2 n$	private key size	$(n-k)^2+n\log_2 n$
encryption cost	$(n-k)t$	verification cost	$(n-k)t$
decryption cost	$n(n-k)\log_2 n$	signature cost	$t!(n-k)^2\log_2 n$
		signature length	$\log_2 n^t$

Table 4.1: Niederreiter and CFS scheme performance comparison. All data represent the big-O values.

Proposed parameters. At the time of CFS's creation, the counter version was preferred and the proposed parameters were $(n,t)=(2^{16},9)$ or $(2^{15},10)$, both corresponding at the time to a security level of about 80 bits [14].

Taking into account the developing attacks, currently the so-called "parallel" CFS version with complete decoding is recommended. In Section 4.5 we provide a detailed description of the parallel version of the CFS as well as the cryptoanalysis explaining why this version is secure against all known attacks. For now we only remark that the parallel CFS scheme behaves, basically, as the simple CFS scheme with the exception that, when signing, two or three signatures are produced in parallel. The signature cost, length and the verification time then multiply accordingly.

Finiasz [19] provides a variety of up-to-date parameters (n,t,δ) together with a required number i of parallel signatures that are suitable for the use in the parallel CFS scheme and offer a roughly 80-bit security level. A sample of these parameters is stated in Table 4.2. Table 4.3 then gives the actual performance data of the CFS scheme for the parameter sets from Table 4.2 [19].

security level	n	t	δ	i (no. of parallel signatures)
80-bit	2^{17}	10	2	2
83-bit	2^{18}	9	2	3
82-bit	2^{20}	8	2	3

Table 4.2: Some recommended parameters for the CFS scheme (taken from [19]).

parameters (n, t, δ, i)	$(2^{17}, 10, 2, 2)$	$(2^{18}, 9, 2, 3)$	$(2^{20}, 8, 2, 3)$
public key size in MB	2.7	5	20
private key size in MB	0.26	0.56	2.5
verification cost in binary op.	3400	4374	3840
signature cost in binary op.	2^{23}	2^{20}	2^{17}
signature length in bits	196	288	294

Table 4.3: Performance data for the CFS scheme with up-to-date parameters offering 80-bit security levels (taken from [19]).

Finally, one may want to compare these data with the performance data of the Niederreiter cryptosystem offering the same security level. Recall from Section 3.2 that the suitable parameters for the 80-bit security level in the Niederreiter system are $(n, k, t) = (2048, 1751, 27)$ without list decoding and $(n, k, t) = (1632, 1269, 33)$ with list decoding [7]. Table 4.4 gives the performance data for these parameters.

parameters (n, k, t)	$(2048, 1751, 27)$	$(1632, 1269, 33)$
public key size in KB	74	72
private key size in KB	13	18
encryption cost in binary op.	8019	11979
decryption cost in binary op.	2^{23}	2^{21}

Table 4.4: Performance data for the Niederreiter scheme with up-to-date parameters offering 80-bit security levels.

We see that even though in Table 4.1 the performance of the Niederreiter and the CFS scheme depends on the input parameters almost identically, the actual performance data for the same security level differ. This is because the two schemes use different parameter sets. For the 80-bit security level, the public and the private keys in the Niederreiter cryptosystem are much smaller than their counterparts in the CFS scheme. Indeed, while the latter keys have sizes of order of magnitude in megabytes, the former keys only have a couple of dozens of kilobytes. The encryption cost in the Niederreiter system is about three times larger than the verification cost in the CFS scheme. Finally, the Niederreiter decryption cost and CFS signature cost are about the same.

Evaluation of the CFS scheme. The CFS scheme produces one of the shortest digital signatures known [19]. It also has a very fast signature verification. On the other hand, the signature time and the size of the keys are all relatively large. The signing time does not scale well with t and is only practical for Goppa codes with small error-correcting capability and large length n, i.e. for codes whose information rate $\frac{k}{n} = \frac{2^m - tm}{2^m}$ is very high. The key sizes are exponential in m. Hence, even a slight increase in the CFS's parameters m, t may be detrimental to the signature scheme.

Possible trade-offs in performance. Courtois et al. [14] describe two ways of realizing a trade-off between the signature length and verification time, enabling thus customization of the signature scheme to one's needs.

Method 1 is to omit a small number of non-zero components from the signature at the expense of an exhaustive search during the verification. More precisely, instead of sending a vector $e \in \mathbb{F}_2^n$ of weight t (or $t + \delta$), one removes w many ones and sends a vector $u \in \mathbb{F}_2^n$ with $wt(u) \leq t - w$ (or $wt(u) \leq t + \delta - w$ in the complete decoding version). When verifying, the receiver computes $s^T = h(\mathrm{m}|i)^T + \hat{H}u^T$ (or $s^T = h(\mathrm{m})^T + \hat{H}u^T$) and accepts the signature as valid if he is able to find w columns of $\hat{H}$ that sum up to s.

Method 2 will be illustrated on the CFS counter version. Applying it to the CFS with complete decoding mostly consists of exchanging t for $(t + \delta)$ in what follows.

We start with a notion of a punctured code.

Definition 4.4.1. *Given a linear code C of length n and a vector $v \in \mathbb{F}_2^n$ with $wt(v) = p$ for some integer $0 \leq p \leq n$, the* punctured code C on positions of v *is a linear code C' of length $n - p$ created from C by removing the coordinates corresponding to the non-zero components in v from each codeword.*

In accordance with Curtois et al. [14], we assume that the dimensions of the punctured and the original code are equal.

Lemma 4.4.2. *Let C' be a linear code obtained from an $[n,k]$-code C by puncturing the first p positions. Let H be a parity-check matrix for C, I a $p \times p$ identity matrix and U an $(n-k)\times(n-k)$ invertible matrix such that*

$$UH = \begin{pmatrix} I & R \\ 0 & H' \end{pmatrix}.$$

Then H' is a parity-check matrix for C'.

Proof. Let G be a generator matrix for C. Then, the restriction of G to its last $(n-p)$ columns forms a generator matrix G' for the punctured code C'. We write $G = (F|G')$. Using the identity $GH^T = 0$, we obtain

$$0 = GH^TU^T = G(UH)^T = (F|G')\begin{pmatrix} I & 0 \\ R^T & H'^T \end{pmatrix} = \begin{pmatrix} F & 0 \\ G'R^T & G'H'^T \end{pmatrix}.$$

Hence, $G'H'^T = 0$. Since C' has dimension k and the rank of H' is $n-k$, exactly the code C' is in the kernel of H'^T. Hence, H' is a parity-check matrix for the code C'.

□

Lemma 4.4.3. *Assume the notation from Lemma 4.4.2 and let $w < t$ be a small integer. Then, the following two problems are equivalent.*

1. *For a given syndrome $s \in \mathbb{F}_2^{n-k}$, find $z \in \mathbb{F}_2^n$ satisfying $Hz^T = s^T$ and such that the first p coordinates of z contain at most $t-w$ ones and the last $n-p$ coordinates of z contain at most w ones.*
2. *For given $s' \in \mathbb{F}_2^{n-k-p}$ and $s'' \in \mathbb{F}_2^p$, find $z' \in \mathbb{F}_2^{n-p}$ and $z'' \in \mathbb{F}_2^p$ satisfying $wt(z') \leq w$, $H'z'^T = s'^T$ and $wt(z'') = wt(Rz'^T + s'') \leq t-w$.*

Proof. Assume that we can solve Problem 1 and are given some $s' \in \mathbb{F}_2^{n-k-p}$ and $s'' \in \mathbb{F}_2^p$. Let $s^T = U^{-1}(s''|s')^T$ and find z solving the Problem 1. Let $z' \in \mathbb{F}_2^{n-p}$ and $z'' \in \mathbb{F}_2^p$ be such that $z = (z''|z')$. Notice that by the assumption on the positions of ones in z, we have $wt(z') \leq w$ and $wt(z'') \leq t-w$. Then,

$$(s''|s')^T = Us^T = UHz^T = \begin{pmatrix} I & R \\ 0 & H' \end{pmatrix}(z''|z')^T = \begin{pmatrix} z''^T + Rz'^T \\ H'z'^T \end{pmatrix}.$$

Hence, $H'z'^T = s'^T$ and $z''^T + Rz'^T = s''^T$, i.e. $z'' = Rz'^T + s''$.

The converse is similar.

□

The idea of the Method 2 trade-off proposal in [14] is to use Lemma 4.4.3 and translate an instance of the syndrome decoding problem in the $[n, k]-$code C into a problem involving syndrome decoding in the shorter punctured code C'. In other words, the signer can send the $t - w$ errors as before, but instead of the verifier searching for the remaining errors in the code C, he may search for them in the shorter code C'. In order to cut down on the length of the code C' as much as possible and to make thus the trade-off more advantageous, Courtois et al. [14] propose to split the coordinates of the error vector $e \in \mathbb{F}_2^n$ obtained at the end of the CFS signing process into n/l sets each of length l. The signature in Method 2 then consists of $(t - w)$ of these sets each containing an error and the counter i.

The verifier creates a code C' by puncturing the public $[n, k]$-code C in the $l \cdot (t-w)$ positions corresponding to the transmitted sets. He performs a Gaussian elimination on the public key $\hat{H}$ to find a $(n-k-l(t-w)) \times (n-l(t-w))$ parity-check matrix H' for C' and an invertible $(n-k) \times (n-k)$ matrix U such that

$$U\hat{H} = \begin{pmatrix} I & R \\ 0 & H' \end{pmatrix}$$

for some matrix R and an $l(t-w) \times l(t-w)$ identity matrix I. The verifier lets $s' \in \mathbb{F}_2^{n-k-l(t-w)}$ and $s'' \in \mathbb{F}_2^{l(t-w)}$ such that, for the received message m, counter i and the used hash function h,

$$U(h(\mathrm{m}|i))^T = (s''|s')^T.$$

He then conducts an exhaustive search, solving an instance of a syndrome decoding problem in C' for the syndrome s'. By Lemma 4.4.3, the signature is valid if he can find a vector $z' \in \mathbb{F}_2^{n-l(t-w)}$ with $wt(z') \leq w$ such that

$$s'^T = H'z'^T$$

and

$$wt(Rz'^T + s'') \leq t - w. \tag{4.1}$$

The length of the signature in Method 2 is $\log_2 \left(\binom{n/l}{t-w} t!\right)$ instead of the original $\log_2 \left(\binom{n}{t} t!\right)$. The larger the value of l is, the shorter the signature and also the code C' becomes, but the longer it takes to find the z' satisfying Inequality (4.1). As explained in [14], the best compromise seems to take $l = m$ (i.e. for the code parameters $(n, t) = (2^{17}, 10)$ it would be $l = 17$).

Courtois et al. [14] note that the cost of the extra Gaussian elimination needed to compute matrices H' and U in the verification process is about $2^{m-1}tm$ column operations. Finally, the authors find that the best overall signature length/verification time trade-off, when using METHOD 2, is for $w = 3$.

Based on the two methods of a trade-off just presented, the authors of [14, 19] propose two options, both applicable to the CFS with the counter as well as with the complete decoding. Namely:

- If aiming for fast verification, setting $w = 1$ and using METHOD 1 is recommended.
- If aiming for short signatures, setting $w = 3$ and using METHOD 2 is recommended.

4.5 CFS scheme: security analysis

Let us start the security analysis of the CFS signature scheme with a minor remark made in [14]. Since the scheme's signature is exceptionally short, the reader may wonder whether a valid message-signature pair cannot be generated by a simple birthday paradox attack. For a typical signature scheme, mapping from a hashed message space M to a signature space S, *every element* $\mathrm{m} \in M$ has a valid corresponding signature $s \in S$ (or more, if the scheme is randomised). The birthday attack then usually runs in the square root of $|S|$. In the case of the CFS scheme, however, only for a fraction of syndromes $\mathrm{m} \in M$ there exists an element $s \in S$ such that the pair (m, s) is a valid message-signature pair. If there exists such an s, it is unique. Altogether, there are $|S|$ valid message-signature pairs and $|M||S|$ possible pairs in total. The probability that a random pair (m, s) is valid is thus $1/|M|$. An attacker needs to generate $|M|$ pairs to obtain, on average, one valid message-signature pair. The birthday attack then consists of generating a list of $\sqrt{|M|}$ syndromes and of $\sqrt{|M|}$ signatures and combining them together. The complexity of the attack in the CFS case is thus $O(\sqrt{|M|})$ and is independent of the signature length. For parameters $(n, t) = (2^{17}, 10)$, the syndrome space is $\mathbb{F}_2^{17 \cdot 10}$ and the attack would require about 2^{85} binary operations.

We now consider the security of the CFS scheme more generally. What are the problems that the scheme's security relies on and what are the most efficient attacks against the CFS?

Being constructed by, basically, inverting the Niederreiter cryptosystem, much of the Niederreiter's cryptoanalysis applies to the CFS as well. Similarly as for the cryptosystem, also in the case of the CFS, some of the literature (see e.g. [19]) divides the known attacks into structural and decoding ones. The structural attacks consist of recovering the private key, i.e. the secret underlying Goppa code, given the public code. The direct decoding attacks, on the other hand, seek to produce signature forgeries. In our analysis, we concentrate on the latter category of attacks. The reasons for this are twofold. Firstly, even with the knowledge of a high-rate Goppa code distinguisher (discussed later on in Section 4.5), no efficient structural attacks against the CFS are currently known [19]. Secondly, the structural attacks may be seen as a subcategory of the direct attacks, since, if an attacker is able to recover the private key, he is also able to forge signatures.

Hard problems behind the CFS. The problem of forging a signature in CFS is at least as hard as producing a corresponding plaintext for some Niederreiter ciphertext. This, in turn, is equivalent to solving the McEliece Problem 3.2.1 for an arbitrary ciphertext. The security of the CFS scheme thus relies on the same assumptions as the McEliece problem. These are

- the hardness of the Bounded decoding problem 2.3.5 (or, equivalently, of the Bounded syndrome decoding problem 2.3.6), and
- the Goppa code indistinguishability problem 2.3.7.

For a reference, see [14, 11, 16, 34].

Bleichenbacher attack. From the above discussion we see that the securities of the Niederreiter/McEliece cryptosystem and the CFS scheme are closely related - any attack on the underlying problems may be applied to both. However, there does exist a class of attacks that is harmful to the CFS scheme but not so much to the cryptosystems.

The original attack is due to Bleichenbacher, but it was never published by him. It was first discussed by Finiasz and Sendrier [20]. The weakness of the CFS springs from the fact that, when signing a message m, many of the syndromes that may potentially be hit are decodable. Each of the syndromes corresponds to a distinct error vector which, together with a proper counter value, forms a distinct valid signature for m. Hence, to successfully forge a CFS signature for a given message, one needs to decode any syndrome out of several possible ones. Just for comparison, this is not the case for the McEliece/Niederreiter cryptosystems. For a given McEliece/Niederreiter ciphertext c, there always exists exactly one corresponding error pattern and decrypting c translates into decoding one particular syndrome. This is the reason why the Bleichenbacher attack is not applicable to the McEliece/Niederreiter cryptosystems.

Consider now the following algorithm, adapted from [34] with the help of [20].

Generalized birthday algorithm 4.5.1 (GBA).
Input: *Integers a, r such that $(a+1)|r$ and sets $\mathcal{L}_0, \ldots, \mathcal{L}_{2^a-1} \subseteq \mathbb{F}_2^r$, each of cardinality $2^{\frac{r}{a+1}}$ and picked uniformly at random.*

Output: *A set of 2^a pairwise distinct vectors $v_0 \in \mathcal{L}_0, \ldots, v_{2^a-1} \in \mathcal{L}_{2^a-1}$ such that $\sum_i v_i = 0$, or an error message.*

Algorithm:

1. *Set $i = 1$.*
2. *At ith repetition of this step, pairwise add the elements of lists $\mathcal{L}_{2j}$ and $\mathcal{L}_{2j+1}$ for $0 \leq j < 2^{a-i}$, creating thus 2^{a-i} lists $\mathcal{L}'_0, \ldots, \mathcal{L}'_{2^{a-i}-1}$ containing sums of 2^i vectors. In these lists, only keep vectors that start with $i \cdot \frac{r}{a+1}$ zeroes.*
3. *Increment i and repeat Step 2 until $i = a-1$, i.e. until only two lists $\mathcal{L}_1^*$, $\mathcal{L}_2^*$ remain. Both lists contain sums of 2^{a-1} vectors starting with $(a-1) \cdot \frac{r}{a+1}$ zeroes.*
4. *If any two elements $l_1 \in \mathcal{L}_1^*$ and $l_2 \in \mathcal{L}_2^*$ sum into an r-bit zero vector, output $l_1 + l_2$. Otherwise, output error message.*

The main idea of the above algorithm is to concentrate on subsets of the r bits and only carry out subsequent operations on the vectors for which a match on the subsets was found [20].

For a better understanding of the algorithm, let us explain the choice of particular values above, namely, the number of lists, their size and the number of the required zero bits at each step.

Assume that we want to carry out the sequence of steps from 4.5.1 in such a way that the expected number of r-bit zero vectors obtained in Step 4 is one. Additionally, require all lists encountered in the algorithm to have equal size.

Since the number of lists is reduced by two at every step, it clearly is desirable to start with 2^a lists. Let k be the number of new bit positions that are required to be zero at each new list merge. Then, assuming that the elements in the lists are random vectors from S and merging two lists $\mathcal{L}_1$ and $\mathcal{L}_2$, the expected size of the merged list is $2^{-k}|\mathcal{L}_1||\mathcal{L}_2|$. Since we require the size of the lists constant throughout the process, we must have $2^{-k}|\mathcal{L}_1|^2 = |\mathcal{L}_1|$. Thus, the size of each list, independent of the merges, is 2^k. Finally, the last two lists contain 2^k vectors, each of which has $r-(a-1)k$ possibly non-zero bits. In order to obtain, on average, one r-bit zero vector at the end, we must have

$$2^{-(r-(a-1)k)}2^k 2^k = 1,$$

i.e. $k = \frac{r}{a+1}$. Hence, we obtain exactly the values used in 4.5.1.

The algorithm thus performs, $2^a - 1$ merges and sorting of lists of size $2^{\frac{r}{a+1}}$. Its complexity is $O(\frac{r}{a}2^a 2^{\frac{r}{a+1}})$ [20].

The GBA was first designed by Wagner in 2002 [40] but not specifically for code-based cryptography. Bleichenbacher noticed that it could be modified to solve the following syndrome decoding problem with multiples instances.

Problem 4.5.2 (Bounded syndrome decoding with multiple instances)**.** *Given a binary linear t-error correcting $[n,k]$ code $\mathcal{C} \subset \mathbb{F}_2^n$, an $(n-k)\times n$ parity-check matrix H for $\mathcal{C}$ and a set $S \subseteq \mathbb{F}_2^{n-k}$ of syndromes, find $e \in \mathbb{F}_2^n$ with $wt(e) \leq t$ such that $He^T = s^T$ for some $s \in S$.*

Indeed, let the elements of the input sets $\mathcal{L}_0, \dots, \mathcal{L}_{2^a-1}$ in Algorithm 4.5.1 be chosen from the n columns of matrix H. In particular, we have $r = n-k$. Solving Problem 4.5.2 means finding $\leq t$ columns of matrix H that sum up to one of the elements in S. Let the list $\mathcal{L}_0$ consist of the syndromes from S. Pick a so that $2^a - 1 \leq t$ and $(a+1)|r$. Let the lists $\mathcal{L}_1, \dots, \mathcal{L}_{2^a-1}$ each contain $2^{\frac{r}{a+1}}$ sums of t_i columns of H, for $i = 1, \dots, 2^a-1$, such that $\sum_i t_i = t$. Running Algorithm 4.5.1 for this choice of lists then produces a solution to Problem 4.5.2.

Clearly, this instance differs from the "ideal" algorithm version 4.5.1. Are all the lists of equal size? How big is the given set S? And do there at all exist $2^{\frac{r}{a+1}}$ sums of t_i columns of H, i.e. is $2^{\frac{r}{a+1}} \leq \binom{n}{t_i}$ true? Also, how do we pick t_i? All these choices influence the overall complexity of the algorithm. A detailed analysis of all the possibilities is done by Sendrier et al. [20, 40]. For our purpose we just note that a simple estimate on the time complexity of the GBA applied to a syndrome decoding problem with multiple instances is $O(L\log(L))$, where L is the size of the largest list manipulated [20]. Similarly, the memory complexity is $O(M\log(M))$ for M the largest list needed to be stored throughout the process [20].

In the Bleichenbacher attack [20, 34, 19] on the CFS signature scheme with the original parameters $(n, t) = (2^{16}, 9)$ or $(2^{15}, 10)$, the attacker generates a list $\mathcal{L}_0$ of possible syndromes $S \subseteq \mathbb{F}_2^{mt}$ by computing hashes $h(\mathrm{m}|i)$ for a counter i and a given message m. He applies the attack with $a = 2$, i.e. with four initial lists, one of which is $\mathcal{L}_0$ and the other three consist of sums of $t_i \approx \frac{t}{3}$ columns of the CFS public matrix $\hat{H}$. The lists will contain about $2^{\frac{r}{a+1}} = 2^{\frac{mt}{3}}$ elements. The complexity of this existential forgery attack on CFS is thus about $O(\frac{mt}{3} \cdot 2^{\frac{mt}{3}})$ binary operations [19, 20].

The Bleichenbacher attack is the most powerful attack against the CFS signature scheme [34, 19, 20]. The earlier attacks were all based on the traditional information set decoding methods as applicable to cryptosystems and described in 3.2. These attacks run in time $O(2^{\frac{mt}{2}})$ [19]. The Bleichenbacher attack affected the security level of the originally proposed parameters $(n, t) = (2^{16}, 9)$ or $(2^{15}, 10)$ and, in order to maintain a security level of about 2^{80} binary operations, new values for (n, t) in the CFS scheme were suggested in [20]. These were $(2^{15}, 12)$, $(2^{19}, 11)$, $(2^{21}, 10)$, and others [11, 38, 20]. Although the parameter increase is relatively small, it translates into huge public key sizes and long signing times (as these are exponential in m and t) and makes the CFS signature scheme rather impractical.

Parallel CFS. A remedy for the CFS signature scheme is suggested by Finiasz [19]. The idea is to sign a given message m by producing several CFS signatures. Finiasz proposes to use distinct hash functions h_1 and h_2 and to build the signature in such a way that an existential forgery attack would need to decode a pair of linked syndromes.

Using the CFS-counter version 4.3.1 is, however, not possible for these purposes [19]. The reason is that the syndromes $h_1(\mathrm{m}|i_1)$ and $h_2(\mathrm{m}|i_2)$ are independent from each other due to the counters i_1, i_2. In other words, an attacker, after decoding the syndrome $h_1(\mathrm{m}|i_1)$ may freely alter i_2 and choose to decode any of the resulting syndromes $h_2(\mathrm{m}|i_2)$. An existential forgery then consists of simply repeating the Bleichenbacher attack twice. Finiasz [19] notes that one may require the signature to use the same counter value i. However, the signing time then increases from $t!$ to $(t!)^2$ and makes the signature scheme impractical.

We may try to use the CFS complete decoding version 4.3.2. We obtain the following signature scheme [19].

Definition 4.5.3 (CFS-Parallel signature scheme).
Public parameters: *All parameters as in Scheme 4.3.2, except that there*

are two public hash functions $h_1, h_2 : \{0,1\}^* \to \mathbb{F}_2^{n-k}$.

Setup: *Same as in Scheme 4.3.2.*

Signing: *Using the signing algorithm 4.3.2, sign a given message* $\mathrm{m} \in \{0,1\}^*$ *twice, once using the hash function* h_1 *and once* h_2 *(note that this, in general, uses two different vectors* w *in Step 2 of the algorithm 4.3.2). We obtain* τ' *and* τ'' *and the signature is* $\tau = (\tau', \tau'')$.

Verification: *Given a message* $\mathrm{m} \in \{0,1\}^*$ *and a signature* $\tau = (\tau', \tau'')$, *check whether* $h_1(\mathrm{m}) = \hat{H}\tau'^T$ *and* $h_2(\mathrm{m}) = \hat{H}\tau''^T$. *If so, return true, otherwise, return* $false$.

The key sizes in the CFS-Parallel remain the same, and the signature time and size, and the verification time are doubled as compared to the standard CFS.

The CFS complete decoding version may be interpreted as decoding *any* given syndrome in $\mathbb{F}_2^{n-k}$ into an error vector of weight $\leq t + \delta$. The two syndromes that need to be decoded in order to forge a signature for the CFS-Parallel are linked to each other and always come in pairs. Indeed, if an attacker decides to forge a signature for message m and decodes the syndrome $h_1(\mathrm{m})$, then, in order to carry out a successful forgery, he also has to decode the syndrome $h_2(\mathrm{m})$.

An attacker now has to chain the Bleichenbacher attack. First, by running multiple attacks, he collects many partial signatures τ_i'. These are the decodings of syndromes $h_1(\mathrm{m}_i)$ for the hash function h_1 and multiple messages m_i. He then computes a list $\mathcal{L}_0$ of the corresponding syndromes $h_2(\mathrm{m}_i)$, for each i. One last Bleichenbacher attack is applied with the list $\mathcal{L}_0$, producing a decoding of some syndrome $h_2(\mathrm{m}_I)$, and so a partial signature τ_I'' for some m_I. Then, $\tau = (\tau_I', \tau_I'')$ is a valid forged CFS-Parallel signature. The cost of this attack is $2L\log(L)$ with $L = 2^{\frac{3}{7}mt}$ [19].

Alternatively, as discussed in [19], one may try to forge a parallel CFS signature by considering it as a one single syndrome decoding problem: given a syndrome $(h_1(\mathrm{m})|h_2(\mathrm{m})) \in \mathbb{F}_2^{2(n-k)}$, find $(e_1|e_2) \in \mathbb{F}_2^{2n}$ with $wt(e_1), wt(e_2) \leq t + \delta$ such that

$$\begin{pmatrix} \hat{H} & 0 \\ 0 & \hat{H} \end{pmatrix} (e_1|e_2)^T = (h_1(\mathrm{m})|h_2(\mathrm{m}))^T.$$

This requires a single Bleichenbacher attack. However, all the parameters are doubled and so the cost of the attack is $O(L\log L)$ with $L = 2^{\frac{2mt}{3}}$ [19].

Hence, the most efficient attack against a parallel CFS scheme has complexity $O(2^{\frac{3mt}{7}})$. This is still smaller than a standard information set decoding attack against a CFS scheme with complexity $O(2^{\frac{mt}{2}})$, but, at the same time, it is a significant improvement compared to the Bleichenbacher attack with complexity $O(2^{\frac{mt}{3}})$. Finiasz [19] concludes that the parallel CFS may securely use parameters much smaller then the simple CFS. He proposes to use $(n, t, \delta) = (17, 10, 2)$, $(18, 9, 2)$, $(20, 8, 2)$ and others [19].

Finally, notice that the parallel CFS scheme may, in general, be extended to include j distinct hash functions and j parallel signatures. However, Finiasz [19] points out that setting $j > 3$ produces too long signature times and sizes to be advantageous.

Security proof and mCFS. In 2007, Dallot [15] realised that using a counter in the CFS scheme provides an attacker with some information. Indeed, if a received signature is $\tau = (i, e)$, then the attacker knows that for all values $0 \leq j < i$ the syndromes $h(\mathrm{m}|j)$ are not decodable. Dallot [15] thus proposes yet another variant of the CFS scheme, the modified mCFS signature scheme. From the simple CFS-counter algorithm 4.3.1, mCFS only differs in the counter i. Instead of increasing the value of i from 0 gradually by one until a decodable syndrome $h(\mathrm{m}|i)$ is found, the mCFS scheme picks i from the set $\{1, \ldots, 2^{n-k}\}$ uniformly at random, again, until a decodable syndrome $h(\mathrm{m}|i)$ is hit. The length of the mCFS signature slightly increases since $n-k$ instead of $\log_2(t!)$ bits, on average, are required to store the counter.

For the mCFS variant, Dallot [15] presents a formal security proof. Assuming the hardness of the Bounded syndrome decoding problem 2.3.6 and the Goppa code indistinguishability assumption 2.3.7, Dallot proves that, in the random oracle model, the mCFS scheme is existentially unforgeable under the chosen message attack. Unfortunately, it turns out that the Goppa code indistinguishability assumption does not hold for Goppa codes with parameters as used in the CFS [17]. This invalidates Dallot's proof.

Distinguisher. Faugere et al. [17] published in 2010 a polynomial-time distinguisher for Goppa (or, more generally, alternant) codes with sufficiently high information rates. We now explain their construction.

For a t-error-correcting $[n, k]$ binary linear code $\mathcal{C}$ with $n = 2^m$ and $k = n - mt$, let $G = (g_{ij}) = (I_k|P) \in \mathbb{F}_2^{k \times n}$ be its generator matrix in systematic form, where we have an $k \times (n-k)$ matrix $P = (p_{ij})$ with $1 \leq i \leq k$ and $k + 1 \leq j \leq n$. Let $x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_n) \in \mathbb{F}_{2^m}^n$

be vectors such that for a matrix

$$V_t(x,y) := \begin{pmatrix} y_1 & \cdots & y_n \\ y_1x_1 & \cdots & y_nx_n \\ \vdots & \ddots & \vdots \\ y_1x_1^{t-1} & \cdots & y_nx_n^{t-1} \end{pmatrix}$$

we have

$$V_t(x,y)G^T = 0. \tag{4.2}$$

In other words, we require $V_t(x,y)$ to be a parity-check matrix for $\mathcal{C}$. Recall from Section 2.2 that in Equation (2.1) the matrix

$$YZ = \begin{pmatrix} g^{-1}(\alpha_1) & g^{-1}(\alpha_2) & \cdots & g^{-1}(\alpha_n) \\ g^{-1}(\alpha_1)\cdot\alpha_1 & g^{-1}(\alpha_2)\cdot\alpha_2 & \cdots & g^{-1}(\alpha_n)\cdot\alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ g^{-1}(\alpha_1)\cdot\alpha_1^{t-1} & g^{-1}(\alpha_2)\cdot\alpha_2^{t-1} & \cdots & g^{-1}(\alpha_n)\cdot\alpha_n^{t-1} \end{pmatrix}$$

is a parity-check matrix for a binary Goppa code $\Gamma_2(L,g)$. Hence, for G a generator matrix for a public code in the McEliece, Niederreiter or CFS scheme, there always exist vectors x and y so that (4.2) holds.

Faugere et al. [17] translate (4.2) into a system of linear equations. Equation (4.2) is equivalent to

$$\{g_{i,1}y_1x_1^e + \cdots + g_{i,n}y_nx_n^e = 0 \text{ for } 1 \le i \le k \text{ and } 0 \le e \le t-1\}$$

which may be rewritten as

$$\{y_ix_i^e = \sum_{j=k+1}^{n} p_{ij}y_jx_j^e \text{ for } 1 \le i \le k \text{ and } 0 \le e \le t-1\}.$$

Next, Faugere et al. point out that, in particular, for $1 \le i \le k$,

$$y_i = \sum_{j=k+1}^{n} p_{ij}y_j,$$

$$y_ix_i = \sum_{j=k+1}^{n} p_{ij}y_jx_j,$$

$$y_ix_i^2 = \sum_{j=k+1}^{n} p_{ij}y_jx_j^2.$$

Using a trivial identity $y_i(y_i x_i^2) = (y_i x_i)^2$, for $1 \leq i \leq k$, Faugere et al. obtain

$$\left(\sum_{j=k+1}^{n} p_{ij} y_j\right)\left(\sum_{j=k+1}^{n} p_{ij} y_j x_j^2\right) = \left(\sum_{j=k+1}^{n} p_{ij} y_j x_j\right)^2$$

which, after rearranging, becomes

$$\sum_{j=k+1}^{n-1} \sum_{j'>j}^{n} p_{ij} p_{ij'} \left(y_j y_{j'} x_{j'}^2 + y_{j'} y_j x_j^2\right) = 0.$$

Finally, Faugere et al. define new variables $Z_{jj'} := y_j y_{j'} x_{j'}^2 + y_{j'} y_j x_j^2$ for $k+1 \leq j \leq n-1$ and $j < j' \leq n$ and a system $\mathcal{L}_P$ of k linear equations

$$\mathcal{L}_P = \left\{ \begin{array}{c} \sum_{j=k+1}^{n-1} \sum_{j'>j}^{n} p_{1j} p_{1j'} Z_{jj'} = 0 \\ \vdots \\ \sum_{j=k+1}^{n-1} \sum_{j'>j}^{n} p_{kj} p_{kj'} Z_{jj'} = 0 \end{array} \right\}.$$

Let the number of variables $Z_{jj'}$ in $\mathcal{L}_P$ be denoted by N and the dimension of the kernel of $\mathcal{L}_P$ by D. Note that $N = \binom{mt}{2}$ and that, by rank-nullity theorem, we have $D = N - rank(\mathcal{L}_P)$.

The main result of [17] is the observation that, for some code $\mathcal{C}$ parameters, the value of D highly probably differs for G a generator matrix for a Goppa code, general alternant code, and a random code.

The authors run experiments by fixing the code parameter m, gradually increasing t from 3 to 50 and computing the values of D for the above categories of linear codes.

Specifically, if G is a random Goppa (or alternant) code generator matrix, Faugere et al. [17] experimentally find formulae for D which depend on m and r and hold, with high probability, when $N - D < k$. Goppa code and, in general, alternant code produce $D > 0$.

On the other hand, in case of G being a generator matrix for a random code, we have $rank(\mathcal{L}_P) = \min(N, k)$ with high probability [17]. Hence, by the rank-nullity, whenever $k \geq N$, a random code highly probably produces $D = 0$. For $N > k$ but such that $k > N - D$, D is nonzero but smaller than for the Goppa (alternant) code [17].

Hence, Faugere et al. [17] conclude that, for any t such that

$$k > N - D, \tag{4.3}$$

we may distinguish a given Goppa code matrix from a random matrix by computing D. Equation (4.3) holds when t is rather small, or, in other words, when the information rate $\frac{k}{n} = \frac{2^m - mt}{2^m}$ of the code is very close to one [17]. For example, when $m = 15$, we must have $t < 34$ and the rate > 0.9845 in order for the code to be distinguishable.

All the parameters proposed for the CFS signature scheme have a sufficiently small t and, therefore, these codes are distinguishable. On the other hand, the method does not apply to most of the codes used in the McEliece/Niederreiter cryptosystems, since these all have much smaller rate [17].

Implications for the CFS scheme. Even though the codes used in the CFS scheme are distinguishable, there has not been any concrete distinguishing attack mounted against the parallel CFS signature scheme [17, 38] and this scheme with parameters $(n, k, \delta, i) = (2^{17}, 10, 2, 2)$, $(2^{18}, 9, 2, 3)$, $(2^{20}, 8, 2, 3)$ as given in Table 4.2 and other parameters from [19] is considered to be secure [38].

On the other hand, the Goppa code indistinguishability assumption 2.3.7 does not hold for high-rate Goppa codes. Hence, the security proof of the CFS scheme by Dallot [15] that assumes the indistinguishability does not hold. Therefore, if one hopes for a provably secure CFS scheme, the proposed code parameters must change. In particular, Preetha et al. [38] compute that, for $n = 2^{18}$ or 2^{19}, we would need $t = 85$ or 114. These values, however, would make the CFS signing times take about 2^{220} operations, i.e. impractically long.

4.6 A new public-key construction

Preetha et al. [38] recently published a new variant of the CFS signature scheme that is meant to securely use the original small Goppa code parameters and yet be provably secure. The authors come up with an idea to modify the public key of the signature scheme in such a way that the distinguishing method [17] would no longer apply to it. This, in particular, means that the public code in the scheme is no longer permutation equivalent to the private code. For their scheme, Preetha et al. [38] also provide a security proof.

We now describe the signature scheme as it is given in [38]. We refer to this scheme as the "PVR scheme".

Definition 4.6.1 (Preetha, Vasant and Rangan signature scheme)**.**
Public parameters: *Integers m, t such that there is a t-error correcting*

binary irreducible Goppa code of length $n = 2^m$ and dimension $k = n - tm$. Denote the set of all such codes by S. A public cryptographic hash function $h : \mathbb{F}_2^n \times \{0,1\}^* \to \mathbb{F}_2^{n'}$, where $\{0,1\}^*$ is the message space and $n' = n-k+1$.

Setup:

1. *Select an $(n-k) \times n$ parity-check matrix H in systematic form for a random $\Gamma_2(L, g) \in S$. Select a random permutation matrix $P \in \mathbb{F}_2^{n \times n}$.*

2. *Pick random matrices $H' \in \mathbb{F}_2^{(n-k)\times n'}$ and $M' \in \mathbb{F}_2^{n' \times (n-k)}$ until the matrix $M := H'M' \in \mathbb{F}_2^{(n-k)\times(n-k)}$ is invertible.*

3. *Pick $\mathbf{a} \in \mathbb{F}_2^{n'}$ such that $H'\mathbf{a}^T = 0$.*

4. *Pick $\mathbf{b} \in \mathbb{F}_2^n$ such that the $n' \times n$ matrix $\hat{H} := M'HP + \mathbf{a}^T\mathbf{b}$ has full rank.*

Private key: *Goppa polynomial $g(X) \in \mathbb{F}_{2^m}[X]$ and support $L \in \mathbb{F}_{2^m}^n$, the parity-check matrix $H \in \mathbb{F}_2^{(n-k)\times n}$ for $\Gamma_2(L, g)$, matrices $M \in \mathbb{F}_2^{(n-k)\times(n-k)}$, $H' \in \mathbb{F}_2^{(n-k)\times n'}$ and $P \in \mathbb{F}_2^{n\times n}$.*

Public key: *The $n' \times n$ matrix $\hat{H} := M'HP + \mathbf{a}^T\mathbf{b}$.*

Signing:

1. *Given a message $\mathrm{m} \in \{0,1\}^*$, pick a random $r \in \mathbb{F}_2^n$ and compute $w' := h(r, \mathrm{m}) \in \mathbb{F}_2^{n'}$.*

2. *Compute $w = H'w'^T \in \mathbb{F}_2^{n-k}$.*

3. *Apply the Niederreiter's decryption algorithm 3.1.2 to $w \in \mathbb{F}_2^{n-k}$.*

4. *If Step 3 fails, go to Step 1. Otherwise, let $s \in \mathbb{F}_2^n$ be the unique vector of weight $\leq t$ with $MHPs^T = w^T$ that was found in Step 3.*

5. *Check whether $w' = \hat{H}s^T$. If not, go to Step 1. Otherwise, output the signature $\tau = (s, r)$.*

Verification: *Given a message* $\mathrm{m} \in \{0,1\}^*$ *and a signature* $\tau = (s, r)$, *compute* $s_1 = h(r, \mathrm{m})$ *and* $s_2 = \hat{H}s^T$. *If* $s_1 = s_2$ *and* $wt(s) \leq t$, *return* true, *otherwise, return* false.

Notice that instead of the two codes in the CFS scheme, namely, the public code with the parity-check matrix MHP and the private code with the parity-check matrix H, there are four codes playing a role in the PVR signature scheme. These are

- the public code with the parity-check matrix $\hat{H}$,
- the Goppa code with the parity-check matrix MHP,
- the Goppa code with the parity-check matrix H (this is the code $\Gamma_2(L, g)$ from above), and
- a private code with the parity-check matrix H'.

The vector $w \in \mathbb{F}_2^{n-k}$ is made to be a syndrome for both the MHP-code and the H'-code. The MHP-code has a secret decoding algorithm – the Niederreiter's decryption algorithm 3.1.2 – that rests on the signer's knowledge of matrices M, H, P, the Goppa polynomial $g(X)$ and the support L for the H-code.

Any $s \in \mathbb{F}_2^n$ produced in Step 3 thus satisfies

$$H'M'HPs^T = MHPs^T = w^T = H'w'^T. \tag{4.4}$$

A valid signature is, in addition, required to satisfy

$$M'HPs^T + \mathbf{a}^T\mathbf{b}s^T = \hat{H}s^T = w'^T. \tag{4.5}$$

The main idea is to require that s is both a decoding of w in the MHP-code as well as a decoding of w' in the $\hat{H}$-code. Then, by applying the decoding algorithm in the MHP-code on w, the signer finds a decoding of w' in the public and much less structured $\hat{H}$-code.

We now elaborate on the relation between the above vectors w and w' (loosely taken from [38]). First, observe that since $H' \in \mathbb{F}_2^{(n-k)\times n'}$, the H'-code has dimension 1. Hence, there are exactly two vectors in the kernel of H'. Put differently, always exactly two distinct vectors $w'_1, w'_2 \in \mathbb{F}_2^{n'}$ map to the same syndrome $w^T = H'w_1'^T = H'w_2'^T \in \mathbb{F}_2^{n-k}$.

Now, clearly, if a given syndrome $w' \in \mathbb{F}_2^{n'}$ is decodable in the $\hat{H}$-code to a vector s, then the corresponding $w^T = H'w'^T$ is also decodable in the MHP-code into the same syndrome s (simply multiply Equation (4.5) by H'

to obtain Equation (4.4)). Equivalently, whenever the syndrome $w \in \mathbb{F}_2^{n-k}$ is not decodable in the MHP-code, neither of the two corresponding syndromes $w'_1, w'_2 \in \mathbb{F}_2^{n'}$ such that $w^T = H'w'^T_1 = H'w'^T_2$ is decodable in the $\hat{H}$-code. What we need to know, however, is what happens when the syndrome $w \in \mathbb{F}_2^{n-k}$ is decodable in the MHP-code.

Lemma 4.6.2. *If the syndrome $w \in \mathbb{F}_2^{n-k}$ is decodable in the MHP-code and $w'_1, w'_2 \in \mathbb{F}_2^{n'}$ are the only two vectors with $w^T = H'w'^T_1 = H'w'^T_2$, then exactly one of w'_1, w'_2 is decodable in the $\hat{H}$-code. Moreover, the two decodings are equal.*

Proof. First we show that at least one of $w'_1.w'_2$ is decodable. Assume that there is an $s \in \mathbb{F}_2^n$ with $wt(s) \leq t$ such that $MHPs^T = w^T$. Further assume that $\hat{H}s^T \neq w'_1$. Then $\hat{H}s^T$ is a vector in $\mathbb{F}_2^{n'}$ that, when multiplied by H', gives w:

$$H'\hat{H}s^T = MHPs^T = w^T.$$

The only such vectors are w'_1 and w'_2. Hence, we must have $\hat{H}s^T = w'_2$. Note that both w and w'_2 are in their respective codes decoded into the same vector s.

Second, not both w'_1, w'_2 can be decodable. If so, then in the MHP-code w would be decodable into two different syndromes, both of weight $\leq t$ (to see this, use Equation (4.5) and multiply it by H'). This is impossible. □

In what follows, we analyze the parameters and the security of the PVR scheme.

Key sizes. The private PVR key is the same as the private key for the CFS scheme, plus the matrix $H' \in \mathbb{F}_2^{(n-k)\times n'}$. Since H' has roughly the same dimensions as the matrix $M \in \mathbb{F}_2^{(n-k)\times(n-k)}$, the private PVR key requires $O(t^2m^2 + 2^m m) = O((n-k)^2 + n\log_2 n)$ bits, exactly as the CFS private key.

The public PVR key consists of the $n' \times n$ matrix $\hat{H}$, and so $O(2^m mt) = O((n-k)n)$ bits are needed to store the public key, as in the CFS scheme case.

Signature cost. Lemma 4.6.2 shows that, when signing, whenever a syndrome $w \in \mathbb{F}_2^{n-k}$ that is decodable in the MHP-code is found (Step 3 of the signing algorithm), the corresponding $w' \in \mathbb{F}_2^{n'}$ from Step 2 is decodable in the $\hat{H}$-code with probability 1/2. In other words, the probability that the condition "$w' = \hat{H}s^T$" in Step 5 of the PVR signing algorithm fails is 1/2.

Similarly as in the CFS scheme, also in the PVR variant one must try to decode, on average, $t!$ different syndromes before a decodable syndrome in Step 3 is found.

Altogether then the signing time of the PVR signature scheme is twice the signing time of the CFS. Hence, the signature time is $O(t!t^2m^3) = O(t!(n-k)^2\log_2 n)$ operations.

Signature length. In a PVR signature $\tau = (s, r)$, s needs $\log_2 \binom{n}{t}$ bits and r needs n bits. Hence, the signature has $O(\log_2 \left(2^n \binom{n}{t}\right))$ bits.

Verification cost. Verification consists of summing up t columns of the $(n' \times n)$ matrix $\hat{H}$, where $n' = n - k + 1 = mt + 1$ and $n = 2^m$. Hence, the verification takes about $O(mt^2) = O((n-k)t)$ binary operations, same as in the CFS scheme.

	Niederreiter	CFS scheme	PVR scheme
public key	$(n-k)n$	$(n-k)n$	$(n-k)n$
private key	$(n-k)^2 + n\log_2 n$	$(n-k)^2 + n\log_2 n$	$(n-k)^2 + n\log_2 n$
encr./verif.	$(n-k)t$	$(n-k)t$	$(n-k)t$
decr./signing	$n(n-k)\log_2 n$	$t!(n-k)^2\log_2 n$	$t!(n-k)^2\log_2 n$
sign. length		$\log_2 n^t$	$\log_2 2^n \binom{n}{t}$

Table 4.5: Niederreiter, CFS and PVR scheme performance comparison. All data represent the big-O values.

Table 4.5 summarizes the performance dependancies of the PVR scheme on the input parameters and compares them to the estimates for the Niederreiter cryptosystem and the CFS signature. We see that these estimates are, basically, same as for the CFS scheme.

It may also easily be noticed that all the trade-offs in the performance for the CFS scheme described in Section 4.4 are also applicable to the PVR signature scheme.

We sum up, that, in terms of the performance, the CFS and the PVR schemes are almost identical.

Security. The only practical difference between the CFS and the PVR schemes lies in the construction of the public key $\hat{H}$ and the security that it offers. Similarly as in the CFS scheme, also in the PVR scheme the hardness of forging a signature reduces to the hardness of the following two problems:

- the Bounded decoding problem 2.3.5 (or, equivalently, the Bounded syndrome decoding problem 2.3.6), and
- the problem of distinguishing the public matrix $\hat{H}$ from a matrix of a random code.

In the CFS scheme, the second problem is simply the Goppa code indistinguishability problem 2.3.7. However, in the PVR scheme, due to a novel construction of the public code $\hat{H}$, the authors replace it with the following assumption [38].

Assumption 4.6.3 (Weak indistinguishability)**.** *Let S be the set of all possible $n' \times n$ public key matrices in the PVR signature scheme 4.6.1. Let R be the set of all $n' \times n$ random matrices of full rank. Pick a value $b \in \{0, 1\}$ uniformly at random. If $b = 0$, let H be a matrix randomly chosen from S, otherwise, pick H randomly from R. Then, there is no algorithm polynomial in n that can correctly decide whether $H \in S$ with probability non-negligably larger than 1/2.*

Preetha et al. [38] check that the distinguishing method by Faugere et al. [17] does not apply and neither can be extended in any simple way to the public PVR code $\hat{H}$. Hence, the authors argue that the Assumption 4.6.3 is weaker than the original Goppa code indistinguishability assumption 2.3.7. Further, to evaluate the hardness of 4.6.3, Preetha et al. [38] note that if a distinguisher $\mathcal{D}$ capable of solving the problem in 4.6.3 with non-negligable probability exists, then this distinguisher can also solve the following problem.

Problem 4.6.4. *Given a full-rank $n' \times n$ matrix $\hat{H}$. Decide whether there exist matrices $H' \in \mathbb{F}_2^{(n-k)\times n'}$, an invertible matrix $M \in \mathbb{F}_2^{(n-k)\times(n-k)}$, a Goppa code parity-check matrix $H \in \mathbb{F}_2^{(n-k)\times n}$ and a permutation matrix $P \in \mathbb{F}_2^{n\times n}$ such that $H'\hat{H} = MHP$.*

The authors claim that the Problem 4.6.4 is an instance of a, so-called, Equivalent punctured codes problem that has been proven NP-complete (see [38]). Although this specific instance contains a Goppa code parity-check matrix H (and not a random matrix), the authors assume that Problem 4.6.4 is still hard to solve. Finally, they conclude that it is unlikely that the distinguisher $\mathcal{D}$ exists and the Assumption 4.6.3 is reasonable [38].

Preetha et al. [38] then show that, provided that the above two assumptions hold, the PVR scheme is provably secure. More precisely:

Theorem 4.6.5. *If the Bounded syndrome decoding problem 2.3.6 is hard and the Weak indistinguishability assumption 4.6.3 holds, then, in the random oracle model, the PVR signature scheme 4.6.1 is ϵ-EUF-CMA with $\epsilon \to 0$ as $n \to \infty$.*

We now present a proof roughly as it is given in [38]. The proof is similar to Dallot's proof [15] for the mCFS and, equally, is based on the methodology of Shoup (see [15]). The idea is to produce a sequence of "games" such that the first game is the standard EUF-CMA scenario 4.1.2, the last game is an oracle for solving the Bounded syndrome decoding problem 2.3.6 and the differences between the successive games are easily quantifiable. Then, one can relate the probabilities of winning in the EUF-CMA game and of solving the syndrome decoding problem.

Proof. We consider a sequence of games as explained above. In particular, the last game attempts to solve the Bounded syndrome decoding problem 2.3.6 for a random matrix $R \in \mathbb{F}_2^{(n-k)\times n}$ of full rank and a random syndrome $v \in \mathbb{F}_2^{n-k}$.

We have a challenger, who sets up each game and a forger $\mathcal{A}$, a polynomial-time algorithm that attempts to win in the individual games. The challenger is responsible for providing the oracles needed in the games. In particular, an EUF-CMA game gives $\mathcal{A}$ access to a signing oracle Σ producing valid signatures for messages of $\mathcal{A}$'s choice. Further, since we are in the random oracle model, $\mathcal{A}$ also has to be given access to the hash function $h : \mathbb{F}_2^n \times \{0,1\}^* \to \mathbb{F}_2^{n'}$ used in the PVR scheme.

In each game, we maintain three lists: Λ, Λ_h, Λ_Σ. The first list, Λ, is indexed by the possible messages $\mathrm{m} \in \{0,1\}^*$. For a given message m, $\Lambda(\mathrm{m})$ stores an index $r \in \mathbb{F}_2^n$ for which the challenger, when simulating the oracles, is able to produce a t-weight decoding of $h(r,\mathrm{m}) \in \mathbb{F}_2^{n'}$ in the $\hat{H}$-code, i.e. the signature for m. During the game, $\Lambda(\mathrm{m})$ can take different values.

The second list, Λ_h, is indexed by an index-message pair (r,m). For a given pair (r,m), $\Lambda_h(r,\mathrm{m})$ stores a triplet $((w',s),s_1)$ where, $w' \in \mathbb{F}_2^{n'}$ is the syndrome assigned to (r,m) under the hash function h, $s_1 \in \mathbb{F}_2^n$ is a vector, of any weight, such that $\hat{H}s_1^T = w'$. Finally, $s \in \mathbb{F}_2^n$ is a 'marker' which is set according to whether the challenger simulating the oracles knows a t-weight decoding of w' or not. In the former case, s is set equal to s_1, in the latter case s remains empty.

The last list, Λ_Σ is indexed by the possible messages $\mathrm{m} \in \{0,1\}^*$. For each message m, $\Lambda_\Sigma(\mathrm{m})$ stores valid signatures $(s,r) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$ as produced by the signing oracle.

All lists start empty at the beginning of each game. If, for a queried value, there is no corresponding value in a list, we denote the output by $\perp$. Let q_h, q_Σ be the maximum number of $\mathcal{A}$'s queries to the hash oracle and the signing oracle, respectively. The signing oracle is given access to the hash oracle and may query h during the game. Let q_h' be the maximum number of $\mathcal{A}$'s hash queries that has not been queried before (either by $\mathcal{A}$ itself or via the signing oracle). Finally, denote the probability of $\mathcal{A}$ winning Game i by $\Pr(W_i)$.

Game 0. Here the challenger plays with the forger $\mathcal{A}$ the EUF-CMA game as described in 4.1.2. In particular, the challenger runs the key generation algorithm for the PVR signature scheme 4.6.1, gives the public key $\hat{H}$ to $\mathcal{A}$ and keeps everything else secret. $\mathcal{A}$ is also given access to the hash oracle h. When $\mathcal{A}$ asks to obtain a valid signature for some message, the challenger runs the signing algorithm from 4.6.1 and gives the corresponding signature to $\mathcal{A}$.

Game 1. In this game, the original hash oracle h is replaced by the following simulation h' [38].

```
Hash oracle simulator $h'$.
Input: $(r, \mathrm{m}) \in (\mathbb{F}_2^n \times \{0,1\}^*)$.
Output: $w' \in \mathbb{F}_2^{n'}$.
Algorithm:
$((w', s), s_1) \leftarrow \Lambda_{h'}(r, \mathrm{m})$;
if $r \neq \Lambda(\mathrm{m})$ then
    if $w' = \perp$ then
        $s_1 \xleftarrow{R} \mathbb{F}_2^n$;
        $w' \leftarrow \hat{H} s_1^T$;
        $s \leftarrow \perp$;
        $\Lambda_{h'}(r, \mathrm{m}) \leftarrow ((w', s), s_1)$;
    end
    return $w'$;
else
    if $w' = \perp$ then
        $s_1 \xleftarrow{R} \mathbb{F}_2^n$ such that $wt(s_1) = t$;
        $w' \leftarrow \hat{H} s_1^T$;
        $s \leftarrow s_1$;
        $\Lambda_{h'}(r, \mathrm{m}) \leftarrow ((w', s), s_1)$;
    end
    return $w'$;
end
```

There are two main cases in h': either $r \neq \Lambda(\mathrm{m})$ or $r = \Lambda(\mathrm{m})$. In the former, h' behaves as a random oracle and picks the vector $w' \in \mathbb{F}_2^{n'}$ at random (by choosing a random $s_1 \in \mathbb{F}_2^n$ and multiplying it by $\hat{H}$). In the latter case, h' constructs w' from a t-weight vector s_1. Here s is set to equal s_1, indicating that s_1 is a valid signature on (r, m) when $h'(r, \mathrm{m}) = w'$. In both cases, the values of w', s and s_1 are stored in the list $\Lambda_{h'}$, guaranteeing that h' produces the same output for any given input, if queried multiple times.

The rest of the game is played exactly as in Game 0. Since the list Λ remains empty, the case $r = \Lambda(\mathrm{m})$ is never visited and all values produced by h' in this game are random. Hence, for the probabilities of $\mathcal{A}$ winning the Game 0 and Game 1, we have

$$\Pr(W_1) = \Pr(W_0). \tag{4.6}$$

Game 2. In this game, the signing oracle Σ (using the signing algorithm from 4.6.1) is replaced by the following signing simulator Σ' [38].

```
Signing oracle simulator $\Sigma'$.
Input: $\mathrm{m} \in \{0,1\}^*$.
Output: $(s, r) \in (\mathbb{F}_2^n)^2$.
Algorithm:
$r \xleftarrow{R} \mathbb{F}_2^n$;
$\Lambda(\mathrm{m}) \leftarrow r$;
run $h'(\Lambda(\mathrm{m}), \mathrm{m})$;
$((w', s), s_1) \leftarrow \Lambda_h(\Lambda(\mathrm{m}), \mathrm{m})$;
if $s = \perp$ then
    ABORT
else
    $\Lambda(\mathrm{m}) \leftarrow \perp$;
end
return $(s, r)$;
```

When signing a message m, Σ' first randomly selects an index $\Lambda(\mathrm{m}) \in \mathbb{F}_2^n$, with which a signature on m will be created. Then the hash oracle h' is called with $(r, \mathrm{m}) = (\Lambda(\mathrm{m}), \mathrm{m})$. Provided that h' has not been called with the pair $(\Lambda(\mathrm{m}), \mathrm{m})$ before, h' produces a syndrome w' with t-weight decoding $s_1 \in \mathbb{F}_2^n$ and sets $s = s_1$. This ensures that the signature output by Σ' will be valid. Finally, Σ' empties $\Lambda(\mathrm{m})$. This guarantees that different attempts to sign the same message produce different signatures. Also, it ensures that, unless being in the process of signing m, $\Lambda(\mathrm{m}) =\perp$ for all m. Hence, the case $r = \Lambda(\mathrm{m})$ in the simulator h' is only ever visited if Σ' is signing m. In particular, this means that, if $\mathcal{A}$ forges a valid signature and wins the game, the forged signature must have used a syndrome w' produced by the "random" case $r \neq \Lambda(\mathrm{m})$ of h'.

The only case when something goes wrong in this game is if a pair (r, m) is first queried at h' on its own, meaning that h' sets $\Lambda_{h'} = ((w', s), s_1)$ with $s =\perp$. If later Σ' is asked to sign m and it happens that in this process $\Lambda(\mathrm{m})$ is chosen to be r, h' does not produce the signature but, stays consistent with its earlier output where w' is random and $s =\perp$. The simulator Σ' then aborts the game, since it is not able to produce a valid signature. This situation happens with probability $\leq \frac{q_\Sigma}{2^n}$ [38]. Also, this is the only case when $\mathcal{A}$ is able to distinguish Game 2 from Game 1. Therefore,

$$|\Pr(W_1) - \Pr(W_2)| \leq \frac{q_\Sigma}{2^n}. \tag{4.7}$$

Game 3. Our hash and signing simulators h' and Σ' no longer need the keys generated in the PVR scheme 4.6.1. In this game we therefore cancel the PVR key generation phase. Instead, we consider the random matrix $R \in \mathbb{F}_2^{(n-k)\times n}$ for which we want to solve the Bounded syndrome decoding problem 2.3.6. As the "public key", we give $\mathcal{A}$ the matrix $R' = \begin{pmatrix} R \\ z \end{pmatrix}$ where $z \in \mathbb{F}_2^n$ is a randomly chosen vector. $\mathcal{A}$ is able to distinguish Game 3 from Game 2 only when it can distinguish the random matrix R' from a parity-check matrix of the PVR scheme. According to the Weak indistinguishability assumption 4.6.3, this happens with negligable probability. Denote this probability by $\mathrm{negl}_{\mathrm{Dist}}(n, k)$. Then,

$$|\Pr(W_2) - \Pr(W_3)| \leq \mathrm{negl}_{\mathrm{Dist}}(n, k). \tag{4.8}$$

Game 4. In this game, we alter the winning condition. At the beginning of the game, the challenger picks a random $c \in \{1, \ldots, q_h'\}$. $\mathcal{A}$ wins the game if, for the forgery output $(\mathrm{m}', (s', r'))$, in addition to the conditions in the previous game, the pair $(r', \mathrm{m}') \in (\mathbb{F}_2^n \times \{0,1\}^*)$ was queried as the c-th distinct input by $\mathcal{A}$ to h'. This happens with probability $\frac{1}{q'_h}$, independent of any choices of $\mathcal{A}$. Therefore,

$$\Pr(W_4) = \frac{\Pr(W_3)}{q'_h}. \tag{4.9}$$

Game 5. In this game, we alter the output of h' to the c-th distinct query by $\mathcal{A}$. Given the syndrome $v \in \mathbb{F}_2^{n-k}$ for which the syndrome decoding problem needs to be solved, pick a random bit v_{ran}. We make the simulator h' output the syndrome $(v|v_{\mathrm{ran}})^T \in \mathbb{F}_2^{n'}$ to the above c-th query. We have

$$\Pr(W_5) = \Pr(W_4). \tag{4.10}$$

If $\mathcal{A}$'s forgery $(\mathrm{m}', (s', r'))$ at the end of the game is valid, we obtain $s' \in \mathbb{F}_2^n$ with $wt(s') \leq t$ such that

$$\begin{pmatrix} R \\ z \end{pmatrix} s'^T = (v|v_{\mathrm{ran}})^T.$$

Then, in particular, $Rs'^T = v^T$ and s' is the solution of the Bounded syndrome decoding problem 2.3.6 for the matrix $R \in \mathbb{F}_2^{(n-k)\times n}$ and the syndrome $v \in \mathbb{F}_2^{n-k}$. By the assumption, the Bounded syndrome decoding problem is hard and the probability of solving it is negligable. Denote this probability by $\mathrm{negl}_{\mathrm{Dec}}(n,k)$. Hence, we must have

$$\Pr(W_5) \leq \mathrm{negl}_{\mathrm{Dec}}(n,k). \tag{4.11}$$

We now combine the above probabilities as in [38]. By triangle inequality, we have

$$|\Pr(W_0) - \Pr(W_3)| \leq$$
$$\leq |\Pr(W_0) - \Pr(W_1)| + |\Pr(W_1) - \Pr(W_2)| + |\Pr(W_2) - \Pr(W_3)|.$$

Also,

$$|\Pr(W_0) - \Pr(W_3)| = |\Pr(W_0) - q'_h \Pr(W_4)| = |\Pr(W_0) - q'_h \Pr(W_5)|$$

by (4.9) and (4.10) and

$$|\Pr(W_0) - \Pr(W_1)| + |\Pr(W_1) - \Pr(W_2)| + |\Pr(W_2) - \Pr(W_3)| \leq$$

$$\leq \frac{q_\Sigma}{2^n} + \mathrm{negl}_{\mathrm{Dist}}(n,k)$$

by (4.6), (4.7) and (4.8). Hence,

$$|\Pr(W_0) - q'_h \Pr(W_5)| \leq \frac{q_\Sigma}{2^n} + \mathrm{negl}_{\mathrm{Dist}}(n, k).$$

Finally, rearranging and using (4.11), we obtain

$$\Pr(W_0) \leq \frac{q_\Sigma}{2^n} + \mathrm{negl}_{\mathrm{Dist}}(n, k) + q'_h \cdot \mathrm{negl}_{\mathrm{Dec}}(n, k).$$

We may conclude that, under our assumptions on the Bounded syndrome decoding problem 2.3.6 and the Weak indistinguishability 4.6.3, the PVR signature scheme is $\left(\frac{q_\Sigma}{2^n} + \mathrm{negl}(n, k, q'_h)\right)$-existentially unforgeable under the chosen message attack, where $\mathrm{negl}(n, k, q'_h)$ is a negligable value dependent on the parameters n, k and q'_h.

$\square$

Attacks. Although the public code in the PVR signature scheme can no longer be distinguished from the random code and the scheme is provably secure, all the attacks solving the syndrome decoding problem described in Section 4.5 still apply to the PVR scheme unchanged. In particular, the Bleichenbacher attack is the most efficient known attack against the PVR signature scheme [38]. Its parameters and running times for the PVR scheme are the same as for the CFS-counter version and can be found in Section 4.5.

Proposed parameters. Since the best known attack on the PVR scheme is the same as on the CFS-counter version scheme and has the same performance in both cases, parameter sets correspond to the same security level in both schemes. In particular, Preetha et al. [38] take the proposed parameters from Finiasz and Sendrier [20] who list parameters for the CFS counter version as a response to the Bleichenbacher attack. A sample of suitable parameters for the PVR scheme corresponding to an 80-bit security level is given in Table 4.6. As expected, these parameters are larger than the parameters for the same security level in the CFS-Parallel scheme.

security level	n	t
81-bit	2^{15}	12
83-bit	2^{19}	11

Table 4.6: Some recommended parameters for the PVR scheme.

In Table 4.7 we calculate the PVR scheme performance data for the parameter values from Table 4.6. For a convenient comparison, we also include a copy of Table 4.3 stating the corresponding performance data for the CFS-Parallel scheme.

parameters (n, t)	$(2^{15}, 12)$	$(2^{19}, 11)$
public key size in MB	0.7	13
private key size in MB	0.05	1.18
verification cost in binary op.	2160	2299
signature cost in binary op.	2^{48}	2^{45}
signature length in KB	4	64

Table 4.7: Performance data for the PVR scheme with up-to-date parameters offering 80-bit security levels.

parameters (n, t, δ, i)	$(2^{17}, 10, 2, 2)$	$(2^{18}, 9, 2, 3)$	$(2^{20}, 8, 2, 3)$
public key size in MB	2.7	5	20
private key size in MB	0.26	0.56	2.5
verification cost in binary op.	3400	4374	3840
signature cost in binary op.	2^{23}	2^{20}	2^{17}
signature length in bits	196	288	294

Table 4.8: Performance data for the CFS-Parallel scheme with up-to-date parameters offering 80-bit security levels (taken from [19]).

We see that, similarly as the CFS-counter version, due to the Bleichenbacher attack, the PVR signature scheme as given in Definition 4.6.1 is hardly practical. The PVR signing times are infeasible and the public key sizes and signature lengths are also rather large.

Possible improvements. As future work with no particular details stated, Preetha et al. [38] propose that to thwart the Bleichenbacher attack, one can consider a parallel PVR scheme. Similarly as in the CFS-Parallel version, we suggest that the signer would create two linked signatures on a given message m using a pair of hash functions h_1 and h_2. The version of the PVR signature scheme, as it is given in 4.6.1, is, however, not suitable for this purpose. This is because of the presence of the random index r as the argument in the hash function. As in the CFS scheme, therefore, simply generating two signatures with h_1 and h_2 would not produce linked signatures. Alternatively, requiring that the r used is the same with both

h_1 and h_2 would, on the other hand, significantly increase the signing times and make the scheme totally impractical. Similarly as was done in the CFS scheme case, we therefore suggest to first reformulate the PVR scheme into a complete decoding version and, only afterwards, introduce the two hash functions h_1 and h_2 and a PVR-Parallel scheme. Such a scheme could then use the same parameters as the parallel CFS scheme for the same levels of security.

Summary. The main contribution of the PVR scheme is the attempt to randomize the public key construct. Although the PVR public code is not entirely random, Preetha et al. [38] manage to formulate a weaker code indistinguishability assumption that seems to hold for the PVR public key, and at the same time allows a security proof for the PVR scheme. Since the Weak indistinguishability 4.6.3 is a new assumption, more reasearch is desirable to confirm its soundness.

As we have seen, regarding the performance and possible attacks, the PVR and the CFS-counter version schemes are almost identical. Hence, also the recommended parameter values, possible trade-offs in performance as well as remedies for the attacks apply to both of these two schemes. The PVR scheme as given in [38] suffers, similarly as the CFS-counter version, from very large signing times and as such is hardly practical. However, it is possible to construct a PVR-Parallel scheme which could securely use the same parameter sets as the CFS-Parallel scheme. It could thus represent the long awaited practical yet provably secure code-based signature scheme.

The remaining shortcomings of the PVR scheme, inherited from the CFS scheme, are the large signing times and public key sizes that, moreover, scale exponentially with the PVR parameters. Even a small increase in parameter values may thus anytime render the PVR(-Parallel) scheme totally impractical. Preetha et al. [38] suggest that a possible solution to this problem may lie in the use of other code families, instead of the binary Goppa codes. Attempts to use the McEliece/Niederreiter cryptosystems with alternative codes in the past have mostly failed because of the distinguishability of these codes from the random ones (see, e.g. [27]). However, Preetha et al. [38] point out that these families may work well if the structure is hidden by randomizing the public key. Thus, even if the PVR signature scheme as such did not become popular, the work of Preetha et al. [38] may play an important role, if the randomized public-key constructs are studied in future.

4.7 Survey of code-based signature schemes

Before closing up, we give a brief survey of alternative code-based constructions related to signing. As for the McEliece/Niederreiter cryptosystems, also in the case of signature schemes, other linear codes have been tried out in place of the Goppa codes. In order to decrease the size of the public key, the use of chained BCH codes is proposed in [23]. Similarly, the so-called double circulant and quasi-dyadic codes are discussed in [11]. Finally, quasi-cyclic matrices are used in the signature schemes in [12, 21]. These matrices have a compact representation, since they are completely determined by their first row [12].

Apart from the CFS signature that, basically, inverts the Niederreiter system, two significantly different ways of constructing a signing primitive have been proposed. These are building a signature scheme from an identification scheme and the KKS construction.

As described in Section 3, Stern proposed in 1994 the first practical code-based identification scheme based on the syndrome decoding problem [16, 34]. Using the Fiat-Shamir method (see, for example, [31] for description), it is possible to convert Stern's scheme into a signature scheme [11, 12, 21, 13]. In this construction, however, the resulting signature length depends on the number of rounds in the identification scheme [13] and traditionally produces signatures of large size – having about 120 Kbits [11, 31].

The KKS signature scheme was published by Kabatianskii, Krouk and Smeets in 1997 [25]. The scheme works with an arbitrary error-correcting code, and, in particular, also with codes for which no efficient decoding algorithm is known. This is because the KKS scheme does not have to perform decoding in order to sign a message (see [25, 12]). The drawbacks of the KKS scheme are a relatively large size of the public key and the fact that the KKS scheme is only a few-times signature scheme [21], as explained below.

The main idea in the scheme is to use a subset of decodable syndromes for a given code that forms a linear subspace of a relatively large dimension. Such a subspace exists for every linear code [25]. One then builds a public matrix with which one is able to decode any syndrome corresponding to a sum of matrix' columns [21]. A detailed description of the scheme is given in [25].

Every signature produced by the KKS scheme gives away some information about the private parameters [11]. After intercepting a few signatures (about 20, but the number depends on the exact choice of parameters), an attacker is able to reconstruct the private key [25, 12, 32].

The most powerful known attack on the KKS scheme is given in [32]. If the parameters are poorly chosen, this attack is able to recover the private key from public parameters even if no signature is kown. The attack is based on an observation that from a pair of public KKS matrices, it is possible to define a linear code in which many low-weight codewords give valid signatures [32]. Stern's low-weight codeword finding algorithm (see Section 3 or, e.g., [16]) is then applied to recover such a signature. This reveals some partial information about the private key which is, in turn, used to produce another valid message-signature pair. The process is repeated, until the entire private key is revealed [32]. The attack works well on all KKS-type schemes published before 2011 [32]. The authors explain that Stern's algorithm turns out to perform in these instances much better than in a general case and conclude that this is because the rates of the pair of codes used are too similar [32]. The attack may be prevented by choosing different parameters [32].

A "noisy" variation of the KKS scheme has been proposed by Barreto et al. (see [32]'s discussion of [2]). Having an additional hash function and an error vector, this one-time signature scheme has been proven to be EUF-1CMA secure in the random oracle model [32]. Similarly as in the KKS case, also this scheme can use arbitrary linear codes and its security stems from the syndrome decoding problem [2].

One-time signature schemes can usually be transformed into multi-times signature schemes via a Merkle tree construction [21]. From a couple of one-time public keys placed at the leaves of a binary tree, one computes a (small) public key at the root that can be securely used a predefined number of times [21]. For a detailed description of a Merkle tree, see [12].

Using a Merkle tree, the authors of [12] extend the KKS signature scheme into a secure multi-time signature. However, problematic is the signature length which is, unfortunately, too large [12].

Another one-time signature that can be turned into a multi-time signature via the Merkle tree is proposed by Gaborit and Schrek in [21]. The authors present the scheme as a good compromise between the existing signature schemes. While the scheme is based on the KKS construction, it only uses codes with a large automorphism group. The aim is to decrease the resulting public key sizes. The basic idea is the following: from a given syndrome and with the help of a so-called syndrome compatible group, one constructs a set of many other decodable syndromes which can be described in a compact way. In particular, the scheme uses cyclic shifts or the action of the group $PSL_2(p)$. Apart from the smaller public key sizes as compared to the KKS scheme, the authors conclude that another advantage of this scheme is the fact that the set of decodable syndromes is not linear as in the KKS which limits the number of potential attacks on the scheme [21].

Other existing constructions related to code-based signing include the so-called ring signatures and threshold ring signatures.

The concept of ring signatures originates from 2001 and is due to Rivest et al. (see [13]'s discussion of [39]). Ring signatures enable a person to sign a message on behalf of a group of people. The signer uses his private key and the public keys of other members of the group without needing consent from anyone. As in the case of group signatures, also here the identity of the signer remains unknown – apart from the fact that the signer must be a member of the group. Unlike in the group signatures, however, the anonymity of the signer is not revocable and ring signatures do not need a group manager, a setup procedure or any coordination [45]. A review of ring signature schemes is given in [31].

The first code-based ring signature scheme was proposed by Zheng et al. [45]. The scheme extends the CFS signature construction. As such, it is also based on the syndrome decoding problem and inherits the short signature lengths. Unfortunately, it also inherits the large signature times [45] making this ring signature scheme not to be very practical [45].

In 2002, Bresson et al. (see [13]) further extended the notion of ring signatures into threshold ring signature scheme. These work similarly as the ring signatures with the exception that, in a group of N members, at least l of them must collaborate in order to be able to produce a valid signature [11]. One such a threshold ring signature scheme has been proposed by Bresson et al. (see [13]).

As for the code-based signature schemes, three threshold ring signature proposals have been made.

The first code-based threshold ring signature scheme is due to Aguilar et al. [31]. Their construction generalizes the Stern's identification scheme and subsequently transforms it into a threshold signature via the Fiat-Shamir method [31]. The scheme guarantees unconditional anonymity and is proven to be existentially unforgeable under a chosen message attack in the random oracle model, assuming the hardness of the minimum distance problem (see [31]). The scheme suffers from a relatively large public key size and signature length [13]. On the other hand, the signing time is linear in the size of the group N and is independent of the number l of the required collaborating signers. With the overall complexity of $O(N)$, the scheme is the fastest known threshold ring signatures, beating other number-theoretic candidates [31].

The second code-based threshold ring signature scheme is due to Dallot et al. (see [11, 13]). As Cayrel and Meziani [11] sum up, the authors combine the generic construction of Bresson et al. with the CFS scheme (see [11]). The scheme is existentially unforgeable under a chosen message attack in the random oracle model and provides unconditional anonymity [13]. The scheme inherits from the CFS scheme short signatures, but also large signing costs and public key sizes [13].

The last code-based threshold ring signature scheme has been proposed by Cayrel et al. [13]. The authors extend the so called q-SD identification scheme previously published by Cayrel et al. (see [13]) into a threshold ring identification scheme which, in turn, is transformed into a threshold ring signature scheme via the Fiat-Shamir method [13]. The scheme uses random linear codes over $\mathbb{F}_q$ and is proven secure in the random oracle model [13]. As compared to the Aguilar et al. scheme above, the construction of Cayrel et al. offers shorter signaturs, smaller public key sizes and faster signature generation [13].

Yet another possible code-based construction are the blind signatures. In such a scheme, the message is disguised before signing so that the signer does not see the original message. However, the validity of the signature can be checked against the original message. These schemes ensure blindness and untracebility of the signed messages [11] and are used in systems where the privacy of the author of the message is needed, for example, in electronic voting [33].

The first non-number-theoretic and no third-party-requiring blind signature is the code-based scheme proposed by Overbeck [33]. The construction is based on the CFS scheme but, as the author explains, may also be applied to lattice-based schemes [33]. The scheme is provably secure assuming the hardness of the so-called permuted kernels problem (see [11, 33]). The drawbacks of the scheme are its large signature sizes and signing times. The scheme is claimed to be rather impractical [33].

Finally, the last code-based constructions that we mention are the identity-based signatures. These are signatures in which the public key is linked to signer's identity, meaning, that there is no need to authenticate the public keys anymore. The first such scheme was proposed in 2001 by Boneh and Franklin (see [10]'s discussion of [9]). The only identity-based signature that does not use elliptic curves or number theory is the code-based scheme proposed by Cayrel et al. [10]. The authors invert a certain syndrome decoding problem which subsequently enables them to relate a private key to any random (identity-based) public value [10]. These public-private values are then used in conjunction with the Stern's signature scheme [10], creating thus a novel scheme. The security of the construction relies on the syndrome decoding problem [10]. The scheme is secure against passive impersonation attacks (see [10]). Finally, its drawbacks are large public key and signature sizes.

As we can see from the survey, a variety of code-based schemes related to signing have been devised. Cayrel and Meziani [11] conclude that, among the possibilities, the Stern's scheme offers the smallest public key sizes, the CFS scheme the shortest signatures and the KKS scheme "a good balance of public key and signature size at the expense of security" [11]. On top of these, signature schemes offering additional functionality, such as ring signatures, threshold ring signatures, blind signatures and identity-based signatures have also been proposed. Although some of these signature schemes are not yet practical, they represent an important milestone: this research contributes to create, in general, alternatives to number-theoretic systems, and, in particular, shows that code-based cryptography has a much wider potential than has been thought some 15 years ago.

Chapter 5

Conclusion

In this thesis, we have given a review of the field of code-based cryptography in general, and of code-based digital signatures in particular. We showed that code-based cryptography is an active research area that has produced a wide variety of practical cryptographic schemes over the past two decades: code-based public-key cryptosystems, digital signature schemes, an identification scheme and hash functions. We have in detail introduced and discussed the McEliece and the Niederreiter cryptosystems and variations of the CFS signature scheme. Although all versions of the cryptographic schemes mentioned in this thesis use the binary Goppa codes, the reader is reminded that many other code families have, in fact, been tried out with the above schemes (see, e.g. [27]). Binary Goppa codes, however, produce systems that have, unlike most of the other families, withstood a long cryptoanalytic scrutiny and are thus considered to be one of the most secure options (see, e.g. [42]).

One of the biggest drawbacks of code-based cryptography is the large sizes of the public keys used in cryptosystems and signature schemes. Although some implementations on small devices exist (see [37, 41]), as well as methods in which the public keys do not have to be stored directly on the platform [43], more research in this direction is needed. A solution may eventually be found in using one of the new code families.

The main shortcoming regarding code-based signatures has until recently been the lack of a provably secure, yet efficient signature scheme. We discussed the development of the CFS scheme [14], the first practical code-based signature scheme, that has been in the last decade modified into the, so-called, CFS-Parallel [19] and mCFS [15] schemes. At the moment, the CFS-Parallel scheme is widely considered as secure since there are no known attacks that could break the scheme in less than fully exponential time [19].

However, at the same time, there does not exist a security proof for the CFS scheme, since the high-rate Goppa codes employed are distinguishable from random codes [17].

We discussed a recent proposal by Preetha et al. [38] that can, potentially, solve this issue. The authors published yet another variant of the CFS scheme in which they randomize the scheme's public code. This modification then enables a security proof [38] under weaker assumptions than the original proof for mCFS [15] did.

What remains to be considered are the relatively large public key sizes (inherited from the cryptosystems) and long signing times in the signature schemes – both the CFS-Parallel as well as the PVR(-Parallel) variant by Preetha et al. [38]. The danger remains that, if, for whatever reason, the parameters of the signature schemes were forced to increase, the schemes may become impractical. This is due to the fact that the public key size and the signing time are exponential in the schemes' parameters.

To conclude, there certainly is a need for more research regarding the above mentioned issues in code-based cryptography. All in all, however, thanks to the active work of many researchers, this field is prepared for being put into practice better than ever before. Especially with the need for quantum-resistant systems, code-based cryptography may be a leading alternative for the post-quantum world.

References

The numbers following each entry indicate the pages on which it is cited.

[1] D. Augot, M. Finiasz, and N. Sendrier. A family of fast syndrome based cryptographic hash functions. In E. Dawson and S. Vaudenay, editors, *Progress in Cryptology Mycrypt 2005*, volume 3715 of *Lecture Notes in Computer Science*, pages 64–83. Springer Berlin Heidelberg, 2005. 1, 23, 32, 33

[2] P.S.L.M. Barreto, R. Misoczki, and M.A. Simplicio Jr. One-time signature scheme from syndrome decoding over generic error-correcting codes. *Journal of Systems and Software*, 84(2):198–204, 2011. 73

[3] E. Berlekamp. Factoring polynomials over large finite fields. *Mathematics of Computation*, 24(111):713–735, 1970. 17

[4] E. Berlekamp. Goppa codes. *Information Theory, IEEE Transactions on*, 19(5):590–592, 1973. 11

[5] E. Berlekamp, R.J. McEliece, and H. Van Tilborg. On the inherent intractability of certain coding problems (corresp.). *Information Theory, IEEE Transactions on*, 24(3):384–386, 1978. 2, 19

[6] D.J. Bernstein. List decoding for binary Goppa codes. In Y.M. Chee, Z. Guo, S. Ling, F. Shao, Y. Tang, H. Wang, and C. Xing, editors, *Coding and Cryptology*, volume 6639 of *Lecture Notes in Computer Science*, pages 62–80. Springer Berlin Heidelberg, 2011. 11, 12

[7] D.J. Bernstein, T. Lange, and C. Peters. Attacking and defending the McEliece cryptosystem. In J. Buchmann and J. Ding, editors, *Post-Quantum Cryptography*, volume 5299 of *Lecture Notes in Computer Science*, pages 31–46. Springer Berlin Heidelberg, 2008. 28, 30, 45

[8] B. Biswas and N. Sendrier. Hymes-an open source implementation of the McEliece cryptosystem (2008) http://www-rocq. inria. fr/secret/cbcrypto/index. php. 17

[9] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer Berlin Heidelberg, 2001. 75

[10] P.-L. Cayrel, P. Gaborit, and M. Girault. Identity-based identification and signature schemes using correcting codes. In D. Augot, N. Sendrier, and J.-P. Tillich, editors, *International Workshop on Coding and Cryptography, WCC*, pages 69–78, 2007. 75, 76

[11] P.-L. Cayrel and M. Meziani. Post-quantum cryptography: Code-based signatures. In T.-H. Kim and H. Adeli, editors, *Advances in Computer Science and Information Technology*, volume 6059 of *Lecture Notes in Computer Science*, pages 82–99. Springer Berlin Heidelberg, 2010. 1, 23, 35, 50, 53, 72, 74, 75, 76

[12] P.-L. Cayrel, A. Otmani, and D. Vergnaud. On Kabatianskii-Krouk-Smeets signatures. In C. Carlet and B. Sunar, editors, *Arithmetic of Finite Fields*, volume 4547 of *Lecture Notes in Computer Science*, pages 237–251. Springer Berlin Heidelberg, 2007. 72, 73

[13] P.-L. Cayrel, A. S. Yousfi, G. Hoffmann, and P. Vron. An improved threshold ring signature scheme based on error correcting codes. In F. Ozbudak and F. Rodriguez-Henriquez, editors, *Arithmetic of Finite Fields*, volume 7369 of *Lecture Notes in Computer Science*, pages 45–63. Springer Berlin Heidelberg, 2012. 72, 74, 75

[14] N.T. Courtois, M. Finiasz, and N. Sendrier. How to achieve a McEliece-based digital signature scheme. In C. Boyd, editor, *Advances in Cryptology ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 157–174. Springer Berlin Heidelberg, 2001. 3, 21, 23, 35, 37, 39, 40, 41, 42, 43, 44, 46, 48, 49, 50, 77

[15] L. Dallot. Towards a concrete security proof of Courtois, Finiasz and Sendrier signature scheme. In S. Lucks, A.-R. Sadeghi, and C. Wolf, editors, *Research in Cryptology*, volume 4945 of *Lecture Notes in Computer Science*, pages 65–77. Springer Berlin Heidelberg, 2008. 4, 35, 37, 55, 58, 64, 77, 78

[16] D. Engelbert, R. Overbeck, and A. Schmidt. A summary of McEliece-type cryptosystems and their security. *Journal of Mathematical Cryptology*, 1(2):151–199, 2007. 1, 9, 11, 15, 18, 23, 24, 28, 29, 32, 33, 35, 50, 72, 73

[17] J.-C. Faugere, V. Gauthier-Umana, A. Otmani, L. Perret, and J. Tillich. A distinguisher for high rate McEliece cryptosystems. In *Information Theory Workshop (ITW), 2011 IEEE*, pages 282–286, 2011. 4, 11, 21, 36, 55, 56, 57, 58, 63, 78

[18] S.V. Fedorenko and P.V. Trifonov. Finding roots of polynomials over finite fields. *Communications, IEEE Transactions on*, 50(11):1709–1711, 2002. 17

[19] M. Finiasz. Parallel-CFS. In A. Biryukov, G. Gong, and D.R. Stinson, editors, *Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Computer Science*, pages 159–170. Springer Berlin Heidelberg, 2011. 4, 35, 38, 40, 41, 42, 44, 45, 46, 49, 50, 53, 54, 55, 58, 70, 77

[20] M. Finiasz and N. Sendrier. Security bounds for the design of code-based cryptosystems. In M. Matsui, editor, *Advances in Cryptology ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 88–105. Springer Berlin Heidelberg, 2009. 4, 35, 50, 51, 52, 53, 69

[21] P. Gaborit and J. Schrek. Efficient code-based one-time signature from automorphism groups with syndrome compatibility. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pages 1982–1986, 2012. 72, 73, 74

[22] V.D. Goppa. A new class of linear correcting codes. *Problemy Peredachi Informatsii*, 6(3):24–30, 1970. 8

[23] O. Hamdi, S. Harari, and A. Bouallegue. Secure and fast digital signatures using BCH codes. *International Journal of Computer Science and Network Security*, 6(10):220–226, 2006. 72

[24] S. Heyse, A. Moradi, and C. Paar. Practical power analysis attacks on software implementations of McEliece. In N. Sendrier, editor, *Post-Quantum Cryptography*, volume 6061 of *Lecture Notes in Computer Science*, pages 108–125. Springer Berlin Heidelberg, 2010. 18

[25] G. Kabatianskii, E. Krouk, and B. Smeets. A digital signature scheme based on random error-correcting codes. In M. Darnell, editor, *Crytography and Coding*, volume 1355 of *Lecture Notes in Computer Science*, pages 161–167. Springer Berlin Heidelberg, 1997. 72

[26] K. Kobara and H. Imai. Semantically secure McEliece public-key cryptosystems – conversions for McEliece PKC. In K. Kim, editor, *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 19–35. Springer Berlin Heidelberg, 2001. 33

[27] G. Landais and J.-P. Tillich. An efficient attack of a McEliece cryptosystem variant based on convolutional codes. In P. Gaborit, editor, *Post-Quantum Cryptography*, volume 7932 of *Lecture Notes in Computer Science*, pages 102–117. Springer Berlin Heidelberg, 2013. 1, 24, 71, 77

[28] Y.X. Li, R.H. Deng, and X.M. Wang. On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. *Information Theory, IEEE Transactions on*, 40(1):271–273, 1994. 23, 27

[29] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-correcting Codes: Part 2*, volume 16. Elsevier, 1977. 2, 8, 9, 11

[30] R.J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN progress report*, 42(44):114–116, 1978. 1, 3, 23, 32, 35

[31] C.A. Melchor, P. Cayrel, P. Gaborit, and F. Laguillaumie. A new efficient threshold ring signature scheme based on coding theory. *Information Theory, IEEE Transactions on*, 57(7):4833–4842, 2011. 72, 74

[32] A. Otmani and J.-P. Tillich. An efficient attack on all concrete KKS proposals. In B.-Y. Yang, editor, *Post-Quantum Cryptography*, volume 7071 of *Lecture Notes in Computer Science*, pages 98–116. Springer Berlin Heidelberg, 2011. 72, 73

[33] R. Overbeck. A step towards QC blind signatures. Technical report, IACR Cryptology ePrint Archive, Report 2009/102, 2009. 75

[34] R. Overbeck and N. Sendrier. Code-based cryptography. In D.J. Bernstein, J. Buchmann, and E. Dahmen, editors, *Post-Quantum Cryptography*, pages 95–145. Springer Berlin Heidelberg, 2009. 1, 23, 32, 33, 35, 50, 51, 53, 72

[35] N. Patterson. The algebraic decoding of Goppa codes. *Information Theory, IEEE Transactions on*, 21(2):203–207, 1975. 11, 15

[36] C. Peters. Information-set decoding for linear codes over $\mathbb{F}_q$. In N. Sendrier, editor, *Post-Quantum Cryptography*, volume 6061 of *Lecture Notes in Computer Science*, pages 81–94. Springer Berlin Heidelberg, 2010. 23, 32, 33

[37] C. Peters. *Curves, Codes, and Cryptography*. PhD thesis, Technische Universiteit Eindhoven, the Netherlands, 2011. 2, 77

[38] M.K. Preetha, S. Vasant, and C.P. Rangan. On provably secure code-based signature and signcryption scheme. Technical report, Cryptology ePrint Archive, Report 2012/585, 2012. 4, 36, 53, 58, 60, 63, 64, 65, 66, 67, 68, 69, 70, 71, 78

[39] R.L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret: Theory and applications of ring signatures. In O. Goldreich, A.L. Rosenberg, and A.L. Selman, editors, *Theoretical Computer Science*, volume 3895 of *Lecture Notes in Computer Science*, pages 164–186. Springer Berlin Heidelberg, 2006. 74

[40] N. Sendrier. Decoding one out of many. In B.-Y. Yang, editor, *Post-Quantum Cryptography*, volume 7071 of *Lecture Notes in Computer Science*, pages 51–67. Springer Berlin Heidelberg, 2011. 52

[41] F. Strenzke. A smart card implementation of the McEliece PKC. In P. Samarati, M. Tunstall, J. Posegga, K. Markantonakis, and D. Sauveron, editors, *Information Security Theory and Practices. Security and Privacy of Pervasive Systems and Smart Devices*, volume 6033 of *Lecture Notes in Computer Science*, pages 47–59. Springer Berlin Heidelberg, 2010. 2, 77

[42] F. Strenzke. Fast and secure root finding for code-based cryptosystems. In J. Pieprzyk, A.-R. Sadeghi, and M. Manulis, editors, *Cryptology and Network Security*, volume 7712 of *Lecture Notes in Computer Science*, pages 232–246. Springer Berlin Heidelberg, 2012. 17, 31, 77

[43] F. Strenzke. Solutions for the storage problem of McEliece public and private keys on memory-constrained platforms. In D. Gollmann and F.C. Freiling, editors, *Information Security*, volume 7483 of *Lecture Notes in Computer Science*, pages 120–135. Springer Berlin Heidelberg, 2012. 2, 24, 31, 77

[44] A. Vardy. The intractability of computing the minimum distance of a code. *Information Theory, IEEE Transactions on*, 43(6):1757–1766, 1997. 2, 19, 20

[45] D. Zheng, X. Li, and K. Chen. Code-based ring signature scheme. *International Journal of Network Security*, 5(2):154–157, 2007. 74