Department of Mathematics,
University of Auckland

Improving the Efficiency of Code-Based Cryptography

Thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy (PhD)
by

Edoardo Persichetti

under the supervision of Ass. Prof. Steven Galbraith

Auckland, November 23rd, 2012

A mamma, papà
Flami e Lulli:
siete la mia vita!

Abstract

Recent public-key cryptography is largely based on number theory problems, such as factoring or computing of discrete logarithm. These systems constitute an excellent choice in many applications, and their security is well defined and understood. One of the major drawbacks, though, is that they will be vulnerable once quantum computers of an appropriate size are available. There is then a strong need for alternative systems that would resist attackers equipped with quantum technology.

One of the most well-known systems of this kind is the McEliece cryptosystem, introduced in 1978, that is based on algebraic coding theory. There are no known vulnerabilities against quantum computers, and it has a very fast and efficient encryption procedure. However, it has also one big flaw, the size of the public key, that makes it impractical for many applications.

The first part of this thesis is dedicated to finding a way to significantly reduce the size of the public key. Latest publications achieve very good results by using codes with particular structures, obtaining keys as small as 4,096 bits. Unfortunately, almost all of the variants presented until now have been broken or proven to be insecure against the so-called *structural attacks*, i.e. attacks that aim to exploit the hidden structure in order to recover the private key. My work is based on Generalized Srivastava codes and represents a generalization of the Quasi-Dyadic scheme proposed by Misoczki and Barreto, with two advantages: a better flexibility, and improved resistance to all the known attacks. An efficient implementation of the above scheme is also provided, as a result of a joint work with P.-L. Cayrel and G. Hoffmann.

In the next chapters, other important aspects of code-based cryptography are investigated. These include the study of a higher security standard, called indistinguishability under a chosen ciphertext attack, in the standard model, and the design of a code-based key encapsulation mechanism (KEM), which is an essential component of the *hybrid encryption* protocol. The last chapter is about digital signatures, a fundamental protocol in modern cryptography; existing code-based signatures schemes are reviewed and a negative result is obtained, showing that the design of an efficient signature scheme based on coding theory is still an open problem.

Acknowledgements

First of all, I would like to thank my supervisor Steven Galbraith, for his constant support throughout the development of this PhD. Thanks for giving me the chance to come to New Zealand and pursue this doctorate, for the endless patience you have shown in every situation, for sharing your expertise with me and allowing all this to become a reality.
A big thanks goes also to the University of Auckland and the Maths Department for being a wonderful host, together with all of its members. I would like to mention in particular Julia Novak, Alastair McNaughton, Eamonn O'Brien, Arkadii Slinko, Tom ter Elst, Greg Oates and all the administration ladies, especially Adina Nagy and Olita Moala.
To my co-authors Pierre-Louis Cayrel and Gerhard Hoffmann: thanks for inviting and welcoming me to Darmstadt, and for starting a pleasant and fruitful collaboration despite the several miles that separate us.
During these three years I had the luck to meet many people with whom I shared some very enjoyable and productive experiences, Paulo Barreto, Christiane Peters and Nicolas Sendrier above all.
Thanks to the many people that supported me at various points during my PhD: Paolo Pietrogrande, all this wouldn't even have started without your assistance; Toto Donà, my biggest fan and eternal model; Kari Buckland, a small woman with a great heart, you are a wonderful person and I will always remember all you've done for me and my family; Valentina Napoli, the very first person I met in Auckland and a true friend.
A special mention for Arnaud Brothier: you've been a great friend and a great inspiration since the very start, and during all your visits to Auckland and in Paris. Merci, mon ami.
Finally, I would like to thank all the people that accompanied me along this long and tortuous path that was the road to my PhD: my girlfriend Alicia and my boys Roberto, Dario, Stefano, Guido, Giovanni, Peppe and Andrea; my officemates Heiko, Tuan and Nazli and my colleagues Manfred, Paul, Maryam, Katie, Mike, Ali, Afshin, Peter, Jennifer, Steffi and all the rest of the PhD guys; Alfio, Martina, Sandra and all the people at Dante Alighieri Society; Claire, Andrecita, Amy, Brigida, Salil, Brice, Muteb, Jordan, Jonathan and all my good Auckland friends; Katy, John and all the Sale St crew; Capo, Vincenzo and the Gina's boys. Thanks, I will not forget.

To Paolo, Giulia, Miki, Stefano, Ricca, Fabri and all my friends back in Italy:
I did it!!

Contents

Glossary

In this thesis, we adopt the following mathematical conventions (unless otherwise specified). We denote all strings and vectors in boldface, and sets and matrices with capital letters. If a is a number then $|a|$ is its absolute value, while $|S|$ denotes the cardinality of the set S. If $\boldsymbol{x}$ is a string of length n, its elements are denoted by $(x_1, \dots, x_n)$ and its length by $|\boldsymbol{x}|$. If M is a matrix we usually adopt the compact notation $M_{i,j}$ to indicate the element in the i-th row and j-th column, and we denote with $M^\intercal$ its transpose and with M^{-1} its inverse (where applicable). Similarly, $\boldsymbol{x}^\intercal$ indicates the transpose of the vector $\boldsymbol{x}$. We use the symbol $||$ to indicate concatenation of strings and $|$ to indicate concatenation of matrices; that is, if $\boldsymbol{x}$ is a string of length n_1, $\boldsymbol{y}$ is a string of length n_2, A is an $m \times n_1$ matrix and B is an $m \times n_2$ matrix, then $(\boldsymbol{x}||\boldsymbol{y})$ is the string of length n_1+n_2 obtained by concatenating $\boldsymbol{x}$ and $\boldsymbol{y}$, and $(A|B)$ is the $m \times (n_1 + n_2)$ matrix obtained by concatenating each row of A with the corresponding row of B.
We denote with $a \overset{\$}{\leftarrow} A$ the action of choosing the element a at random from the set or distribution A. We denote by $\mathsf{Pr}[E]$ the probability that the event E occurs and with $\mathsf{Pr}[E|F]$ the conditional probability, that is, the probability that E occurs given that F occurs.
Algorithms that are part of cryptographic protocols are denoted as follows: we use superscript notation to indicate the scheme they refer to, and subscript notation to specify the key in use. The input of the algorithm is given within brackets. So for example $\mathsf{Enc}^{\mathsf{PKE}}_{\mathsf{pk}}(\phi)$ means encryption of ϕ in the cryptosystem PKE under the key pk.

The symbols that we will use most frequently are listed below:

Symbol	Description
$\oplus$	bitwise XOR
$\mathbb{F}_q$	finite field with q elements
$\{0,1\}^*$	set of bit strings of arbitrary length
$\mathbb{W}_{q,n,w}$	set of words of length n and Hamming weight w over $\mathbb{F}_q$
$\perp$	failure/reject
wt	Hamming weight
d	Hamming distance

Introduction

Cryptology is, as defined by Rivest in [103], “the study of techniques for secure communication in the presence of third parties (called adversaries)”. It is commonly divided into two distinct areas known as *Cryptography* and *Cryptanalysis*. The former refers to the use and practice of the techniques in order to create secure communication protocols, while the latter is the study of methods for obtaining the encrypted information without access to the key normally required to do so, i.e. it is the study of how to break the cryptographic protocols. Of course, no area would exist without the presence of the other, and the interaction between the two parts is of vital importance.
For many years the subject was considered only in the context of privacy, and the words “cryptography” and “encryption” were synonyms. Moreover, encryption was always intended as an exchange between two parties in possession of the same key (symmetric cryptography).
Modern cryptology evolved in many directions and, also thanks to the development of computers, features now various types of protocols, such as public-key encryption schemes, signature schemes, zero-knowledge identification schemes, multi-party computations and so on. We will see definitions and examples of the main cryptographic protocols in Chapter 2.

This thesis focuses on the area known as Public-Key Cryptography (see Section 2.1.2 for details), and, in particular, investigates code-based cryptography, that is, the branch of cryptography that makes use of primitives based on hard coding theory problems. As we will see in Section 2.2, coding theory was initially studied with the purpose of solving a variety of problems in electronic communication. The first application of coding theory in a cryptographic context is the 1978 seminal work of R. J. McEliece [80]. Since then, the area has attracted the attention of the community as one of the candidates for the so-called “post-quantum cryptography”. This is the name commonly used to indicate the area of cryptographic research that considers a scenario in which adversaries are equipped with quantum technology. Quantum computers of a small size are already a reality and, although hard to estimate, it is plausible that in the near future such a scenario would be concrete. With enough quantum computational power, an adversary could make use of techniques such as Shor’s algorithm [114] to break many current cryptographic protocols relying on number-theoretic primitives such as RSA and Diffie-Hellman. It is therefore important to provide alternative schemes whose security won’t be affected in case this scenario becomes real. The McEliece cryptosystem has no known vulnerabilities against quantum algorithms. However, code-based cryptography has never been truly considered practical for many cryptographic applications, the most important reason being the very large size of the public key. The aim of this thesis is then to improve the efficiency and the credibility of code-based cryptography by studying and addressing some of these issues.

The work is structured as follows: Chapter 2 is divided into two main sections which provide definitions and notions for, respectively, cryptography and coding theory. A final small section is dedicated to the hard problems based on coding theory, i.e. the connecting point between the two areas. Chapter 3 introduces the McEliece cryptosystem and all the previous work related to it: the Niederreiter variant, a cryptanalytic overview and the most recent algebraic variants. Chapter 4 puts together the content of two distinct, previously published papers [98, 22]. The first paper is an individual work by the author and features an original scheme that was designed with the aim of reducing the public key size. The scheme is a variant of the McEliece cryptosystem that arises from a 2009 proposal by Misoczki and Barreto [85]; it consists of a construction based on the family of Generalized Srivastava codes. To the best of our knowledge, it is the first time that this family of codes has been employed in a cryptographic setting; these codes are compatible with the quasi-dyadic framework of Misoczki and Barreto, and the results obtained by employing Generalized Srivastava codes are comparable to the ones obtained by employing Goppa codes, at the same time providing more generality and responding to security requirements dictated by the most recent structural attack by Faugère, Otmani, Perret and Tillich [38]. On the other hand, [22] is a joint work with Pierre-Louis Cayrel and Gerhard Hoffmann that provides an implementation of the scheme for C++ language and for an embedded microcontroller, together with a conversion that achieves IND-CCA2 security (the most desirable level of security for public-key encryption schemes). It is shown that the costs of the conversion affect the overall timings only minimally, and that the global scheme is very fast, thus making another point in favor of code-based cryptography. Chapter 5 is a short chapter dedicated to the construction of a key encapsulation mechanism (KEM) based on coding theory, specifically on the Niederreiter cryptosystem. KEMs are the public-key component of a recent general approach (KEM-DEM) for "hybrid" encryption (Cramer and Shoup, [30]). The scheme achieves IND-CCA2 security with a very tight security reduction and a very simple construction, and promises even faster implementation results. The implementation work is currently in progress and therefore falls beyond the scope of this thesis. Chapter 6 discusses a work on IND-CCA2 security of code-based cryptography in the standard model. This is also another very important aspect to be treated since schemes that are secure in the standard model are much more desirable, in practice, than schemes that require the use of a random oracle (normally simulated by a cryptographic hash function). The work, which was also previously published as a preprint, stems from a recent paper by Dowsley, Müller-Quade and Nascimento [36] that introduces a new scheme called "k-repetition PKE", inspired by a framework by Rosen and Segev [106], with the suggestion of using a randomized version of McEliece in the framework. A flaw in the security proof is noted and corrected, and an accurate security proof is then presented, together with an instantiation of a McEliece construction that is much closer to the original Rosen-Segev approach. Finally, we present a work on coding signatures in Chapter 7. A detailed literature review forms the first part of the chapter, describing the three main prototypes of code-based signature schemes: CFS [29], KKS [62] and Stern's identification scheme [120]. All three have been studied extensively over the years and many variants have been proposed, the most relevant being included in the chapter; however, none of the proposed schemes or variants managed to achieve efficiency due to multiple issues such as a very long signature size, a very large public key size, a very slow signing algorithm, or simply not enough security. Next, we describe a new approach, initiated by Lyubashevsky for the lattice setting [74]. The construction is simple and elegant, although the required choice of parameters makes it essentially impractical. We then argue the impossibility of translating such approach to a coding theory scenario, mostly due to properties that are inherent to the metric used. We conclude that producing an efficient code-based signature scheme is still an open problem.

Background

2.1 Cryptology

In this section we introduce the basic cryptographic schemes that we will need throughout this thesis. Furthermore, in modern cryptology it is common practice to give precise mathematical definitions for security properties and to consider very powerful adversaries. We will also define those precisely along with the corresponding schemes.

2.1.1 Symmetric Cryptography

Symmetric cryptography's distinctive feature is the use of the same key (hence *symmetric*) for encryption and decryption. The key represents therefore a shared secret between two (or more) parties that wish to communicate.
A Symmetric Encryption (SE) scheme is a 6-tuple $(\mathsf{K}, \mathsf{P}, \mathsf{C}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows.

Table 2.1: Symmetric Encryption scheme.

K	The *key space.*
P	The set of messages to be encrypted, or *plaintext space.*
C	The set of the messages transmitted over the channel, or *ciphertext space.*
KeyGen	A probabilistic key generation algorithm that takes as input a security parameter 1^λ and outputs a key $\kappa \in \mathsf{K}$.
Enc	A deterministic encryption algorithm that receives as input a key $\kappa \in \mathsf{K}$ and a plaintext $\phi \in \mathsf{P}$ and returns a ciphertext $\psi \in \mathsf{C}$.
Dec	A deterministic decryption algorithm that receives as input a key $\kappa \in \mathsf{K}$ and a ciphertext $\psi \in \mathsf{C}$ and outputs a plaintext $\phi \in \mathsf{P}$.

Symmetric schemes are commonly called *ciphers*. The first cipher known dates back to the Romans: there is evidence of Julius Caesar using this method to communicate with his generals, hence the scheme is usually referred to as "Caesar cipher". It consists simply of shifting the letters in a message by a certain number of positions. Modern ciphers are divided into two families: *stream ciphers* and *block ciphers*. Schemes in the first family encrypt the bits of a message one at a time, while the block ciphers, as the name suggests, take a certain number of bits and encrypt them as a single unit, padding the plaintext so that it is a multiple of the block size. An example is the very famous AES [31], that uses blocks of size 128, 192 or 256.
Block ciphers operate in different modes, depending whether the encryption algorithm is applied "as is" (EBC), using an "initialization vector" (CBC and CBCC) or a random "starting point" (CTR and CTRC). We will not go into details here, but we refer the reader to [9, Chapter 4] for precise definitions.

We now present a very popular scheme, the one-time pad (Vernam, 1917, U.S. Patent 1,310,719), that we will need for some of our constructions later on.

Table 2.2: The One-Time Pad.

Setup	Fix system parameters $k, n \in \mathbb{N}$ such that $n \leq k$.
K	The set of binary strings $\{0,1\}^k$.
P	The set of binary strings $\{0,1\}^n$.
C	The set of binary strings $\{0,1\}^n$.
KeyGen	Generate at random a key $\boldsymbol{\kappa} \in \{0,1\}^k$.
Enc	On input a key $\boldsymbol{\kappa} \in \mathsf{K}$ and a plaintext $\phi = (x_1, \dots, x_n) \in \mathsf{P}$, compute $y_i = x_i \oplus \kappa_i$ for $i = 1, \dots, n$ and return the ciphertext $\psi = (y_1, \dots y_n) \in \mathsf{C}$.
Dec	On input a key $\boldsymbol{\kappa} \in \mathsf{K}$ and a ciphertext $\psi = (y_1, \dots, y_n) \in \mathsf{C}$, compute $x_i = y_i \oplus \kappa_i$ for $i = 1, \dots, n$ and return the plaintext $\phi = (x_1, \dots x_n) \in \mathsf{P}$.

Usually in practice one chooses $k = n$.

The one-time pad as presented above achieves *perfect secrecy* (in the sense of unconditional security) as long as the keys are used only once, and then discarded (hence the "one-time"). We will define this concept more accurately among other security notions in Section 2.1.3.
Obviously, the fact that the key needs to be at least as large as the message constitutes a severe limitation to the use of the one-time pad.

2.1.2 Public-key Cryptography

Public-key cryptography was first introduced in the 1970's through the work of Diffie and Hellman [33] and represents a major breakthrough in the cryptographic world. The most famous public-key scheme is probably RSA [104], presented in 1978 by Rivest, Shamir and Adleman and still widely used at the present time. The key feature is the idea of a asymmetric key, as opposed to the symmetric schemes described above; that is, each key is composed of a *pair* of keys. One of the keys is public, used for encryption, and is distributed over the channel, while the other one is private and is in possession of the authorized user(s) only, in order to allow decryption.
Due to this particular nature, to realize the protocol is necessary to have a function that is easy to compute, but hard to invert. In cryptography these are called trapdoor one-way functions.

Definition 2.1 A collection of *Efficiently Computable Functions* is a pair of algorithms $\mathcal{F} = (\mathsf{G}, \mathsf{F})$ where G is a generation algorithm that samples the description f of a function and $\mathsf{F}(f, x)$ is an evaluation algorithm that evaluates the function f on a given input x.

Definition 2.2 A *Trapdoor One-Way Function* is an efficiently computable function that, given the image of a uniformly chosen input, is easy to invert with the use of a certain trapdoor td but hard to invert otherwise. In particular, there exists an algorithm F^{-1} such that $\mathsf{F}^{-1}(\mathsf{td}, \mathsf{F}(f, x)) = x$.

Definition 2.3 Given a one-way function f, a *Hard-Core Predicate* of f is a predicate b (i.e., a function whose output is a single bit) which is easy to compute given the input x but is hard to compute given $f(x)$. That is, there is no probabilistic polynomial-time algorithm that computes $b(x)$ from $f(x)$ with non-negligible advantage.

A Public-Key Encryption (PKE) scheme is a 6-tuple $(\mathsf{K}, \mathsf{P}, \mathsf{C}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows.

Table 2.3: Public-Key Encryption scheme.

K	$\mathsf{K}_{\mathsf{publ}}$ the *public key space.* $\mathsf{K}_{\mathsf{priv}}$ the *private key space.*
P	The set of messages to be encrypted, or *plaintext space.*
C	The set of the messages transmitted over the channel, or *ciphertext space.*
KeyGen	A probabilistic key generation algorithm that takes as input a security parameter 1^λ and outputs a public key $\mathsf{pk} \in \mathsf{K}_{\mathsf{publ}}$ and a private key $\mathsf{sk} \in \mathsf{K}_{\mathsf{priv}}$.
Enc	A (possibly probabilistic) encryption algorithm that receives as input a public key $\mathsf{pk} \in \mathsf{K}_{\mathsf{publ}}$ and a plaintext $\phi \in \mathsf{P}$ and returns a ciphertext $\psi \in \mathsf{C}$.
Dec	A deterministic decryption algorithm that receives as input a private key $\mathsf{sk} \in \mathsf{K}_{\mathsf{priv}}$ and a ciphertext $\psi \in \mathsf{C}$ and outputs either a plaintext $\phi \in \mathsf{P}$ or the failure symbol $\perp$.

Most of the trapdoor one-way functions used in cryptography are based on some hard problems coming from number theory, such as the case of prime factorization for RSA or the discrete logarithm for the Diffie-Hellman scheme. Of course, since the encryption key is public, an attacker trying to decrypt the ciphertext could try to encrypt every possible message and eventually come up with the desired plaintext. Clearly, this happens only in theory; however, it doesn't make sense anymore to speak about perfect secrecy. Instead, public-key schemes are designed so to obtain *computational security.* We define this in the next section.

2.1.3 Security of Encryption Schemes

The word "security" in cryptography has multiple meanings and often depends on external factors rather than just on the scheme itself. There are two main types of security:

- *Unconditional Security*
- *Computational Security*

Unconditional Security

The idea of unconditional security is a concept of Information Theory (hence also called *information-theoretic security*) and dates back to Claude Shannon [113]. It

is the strongest possible notion of security for cryptosystems, and implies that the system is unbreakable even if the attacker has unlimited computational power: the adversary simply does not have enough information to break the security. Thus, an unconditionally secure scheme does not rely on any computational assumptions.
A special case is known as perfect secrecy. This was defined by Shannon in [113].

Definition 2.4 Let $\mathcal{E}$ be an encryption scheme. $\mathcal{E}$ achieves *Perfect Secrecy* if a ciphertext produced by Enc provides no information about the plaintext without knowledge of the key. That is, if we fix probability distributions on P and C with random variables, respectively, Φ and Ψ, then[1]

$$\forall \phi \in \mathsf{P}, \forall \psi \in \mathsf{C} \quad \mathsf{Pr}[\Phi = \phi \mid \Psi = \psi] = \mathsf{Pr}[\Phi = \phi]. \tag{2.1}$$

The one-time pad is the only known perfectly secret cryptographic scheme; the perfect secrecy was proved by Shannon in [113].

Computational Security

A cryptosystem is said to be computationally secure if, for any existing adversary, the computational power needed to break the scheme would exceed the available resources. Therefore, computational security is characterized by levels: a cryptosystem is or isn't secure given a fixed amount of computational resources. These are usually defined by a certain, large, number of operations, such as binary operations, field operations etc. For example, a desirable security level for a public-key cryptosystem is 2^{128} or 2^{256} bit operations, depending on the application. In public-key cryptography, computational security is achieved relying on the hardness of some well-known problem. Because the hardness of a problem is difficult to prove, most of the times this is just "assumed" to hold in order to guarantee the desired security for the scheme. We will see some examples in e.g. Chapter 3. Of course, when designing a cryptographic protocol, one always aims to rely on the weakest possible assumption.

Provable Security

The concept of provable security is relatively recent (Goldwasser and Micali, [53]), and very different from the previous ones. Rather than stating universal security properties (as in unconditional security) or just limiting computational resources (as in computational security), provable security also defines different security notions so that the cryptographer knows what to expect and what to aim for. More precisely, a cryptosystem is said to be provably secure if its security requirements can be stated formally in an adversarial model. Clear assumptions are made about what informations the adversary has access to, as well as the available computational resources. Most of all, it is possible to provide a "proof of security" (hence the term provable), usually called a *reduction*, that works by connecting the scheme to a certain problem for which the hardness is assumed

[1]Note that, while Φ is an independent variable, Ψ depends on the implicit random variable K for a certain probability distribution over K.

to hold. The first object to be formally defined, of course, is the universe in which the proof is given. There are two main kinds, called the Standard Model and the Random Oracle Model, that we will define below.

Definition 2.5 The *Standard Model* is the model of computation where the adversary is only limited by the amount of time and computational power available.

This is the "real-life" scenario. Schemes that are proven secure using only complexity assumptions are said to be secure in the standard model. Since security proofs are notoriously difficult to achieve in the standard model, often cryptographic primitives are replaced by idealized versions, called random oracles.

Definition 2.6 A *Random Oracle* is a mathematical abstraction that works as a theoretical black box, that is, an oracle that answers to every query with a truly random output, chosen uniformly from its output domain. For any specific query, the output returned is always the same.

In this sense, the random oracle is like a mapping that associates to each query a fixed, but random output.
Random oracles are very useful to represent functions that need to have a truly random behavior, most commonly cryptographic hash functions.

Definition 2.7 Let $\mathcal{H}$ be a function on A whose range B is a set of strings of fixed length n. Then $\mathcal{H}$ is a *Cryptographic Hash Function* if it satisfies the following properties:

- *Computability*
  For all $x \in A$ it is easy to compute $\mathcal{H}(x)$.
- *Preimage resistance*
  For all $y \in B$ it is infeasible[2] to find $x \in A$ such that $y = \mathcal{H}(x)$.
- *Second-preimage resistance*
  For all $x \in A$ it is infeasible to find $x' \neq x$ such that $\mathcal{H}(x') = \mathcal{H}(x)$.
- *Collision resistance*
  It is infeasible to find $x_1, x_2 \in A$ such that $x_1 \neq x_2$ and $\mathcal{H}(x_1) = \mathcal{H}(x_2)$.

The value $\mathcal{H}(x)$ is called *message digest* or simply *digest*. Clearly, all the properties are required in order to ensure that a malicious adversary is unable to modify the input without changing its digest. Usually the data is encoded in binary, and we have $A = \{0,1\}^*$ (bit-strings of arbitrary length) and $B = \{0,1\}^n$.

Definition 2.8 The *Random Oracle Model* is the model of computation that allows the functions with random behavior to be modelled as random oracles.

Generally, proofs in this environment aim to show that an attacker must require impossible behavior from the oracle, or solve some problem believed hard. Schemes that admit a security proof of this kind are said to be secure in the random oracle model.

[2] i.e. the computation would take longer than some time bound T.

Other cases such as the Generic Group Model or the Public-Key Infrastructure (PKI) Model are not relevant for this thesis and won't be discussed here. We will instead proceed to describe the main types of *attack models*.

We already saw (Definition 2.2) what is a trapdoor one-way function. We now define one-way security for a PKE scheme; the definition is completely analogous for symmetric schemes.

Definition 2.9 A One-Way adversary is a polynomial-time algorithm $\mathcal{A}$ that takes as input a public key $\mathsf{pk} \in \mathsf{K_{publ}}$ and a ciphertext $\psi = \mathsf{Enc_{pk}}(\phi) \in \mathsf{C}$ and outputs $\phi' \in \mathsf{P}$. The adversary succeeds if $\phi' = \phi$. We say that a PKE scheme is *One-Way Secure* if the probability of success of any adversary $\mathcal{A}$ is negligible in the security parameter, i.e.

$$\mathsf{Pr}[\mathsf{pk} \overset{\$}{\leftarrow} \mathsf{K_{publ}}, \phi \overset{\$}{\leftarrow} \mathsf{P} : \mathcal{A}(\mathsf{pk}, \mathsf{Enc_{pk}}(\phi)) = \phi] \in \mathrm{negl}(\lambda). \tag{2.2}$$

In practice, one-way security only requires that recovering the entire plaintext given a ciphertext and the public key is infeasible, but doesn't tell anything about the *indistinguishability* of a certain ciphertext.

Definition 2.10 An adversary $\mathcal{A}$ for the indistinguishability (IND) property is a two-stage polynomial-time algorithm. In the first stage, $\mathcal{A}$ takes as input a public key $\mathsf{pk} \in \mathsf{K_{publ}}$, then outputs two arbitrary plaintexts ϕ_0, ϕ_1. In the second stage, it receives a ciphertext $\psi^* = \mathsf{Enc_{pk}}(\phi_b)$, for $b \in \{0,1\}$, and returns a bit b^*. The adversary succeeds if $b^* = b$. More precisely, we define the *advantage* of $\mathcal{A}$ against PKE as

$$\mathsf{Adv}(\mathcal{A}, \lambda) = \left|\mathsf{Pr}[b^* = b] - \frac{1}{2}\right|. \tag{2.3}$$

We say that a PKE scheme enjoys *Indistinguishability* if the advantage of any adversary $\mathcal{A}$ over all choices of pk, ψ^* and the randomness used by $\mathcal{A}$ is negligible in the security parameter.

Indistinguishability can be achieved in various attack models. We present here two of the most famous.

Definition 2.11 The attack game for IND-CPA (or *passive attack*) proceeds as follows:

1. Query a key generation oracle to obtain a public key pk.
2. Choose $\phi_0, \phi_1 \in \mathsf{P}$ and submit them to an encryption oracle. The oracle will choose a random $b \in \{0,1\}$ and reply with the "challenge" ciphertext $\psi^* = \mathsf{Enc_{pk}}(\phi_b)$.
3. Output $b^* \in \{0,1\}$.

We say that a PKE scheme has *Indistinguishability against Chosen Plaintext Attacks (IND-CPA)* if the advantage $\mathsf{Adv_{CPA}}$ of any IND adversary $\mathcal{A}$ in the CPA attack model is negligible.

The above model was first introduced in [53] and captures the idea of an adversary being unable of extract even partial information about a plaintext given its corresponding ciphertext. An even stronger attack model, called CCA2 (Rackoff and Simon, [100]), allows the adversary to make use of a decryption oracle during the game, with the only exception that it is not allowed to ask for the decryption of the challenge ciphertext.

Definition 2.12 The attack game for IND-CCA2 (or *active attack*) proceeds as follows:

1. Query a key generation oracle to obtain a public key pk.
2. Make a sequence of calls to a decryption oracle, submitting any string ψ of the proper length (not necessarily an element of C). The oracle will respond with $\mathsf{Dec}_{\mathsf{sk}}(\psi)$.
3. Choose $\phi_0, \phi_1 \in \mathsf{P}$ and submit them to an encryption oracle. The oracle will choose a random $b \in \{0, 1\}$ and reply with the "challenge" ciphertext $\psi^* = \mathsf{Enc}_{\mathsf{pk}}(\phi_b)$.
4. Keep performing decryption queries. If the submitted ciphertext is $\psi = \psi^*$, the oracle will return $\perp$.
5. Output $b^* \in \{0, 1\}$.

We say that a PKE scheme has *Indistinguishability against Adaptive Chosen Ciphertext Attacks (IND-CCA2)* if the advantage $\mathsf{Adv}_{\mathsf{CCA2}}$ of any IND adversary $\mathcal{A}$ in the CCA2 attack model is negligible.

The equivalent scenario for symmetric schemes is a model called *find-guess* (Bellare et al., [5]). The definition is similar to IND, except that in this case some extra information is needed before producing the response bit. This replaces the role of the randomness in the adversary since we are now operating with symmetric encryption. The names "find" and "guess" refer to the two stages of the algorithm.

Definition 2.13 An adversary $\mathcal{A}$ for the find-guess (FG) property is a two-stage polynomial-time algorithm. In the first stage (find), $\mathcal{A}$ takes as input a key $\kappa \in \mathsf{K}$, then outputs two arbitrary plaintexts ϕ_0, ϕ_1 along with some extra information ι to be used later. In the second stage (guess), it receives a ciphertext $\psi^* = \mathsf{Enc}_\kappa(\phi_b)$ for $b \in \{0, 1\}$, and returns a bit $b^* = \mathcal{A}(\kappa, \psi^*, \iota)$. The adversary succeeds if $b^* = b$. More precisely, we define the *advantage* of $\mathcal{A}$ against SE as

$$\mathsf{Adv}(\mathcal{A}, \lambda) = \left|\mathsf{Pr}[b^* = b] - \frac{1}{2}\right|. \tag{2.4}$$

We say that a SE enjoys *Find-Guess security* if the probability of success of any adversary $\mathcal{A}$ over all choices of pk, ψ^* and ι is negligible in the security parameter.

A slightly different notion is the one called *non-malleability*, introduced by Dolev, Dwork and Naor [34]. In this case the adversary has again access to a decryption oracle, but instead of recovering partial information about the plaintext, the aim is to produce another encryption of a different plaintext that is

somehow related to the original. Non-malleability and CCA2 have been proven to be equivalent, in certain settings, by the same authors in [35] and by Bellare et al. in [6].
Other "intermediate" notions of security have been proposed, for example by Naor and Yung [87]. This model, commonly called *indifferent chosen ciphertext attack (CCA1)* as opposed to the one presented above, or sometimes *lunch-time* or *midnight* attack, allows the adversary to query the decryption oracle only before receiving the challenge ciphertext. However, this attack model is much less popular than CCA2 and we will therefore omit a detailed definition.

2.1.4 Digital Signatures

Digital signatures arose approximately at the same time of public-key cryptography; initially conjectured in [33], they were successively formalized by Goldwasser, Micali and Rivest in [54]. Digital signatures are a cryptographic protocol with a different aim from encryption schemes: rather than disguising the message itself, these protocols produce a *signature* to be attached to the transmitted document, in order to preserve its authenticity and to avoid forgeries. The signature is *verified* with the help of a dedicated, public verification algorithm.
The tools used to construct digital signatures are very similar to the ones used in public-key encryption, namely an asymmetric key scheme, and often the same cryptographic primitives can be adapted to produce a signature scheme. This is the case of, for example, RSA [104][3]. Arguably, signatures are nowadays at least as important as encryption schemes in the context of modern communications.

Formally, a Digital Signature scheme, or simply Signature scheme (SS) is a 6-tuple $(\mathsf{K}, \mathsf{M}, \Sigma, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ defined as follows:

Table 2.4: Signature scheme.

K	$\mathsf{K}_{\mathsf{sign}}$ the signing key space. $\mathsf{K}_{\mathsf{ver}}$ the verification key space.
M	The set of documents to be signed, or *message space*.
Σ	The set of the signatures to be transmitted along with the messages, or *signature space*.
KeyGen	A probabilistic key generation algorithm that takes as input a security parameter 1^λ and outputs a signing key $\mathsf{sgk} \in \mathsf{K}_{\mathsf{sign}}$ and a verification key $\mathsf{vk} \in \mathsf{K}_{\mathsf{ver}}$.
Sign	A (possibly probabilistic) signing algorithm that receives as input a signing key $\mathsf{sgk} \in \mathsf{K}_{\mathsf{sign}}$ and a message $\mu \in \mathsf{M}$ and returns a signature $\sigma \in \Sigma$.
Ver	A deterministic decryption algorithm that receives as input a verification key $\mathsf{vk} \in \mathsf{K}_{ver}$, a message $\mu \in \mathsf{M}$ and a signature $\sigma \in \Sigma$ and outputs 1, if the signature is recognized as valid, or 0 otherwise.

[3]Although "plain" RSA signatures would not be secure.

Obviously, being a public-key scheme, a signature scheme cannot possibly achieve unconditional security. Instead, just as for PKE schemes, security is tailored on computational assumptions and varies according to different attack models, similar to the ones presented in the previous section, all illustrated in [54]:

- *Key-only attack*
  The attacker is only given the public verification key.
- *Known message attack*
  The attacker is in possession of valid signatures for a set of messages known to him, but not of his choice.
- *Adaptive chosen message attack*
  The attacker can request signatures on arbitrary messages.

Clearly, recovering the signing key would result in a total break of the scheme. Other attack results are categorized as follows:

- *Universal forgery*
  The ability to reproduce valid signatures on any message.
- *Selective forgery*
  The ability to reproduce valid signatures on a set of messages chosen by the adversary and fixed before the attack.
- *Existential forgery*
  The ability to reproduce at least one valid message/signature pair.

Since existential forgery is the weakest possible adversarial model, the strongest notion of security for signature schemes is existential unforgeability under an adaptive chosen message attack.

We now present a slightly different version of unforgeability called *one-time strong unforgeability*, which we will employ later in this thesis.

Definition 2.14 We define an adversary $\mathcal{A}$ as a polynomial-time algorithm that acts as follows:

1. Query a key generation oracle to obtain a verification key vk.
2. Choose a message $\mu \in \mathsf{M}$ and submit it to a signing oracle. The oracle will reply with $\sigma = \mathsf{Sign}_{\mathsf{sgk}}(\mu)$.
3. Output a pair (μ^*, σ^*).

The adversary succeeds if $\mathsf{Ver}_{\mathsf{vk}}(\mu^*, \sigma^*) = 1$ and $(\mu^*, \sigma^*) \neq (\mu, \sigma)$. We say that a signature scheme is *One-Time Strongly Unforgeable* if the probability of success of any adversary $\mathcal{A}$ is negligible in the security parameter, i.e.

$$\mathsf{Pr}[\mathsf{vk} \xleftarrow{\$} \mathsf{K}_{\mathsf{ver}} : \mathsf{Ver}_{\mathsf{vk}}(\mathcal{A}(\mathsf{vk}, \mathsf{Sign}_{\mathsf{sgk}}(\mu))) = 1] \in \text{negl}(\lambda). \tag{2.5}$$

A famous example of one-time signature scheme is the Lamport scheme [65], introduced in 1979. The message bits are signed one at a time, and the scheme requires the use of a one-way function (commonly a cryptographic hash function).

Table 2.5: The Lamport Signature Scheme.

Setup	Fix a one-way function $f : Y \to Z$.
K	$\mathsf{K}_{\mathsf{sign}}$ the Cartesian product Y^{2k}. $\mathsf{K}_{\mathsf{ver}}$ the Cartesian product Z^{2k}.
M	The set of binary strings $\{0,1\}^k$.
Σ	The set of binary strings $\{0,1\}^k$.
KeyGen	Choose at random $2k$ elements $y_{i,j} \in Y$ and compute the corresponding images $z_{i,j} = f(y_{i,j})$ for $i = 1,\dots,k$, $j = 0,1$. Return the signing key $\boldsymbol{y} = \{y_{i,j}\} \in \mathsf{K}_{\mathsf{sign}}$ and the verification key $\boldsymbol{z} = \{z_{i,j}\} \in \mathsf{K}_{\mathsf{ver}}$.
Sign	On input a signing key $\boldsymbol{y} \in \mathsf{K}_{\mathsf{sign}}$ and a message $\boldsymbol{\mu} \in \mathsf{M}$, return the signature $\sigma = (y_{1,\mu_1}, \dots, y_{k,\mu_k}) \in \Sigma$.
Ver	On input a verification key $\boldsymbol{z} \in \mathsf{K}_{\mathsf{ver}}$, a message $\boldsymbol{\mu} \in \mathsf{M}$ and a signature $\sigma \in \Sigma$, output 1 if $f(\sigma_i) = z_{i,\mu_i}$ for $i = 1,\dots,k$, else return 0.

Clearly, in order to forge a signature, an attacker would need to invert the function f, contradicting the one-way assumption. It is also immediate to see that this holds as long as each key is used to sign exactly one message.

2.2 Coding Theory

Coding theory began as an engineering problem in the 1940's, with the work of Golay, Hamming and Shannon. It developed thereafter using more and more complex mathematical tools. Modern coding theory sits comfortably in between those two areas, encompassing very diverse families of codes, such as the *algebraic-geometric (AG)* codes coming from algebraic geometry, the *low-density parity-check (LDPC)* codes based on graph theory, and so on. In this thesis, we treat just codes of the first kind.
Coding theory studies the transmission of data, and consists mainly of two aspects: data compression (*source coding*) and error correction (*channel coding*). We will now focus on the latter, and from now on we will therefore speak of *error-correcting codes.*

2.2.1 Error-Correcting Codes

We start by introducing the notion of linear code.

Definition 2.15 Let $\mathbb{F}_q$ be the finite field with q elements. An $[n,k]$ *Linear Code* $\mathcal{C}$ is a subspace of dimension k of the vector space $\mathbb{F}_q^n$.

Elements of the code are called *codewords*. Each message is represented as a vector of $\mathbb{F}_q^k$ and mapped to a unique codeword. The parameter n is the *code length*, k is the *code dimension* and the difference $n-k$ is the *redundancy* of the code. The ratio $R = k/n$ is known as *code rate* and measures the information rate, i.e. the proportion of useful (non-redundant) data transmitted in each codeword.

Codes are usually studied in the context of the Hamming metric, determined by the distance defined below.

Definition 2.16 Let $\mathcal{C}$ be an $[n,k]$ linear code over $\mathbb{F}_q$. Let $\boldsymbol{x} = (x_1, \dots, x_n)$, $\boldsymbol{y} = (y_1, \dots, y_n) \in \mathcal{C}$ be two codewords. The *Hamming Distance* $\mathsf{d_H}(\boldsymbol{x}, \boldsymbol{y})$ between the codewords is the number of positions in which they differ, that is

$$\mathsf{d_H}(\boldsymbol{x}, \boldsymbol{y}) = |\{i : x_i \neq y_i, 1 \leq i \leq n\}|. \tag{2.6}$$

It is easy to see that $\mathsf{d_H}$ is non-negative, symmetric and sub-additive, hence it is effectively a distance.

Definition 2.17 Let $\mathcal{C}$ be an $[n,k]$ linear code over $\mathbb{F}_q$. Let $\boldsymbol{x} = (x_1, \dots, x_n) \in \mathcal{C}$ be a codeword. The *Hamming Weight* $\mathsf{wt_H}(\boldsymbol{x})$ of the codeword is the number of non-zero positions, that is:

$$\mathsf{wt_H}(\boldsymbol{x}) = |\{i : x_i \neq 0, 1 \leq i \leq n\}|. \tag{2.7}$$

Clearly, the Hamming distance and the Hamming weight define each other in the sense that $\mathsf{wt_H}(\boldsymbol{x}) = \mathsf{d_H}(\boldsymbol{x}, 0)$ and $\mathsf{d_H}(\boldsymbol{x}, \boldsymbol{y}) = \mathsf{wt_H}(\boldsymbol{x} - \boldsymbol{y})$.
Alternative metrics, such as the Lee metric, are often used in other scenarios, for example codes over rings, and will not be discussed here. For simplicity, we will then denote the Hamming distance and weight by, respectively, d and wt.

The following is a very important concept for linear codes.

Definition 2.18 Let $\mathcal{C}$ be an $[n,k]$ linear code over $\mathbb{F}_q$. The *Minimum Distance* d of $\mathcal{C}$ is the minimum of the distances among all the codewords, that is

$$d = \min\{\mathsf{d}(\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}, \boldsymbol{x} \neq \boldsymbol{y}\}. \tag{2.8}$$

The minimum distance of a code is fundamental to determine its *error-correction* capabilities. Imagine a codeword $\boldsymbol{x}$ is transmitted over a noisy channel, and errors occur in a certain number of positions, say w. We represent this as an *error vector* $\boldsymbol{e}$ of weight w having non-zero positions exactly where the errors occur. The received word will then be $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{e}$. We say that a code $\mathcal{C}$ is *able to correct w errors* if, for each codeword, it is possible to detect and correct any configuration of w errors occurred during transmission.
The following theorem holds.

Theorem 2.1 *Let $\mathcal{C}$ be an $[n,k]$ linear code over $\mathbb{F}_q$ having minimum distance d. Then $\mathcal{C}$ is able to correct at most $w = \lfloor \frac{d-1}{2} \rfloor$ errors.*

Proof For every codeword $\boldsymbol{x} \in \mathcal{C}$ define the *sphere* of radius w centered in $\boldsymbol{x}$ as $S_{\boldsymbol{x}} = \{\boldsymbol{z} \in \mathbb{F}_q^n : \mathsf{d}(\boldsymbol{z}, \boldsymbol{x}) \leq w\}$. Now consider two spheres $S_{\boldsymbol{x}}$ and $S_{\boldsymbol{y}}$ for $\boldsymbol{x} \neq \boldsymbol{y}$ and let $\boldsymbol{z} \in S_{\boldsymbol{x}} \cap S_{\boldsymbol{y}}$. Then $\mathsf{d}(\boldsymbol{z}, \boldsymbol{x}) \leq w$ and $\mathsf{d}(\boldsymbol{z}, \boldsymbol{y}) \leq w$, hence $\mathsf{d}(\boldsymbol{z}, \boldsymbol{x}) + \mathsf{d}(\boldsymbol{z}, \boldsymbol{y}) \leq 2w$ and this is a contradiction since, by the triangular inequality, $\mathsf{d}(\boldsymbol{z}, \boldsymbol{x}) + \mathsf{d}(\boldsymbol{z}, \boldsymbol{y}) \geq \mathsf{d}(\boldsymbol{x}, \boldsymbol{y}) \geq d$. This shows that the two spheres are disjoint; hence, if the error vector occurred on a codeword has weight $\leq w$, the corresponding vector $\boldsymbol{z}$ belongs to an uniquely determined sphere and it is then possible to recover the correct codeword. $\triangle$

Linear codes can be efficiently described by matrices.

Definition 2.19 Let $\mathcal{C}$ be an $[n, k]$ linear code over $\mathbb{F}_q$. Let $\mathcal{B} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k\}$ be a basis for the vector subspace determined by $\mathcal{C}$. The $k \times n$ matrix G having the vectors of $\mathcal{B}$ as rows is called *Generator Matrix* for $\mathcal{C}$, that is

$$G = \begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \vdots \\ \boldsymbol{v}_k \end{pmatrix}. \tag{2.9}$$

The matrix G *generates* the code as a linear map: for each message $\boldsymbol{m} \in \mathbb{F}_q^k$ we obtain the corresponding codeword $\boldsymbol{m}G$. Of course, since the choice of basis is not unique, so is the choice of generator matrix. More specifically, given a generator matrix G, then the matrix SG, where S is any invertible matrix, generates the same code. It is possible to choose S in a particular way, so that $G = (I_k | M)$. This is called *systematic form* of the generator matrix.

Note that using a generator matrix in systematic form each message appears in the first k positions of the corresponding codeword (i.e. the first k positions carry the *information symbols*).
We now provide another important way to describe a code. We start by introducing the dual code.

Definition 2.20 Let $\mathcal{C}$ be an $[n, k]$ linear code over $\mathbb{F}_q$. The *Dual Code* of $\mathcal{C}$ is the set $\mathcal{C}^\perp = \{\boldsymbol{x} \in \mathbb{F}_q^n : \boldsymbol{x} \cdot \boldsymbol{y} = 0 \, \forall \boldsymbol{y} \in \mathcal{C}\}$.

Theorem 2.2 *Let $\mathcal{C}$ be an $[n, k]$ linear code over $\mathbb{F}_q$. Then the dual code $\mathcal{C}^\perp$ is an $[n, n-k]$ linear code. Moreover, if $G = (I_k | M)$ is a generator matrix in systematic form for $\mathcal{C}$, then $H = (-M^\intercal | I_{n-k})$ is a generator matrix for $\mathcal{C}^\perp$.*

The matrix H is a very important matrix for the code $\mathcal{C}$ itself.

Definition 2.21 Let $\mathcal{C}$ be an $[n, k]$ linear code over $\mathbb{F}_q$ and let $\mathcal{C}^\perp$ be its dual code. The $(n-k) \times n$ generator matrix H is called *Parity-Check Matrix* for $\mathcal{C}$.

The parity-check matrix describes the code as follows:

$$\forall \boldsymbol{x} \in \mathbb{F}_q^n, \ \boldsymbol{x} \in \mathcal{C} \iff H\boldsymbol{x}^\intercal = 0. \tag{2.10}$$

The name comes from the first, somewhat crude method for error detection, the *parity check*, in which a single redundancy bit is added at the end of a codeword, the bit being a 0 if the codeword has an even number of 1's and a 1 otherwise. In this way, if the received word has an odd number of 1's, it is sure that at least an error has occurred. The vector $\boldsymbol{Hx}^\intercal$ is called *syndrome* of $\boldsymbol{x}$, and gives its name to a very efficient error-correcting method, known as *syndrome decoding.* This works by splitting the code $\mathcal{C}$ in q^{n-k} cosets and then pre-computing a table containing the syndromes of all the corresponding coset leaders (that is, the minimal weight elements for each coset).

Table 2.6: Syndrome Decoding.

Input	An $(n-k)\times n$ parity-check matrix H and the received word $\boldsymbol{z}=\boldsymbol{x}+\boldsymbol{e}\in\mathbb{F}_q^n$.
Output	The codeword $\boldsymbol{x}$.
1.	Calculate the syndrome $\boldsymbol{s}=H\boldsymbol{z}^\intercal$.
2.	Find the coset leader $\boldsymbol{\ell}$ associated to $\boldsymbol{s}$.
3.	If $\boldsymbol{\ell}$ is found, return $\boldsymbol{x}=\boldsymbol{z}-\boldsymbol{\ell}$, else return $\perp$.

This method succeeds as long as $w=\mathsf{wt}(\boldsymbol{e})$ is within the correcting radius of the code, i.e. $w\leq\lfloor\frac{d-1}{2}\rfloor$, where d is the minimum distance of the code. In fact, since $\boldsymbol{x}$ is a codeword, we have $\boldsymbol{Hz}^\intercal=\boldsymbol{Hx}^\intercal+\boldsymbol{He}^\intercal=0+\boldsymbol{He}^\intercal=\boldsymbol{He}^\intercal$ and, because its weight is within the correcting radius, $\boldsymbol{e}$ is a uniquely determined coset leader. It is then easy to find the corresponding syndrome on the table.

We will see more advanced decoding methods in the next sections.

2.2.2 Cyclic Codes

A special subfamily of linear codes is that of cyclic codes.

Definition 2.22 Let $\mathcal{C}$ be an $[n,k]$ linear code over $\mathbb{F}_q$. We call $\mathcal{C}$ *Cyclic* if

$$\forall\boldsymbol{a}=(a_0,a_1\dots,a_{n-1}),\ \boldsymbol{a}\in\mathcal{C}\Longrightarrow\boldsymbol{a}'=(a_{n-1},a_0\dots,a_{n-2})\in\mathcal{C}. \tag{2.11}$$

Clearly, if the property holds, then all the right shifts, for any number of positions, have to belong to $\mathcal{C}$ as well.

An algebraic characterization can be given in terms of polynomial rings. In fact, it is natural to build a bijection between cyclic codes and ideals of the polynomial ring $\mathbb{F}_q[x]/(x^n-1)$. We identify the vector $(a_0,a_1\dots,a_{n-1})$ with the polynomial $a_0+a_1x+\cdots+a_{n-1}x^{n-1}$, and then the right shift operation corresponds to the multiplication by x in the ring.
Each ideal is generated by a certain polynomial $g(x)$ (for simplicity, we assume always g to be monic) such that $g(x)$ divides x^n-1. To each polynomial corresponds a distinct cyclic code, and we therefore call g the *generator polynomial* of the code. Like before, we can produce a generator matrix: this will have a special form.

Definition 2.23 Let $\mathcal{C}$ be an $[n, k]$ cyclic code over $\mathbb{F}_q$. Then $\mathcal{B} = \{g(x), xg(x), \dots, x^{k-1}g(x)\}$ is a basis for $\mathcal{C}$ and we obtain the generator matrix

$$G = \begin{pmatrix} g(x) \\ xg(x) \\ \vdots \\ x^{k-1}g(x) \end{pmatrix}. \tag{2.12}$$

Note that G will be in *circulant* form, where the i-th row corresponds to the cyclic right shift by i positions of the first row.

Generalizations include *constacyclic* codes, where in Equation (2.11) $\boldsymbol{a}'$ changes to $(\gamma a_{n-1}, a_0 \dots, a_{n-2})$ for a certain constant $\gamma \in \mathbb{F}_q$, and in particular the special case of $\gamma = -1$ (*negacyclic* codes). Another generalization, the *quasi-cyclic* codes, we will see in the next chapter.

Among cyclic codes are some important families of codes such as Hamming codes, Quadratic-residue codes and especially BCH codes (Hocquenghem [59], Bose and Ray-Chaudhuri [20]), which we will present briefly.

Definition 2.24 Let q be a prime power, $b, \delta \leq n$ positive integers with $(q, n) = 1$, m the multiplicative order of q modulo n and α a primitive n-th root of unity in $\mathbb{F}_{q^m}$. The *BCH Code* over $\mathbb{F}_q$ of length n and designated distance δ is the cyclic code generated by $g(x) = \operatorname{lcm}\{m_i(x) : b \leq i \leq b + \delta - 2\}$, where $m_i(x)$ is the minimal polynomial of α^i over $\mathbb{F}_q$.

If $b = 1$ then the code is said to be *narrow-sense*, and if the length is exactly $n = q^m - 1$ the code is called *primitive*.

The following is known as *BCH Bound*.

Proposition 2.1 *Let $\mathcal{C}$ be a BCH code with designated distance δ. Then $\mathcal{C}$ has minimum distance at least δ.*

BCH codes enjoy many dedicated decoding algorithms, the most famous being probably the Berlekamp-Massey algorithm [12, 76]. They are appreciated for their ease of use, resulting in many applications such as satellite communications, DVD's, two-dimensional bar codes etc.
A subclass of BCH codes is of particular interest to us.

Definition 2.25 Let q be a prime power and k a positive integer. A *Reed-Solomon (RS) Code* is a BCH code having length $n = q - 1$ and designated distance $\delta = n - k + 1$.

These codes were introduced by Reed and Solomon in [101]. Again, if $b = 1$ we talk about *narrow-sense* Reed-Solomon codes.
Narrow-sense RS codes admit an alternative definition in terms of polynomial evaluation.

Definition 2.26 Let q be a prime power and k a positive integer. Let $\mathbb{P}_k$ be the set of polynomials of degree $\leq k$ over $\mathbb{F}_q$, α a primitive n-th root of unity in $\mathbb{F}_q$ and $n = q-1$. Then the code $\mathcal{C} = \{(f(1), f(\alpha), \dots, f(\alpha^{q-2})) : f \in \mathbb{P}_k\}$ is the narrow-sense $[n, k, n-k+1]$ RS code over $\mathbb{F}_q$.

It is straightforward to see that the two definitions are equivalent. The above can be further generalized (Kasami, Lin and Peterson [63]) to define an even more important family of codes.

Definition 2.27 Let q be a prime power and n, k positive integers such that $1 \leq k \leq n \leq q$. Let m be the multiplicative order of q modulo n, α a primitive n-th root of unity in $\mathbb{F}_{q^m}$ and $\mathbb{P}_{m,k}$ be the set of polynomials of degree $\leq k$ over $\mathbb{F}_{q^m}$. Fix distinct $\boldsymbol{x} = (x_1, \dots, x_n)$ and non-zero $\boldsymbol{y} = (y_1, \dots, y_n)$ in $\mathbb{F}_{q^m}^n$. Then the *Generalized Reed-Solomon (GRS) Code* of order $r = n-k$ is the code $\mathsf{GRS}_r(\boldsymbol{x}, \boldsymbol{y}) = \{(y_1 f(x_1), y_2 f(x_2), \dots, y_n f(x_n)) : f \in \mathbb{P}_{m,k}\}$.

Clearly, the narrow-sense RS code $\mathcal{C}$ defined above is the GRS code $GRS_r(\boldsymbol{x}, \boldsymbol{y})$ having $m = 1$, $n = q-1$, $x_i = \alpha^{i-1}$ and $y_i = 1$ for all $i = 1, \dots, n$.

GRS codes have the important property of being *maximum distance separable (MDS)*, since their minimum distance is exactly $n-k+1$. Moreover, it is possible to prove (for example, MacWilliams and Sloane [75]) that $\mathsf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^\perp = \mathsf{GRS}_{n-r}(\boldsymbol{x}, \boldsymbol{y}')$ for a certain sequence $\boldsymbol{y}' \in \mathbb{F}_{q^m}$. With the canonical choice of basis $(1, x, \dots, x^{k-1})$ we can describe the generator matrix of the dual, that, as we know, is a parity-check matrix for $\mathsf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})$, in the following form:

$$H(\boldsymbol{x}, \boldsymbol{y}') = \begin{pmatrix} y'_1 & \dots & y'_n \\ y'_1 x_1 & \dots & y'_n x_n \\ \vdots & \vdots & \vdots \\ y'_1 x_1^{r-1} & \dots & y'_n x_n^{r-1} \end{pmatrix}. \tag{2.13}$$

It is then possible to describe GRS codes through the above parity-check matrix (for ease of notation, we swap the roles of $\boldsymbol{y}$ and $\boldsymbol{y}'$).

Definition 2.28 Let the integers q, m, n, k, the field element α and the sequences $\boldsymbol{x}, \boldsymbol{y}$ be defined as above. Then $\mathsf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})$ is the code with parity-check matrix $H(\boldsymbol{x}, \boldsymbol{y})$.

2.2.3 Alternant Codes

We now present the family of alternant codes, that are defined as *subfield subcodes* of GRS codes.

Definition 2.29 Let $\mathcal{C}$ be an $[n, k]$ linear code over $\mathbb{F}_{q^m}$. The *Subfield Subcode* $\mathcal{C}|_{\mathbb{F}_q}$ of $\mathcal{C}$ over $\mathbb{F}_q$ is the vector space $\mathcal{C} \cap \mathbb{F}_q^n$.

The easiest way to obtain a subfield subcode is to use the trace construction.

Definition 2.30 Let $H = \{h_{i,j}\}$ be an $r \times n$ matrix over $\mathbb{F}_{q^m}$. Fix an ordered basis $E = \{\boldsymbol{e}_1, \dots, \boldsymbol{e}_m\}$ for $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ and the corresponding projection function $\phi_E : \mathbb{F}_{q^m} \to \mathbb{F}_q^m$ defined by $\phi_E(\alpha) = (a_1, \dots, a_m)^\intercal$ for $\alpha = a_1\boldsymbol{e}_1 + \cdots + a_m\boldsymbol{e}_m$. We define the *Trace Matrix* $\mathcal{T}(H)$ as the $rm \times n$ matrix obtained by replacing each element $h_{i,j}$ with $\phi_E(h_{i,j})$, and the *Co-Trace Matrix* $\mathcal{T}'(H)$ as the $rm \times n$ matrix whose $((l-1)r+i, j)$ element is $\phi_E(h_{i,j})_l$, for $i = 1, \dots r$, $j = 1, \dots n$ and $l = 1, \dots, m$. Note that $\mathcal{T}'(H)$ is equivalent to $\mathcal{T}(H)$ by a left permutation.

It is shown in [75] that the dual of $\mathcal{C}|_{\mathbb{F}_q}$ is the trace of the dual of $\mathcal{C}$. Since a generator matrix for the dual code is in fact a parity-check matrix for $\mathcal{C}$, in practice this means that we can build a parity-check matrix for the subfield subcode directly from $\mathcal{C}$.

Theorem 2.3 *Let $\mathcal{C}$ be an $[n, k, d]$ linear code over $\mathbb{F}_{q^m}$ and H be a parity-check matrix for $\mathcal{C}$. Then the subfield subcode $\mathcal{C}|_{\mathbb{F}_q}$ is an $[n, k', d']$ linear code over $\mathbb{F}_q$, where $k' \geq n - m(n-k)$, $d' \geq d$ and $\hat{H} = \mathcal{T}(H)$ is a parity-check matrix for it.*

Proof It is immediate to prove that $\mathcal{C}|_{\mathbb{F}_q}$ is linear. In fact, $\forall\, \boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}|_{\mathbb{F}_q}$ and $\forall\, a, b \in \mathbb{F}_q$, we have $a\boldsymbol{x} + b\boldsymbol{y} \in \mathcal{C}$ (since $\mathcal{C}$ is linear) and $a\boldsymbol{x} + b\boldsymbol{y} \in \mathbb{F}_q^n$ since all the components are elements of $\mathbb{F}_q$. Therefore, $a\boldsymbol{x} + b\boldsymbol{y} \in \mathcal{C} \cap \mathbb{F}_q^n = \mathcal{C}|_{\mathbb{F}_q}$.
It is also obvious that the code length is still n, and since $\mathcal{C}|_{\mathbb{F}_q}$ is a proper subset of $\mathcal{C}$, clearly d' cannot be less than d. To prove $k' \geq n - m(n-k)$, we build the parity-check matrix and then look at the dimension.
For any vector $\boldsymbol{u} = (u_1, \dots, u_n) \in \mathbb{F}_{q^m}^n$, write the projection of each element $\phi_E(u_j) = (a_{1,j}, \dots, a_{m,j})^\intercal$ for $1 \leq j \leq n$, and call $\boldsymbol{u}^{[i]} = (a_{i,1}, \dots, a_{i,n})$ for $1 \leq i \leq m$. Let $\boldsymbol{v} = (v_1, \dots, v_n) \in \mathbb{F}_q^n$. Then

$$\begin{aligned}
\boldsymbol{u} \cdot \boldsymbol{v} = 0 \iff & \textstyle\sum_{j=1}^n u_j v_j = 0 \\
\iff & \textstyle\sum_{j=1}^n (\sum_{i=1}^m a_{i,j} e_i) v_j = 0 \\
\iff & \textstyle\sum_{i=1}^m (\sum_{j=1}^n a_{i,j} v_j) e_i = 0 \\
\iff & \textstyle\sum_{j=1}^n a_{i,j} v_j = 0 \quad \forall i = 1, \dots, m \\
\iff & \boldsymbol{u}^{[i]} \cdot \boldsymbol{v} = 0 \quad \forall i = 1, \dots, m.
\end{aligned}$$

Now, since H is a parity-check matrix for $\mathcal{C}$, it defines the code as usual by $\boldsymbol{v} \in \mathcal{C} \iff H\boldsymbol{v}^\intercal = 0$. So clearly, if $\boldsymbol{v} \in \mathbb{F}_q^n$, we have $\boldsymbol{v} \in \mathcal{C}|_{\mathbb{F}_q} \iff H\boldsymbol{v}^\intercal = 0$. If $\boldsymbol{h}_j$ is the j-th row of H, this means that $\boldsymbol{v} \in \mathcal{C}|_{\mathbb{F}_q} \iff \boldsymbol{h}_j\boldsymbol{v} = 0$ for all $j = 1, \dots, n-k$. For what we have just seen, this is equivalent to say $\boldsymbol{v} \in \mathcal{C}|_{\mathbb{F}_q} \iff \boldsymbol{h}_j^{[i]}\boldsymbol{v} = 0$ for all $j = 1, \dots, n-k$ and $i = 1, \dots, m$. This defines exactly the trace matrix $\mathcal{T}(H)$. We have m rows for each row of H, but $\hat{H}$ is not necessarily of full rank and must be reduced by a Gaussian elimination. Therefore the dimension is $\geq n - m(n-k)$, as claimed. △

We are now ready to define alternant codes.

Definition 2.31 Let $\mathsf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})$ be a GRS code of order r over $\mathbb{F}_{q^m}$ for a certain prime power q and extension degree $m > 1$. The *Alternant Code* $\mathsf{A}_r(\boldsymbol{x}, \boldsymbol{y})$ is the subfield subcode $\mathsf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})|_{\mathbb{F}_q}$.

Alternant codes admit a modified version of the Berlekamp-Massey algorithm, which we present below. First, though, we need to introduce a few important notions.

Definition 2.32 Let $\mathsf{A}_r(\boldsymbol{x}, \boldsymbol{y})$ be an alternant code over $\mathbb{F}_q$ as defined above and let $\boldsymbol{x}$ be the transmitted codeword. Suppose we receive the vector $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{e}$ having $\mathsf{wt}(\boldsymbol{e}) = w$ within the correction range, with error values $v_1, \dots, v_w$ in positions $p_1, \dots, p_w$. We call:

- *Error Locators* the elements $x_{p_1}, \dots, x_{p_w}$
- *Error Locator Polynomial* the polynomial $\Lambda(z) = \prod_{i=1}^{w}(1 - x_{p_i} z)$
- *Error Evaluator Polynomial* the poly $\Omega(z) = \sum_{j=1}^{w} v_j y_{p_j} \prod_{\substack{1 \le i \le w \\ i \neq j}} (1 - x_{p_i} z)$.

It is evident that the error positions are uniquely determined by the reciprocals of the roots of Λ. Once these are found, the error values are given by

$$v_j = \frac{\Omega(x_{p_j}^{-1})}{y_{p_j} \prod_{\substack{1 \le i \le w \\ i \neq j}} (1 - x_{p_i} x_{p_j}^{-1})}. \tag{2.14}$$

Table 2.7: Alternant decoding.

Input	An $r \times n$ parity-check matrix $H(\boldsymbol{x}, \boldsymbol{y})$ and the received word $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{e} \in \mathbb{F}_q^n$.
Output	The codeword $\boldsymbol{x}$.
1.	Calculate the syndrome $\boldsymbol{s} = H\boldsymbol{z}^\mathsf{T}$ and write down the corresponding polynomial $S(z) = \sum_{i=0}^{r-1} s_i z^i$.
2.	Use the Euclidean algorithm for polynomials to solve the *key equation* $$\Omega(z) \equiv \Lambda(z) S(z) \pmod{z^r} \tag{2.15}$$ and retrieve Λ and Ω.
3.	Use a root-finding algorithm[4] to find the roots of Λ. Find the corresponding error positions $p_1, \dots, p_w$ and then the values $v_1, \dots, v_w$; build the error vector $\boldsymbol{e}$ having the value v_i in position p_i for $i = 1, \dots, w$ and 0 everywhere else. Return $\boldsymbol{x} = \boldsymbol{z} - \boldsymbol{e}$.

[4]Commonly a *Chien search* [27].

Among alternant codes are some very important families of algebraic codes, such as:

- *Chien-Choy generalized BCH codes*
- *Goppa codes*
- *Generalized Srivastava codes*

We will analyze in detail the last two, which are of cryptographic importance.

Goppa codes

Goppa codes were first introduced in 1970s by Victor Goppa [55] and represent a simple case of *algebraic-geometric codes*. Those are evaluation codes, like RS codes in Definition 2.26, but where the objects involved, rather than polynomials, are functions evaluated on rational points of a certain algebraic curve. The original formulation [56] is the following.

Let χ be an algebraic curve over $\mathbb{F}_q$, $P_1, \dots, P_n$ distinct rational points on χ and $\mathcal{D}$ the divisor $P_1 + \cdots + P_n$. Let $\mathcal{G}$ be another divisor such that $\mathrm{supp}(\mathcal{G}) \cap \mathrm{supp}(\mathcal{D}) = \emptyset$ and denote by $L(\mathcal{G})$ the unique[5] finite-dimensional vector space, with respect to the divisor $\mathcal{G}$, such that $L(\mathcal{G})$ is a subspace of the function field of χ. The *Goppa Code* $\Gamma(\mathcal{D}, \mathcal{G})$ is defined by

$$\Gamma(\mathcal{D}, \mathcal{G}) = \{(f(P_1), \dots, f(P_n)) : f \in L(\mathcal{G})\}. \tag{2.16}$$

Sometimes, Goppa codes expressed in this way are referred to as *geometric Goppa codes*.
An equivalent, more common formulation is given by means of a generator polynomial, much like BCH codes, and makes use of the subfield subcode construction.

Definition 2.33 Fix a finite field $\mathbb{F}_q$ and an extension degree $m > 1$. Choose a polynomial $g(x)$ in $\mathbb{F}_{q^m}[x]$ of degree $\ell < n/m$ and a sequence of distinct elements $\alpha_1, \dots, \alpha_n \in \mathbb{F}_{q^m}$ (called *support*) such that $g(\alpha_i) \neq 0$ for all i. The polynomial $g(x)$ is called the *Goppa Polynomial.* Define the $[n, n-\ell]$ linear code $\mathcal{C}$ over $\mathbb{F}_{q^m}$ as the set of words $\boldsymbol{a} = (a_1, \dots, a_n) \in \mathbb{F}_{q^m}^n$ such that

$$\sum_{i=1}^{n} \frac{a_i}{x - \alpha_i} \equiv 0 \pmod{g(x)}. \tag{2.17}$$

The *Goppa Code* $\Gamma = \Gamma(\alpha_1, \dots, \alpha_n, g)$ over $\mathbb{F}_q$ is the corresponding subfield subcode $\mathcal{C}|_{\mathbb{F}_q}$.

[5]By the Riemann-Roch theorem [102, 105].

It is easy to see that a Goppa code defined in this way admits a parity-check matrix of the form

$$H(\boldsymbol{\alpha}, g) = \begin{pmatrix} \dfrac{1}{g(\alpha_1)} & \cdots & \dfrac{1}{g(\alpha_n)} \\ \vdots & \vdots & \vdots \\ \dfrac{\alpha_1^{\ell-1}}{g(\alpha_1)} & \cdots & \dfrac{\alpha_n^{\ell-1}}{g(\alpha_n)} \end{pmatrix} \tag{2.18}$$

from which is possible to see that the Goppa code $\mathit{\Gamma}$ is de facto an alternant code, precisely $\mathsf{A}_\ell(\boldsymbol{x}, \boldsymbol{y})$ with $x_i = \alpha_i$, $y_i = 1/g(\alpha_i)$ for $i = 1, \ldots, n$.
It is then also evident that a Goppa code has dimension $k \geq n - m\ell$. The minimum distance is $\ell + 1$, or $2\ell + 1$ in the special binary case ($q = 2$).

Goppa codes enjoy a particularly efficient decoding algorithm, an adaptation of the Berlekamp-Massey algorithm given by Patterson [95]. We will not present this in detail, and we will instead proceed to the next family of codes that is important for our purposes.

Generalized Srivastava codes

This family of codes was introduced in an unpublished work by J. N. Srivastava in 1967 and successively presented by Helgert in [57]. Before the definition, we briefly return to alternant codes. Recall the special form for the parity-check matrix of the alternant code $\mathsf{A}_r(\boldsymbol{x}, \boldsymbol{y})$:

$$H(\boldsymbol{x}, \boldsymbol{y}) = \begin{pmatrix} y_1 & \cdots & y_n \\ y_1 x_1 & \cdots & y_n x_n \\ \vdots & \vdots & \vdots \\ y_1 x_1^{r-1} & \cdots & y_n x_n^{r-1} \end{pmatrix}. \tag{2.19}$$

Remember that for every $r \times r$ invertible matrix S, the matrix SH is an equivalent parity-check matrix. It is then clear that an alternative form for $H(\boldsymbol{x}, \boldsymbol{y})$ is

$$H = \begin{pmatrix} s_{1,1} & \cdots & s_{1,r} \\ s_{2,1} & \cdots & s_{2,r} \\ \vdots & \vdots & \vdots \\ s_{r,1} & \cdots & s_{r,r} \end{pmatrix} \begin{pmatrix} y_1 & \cdots & y_n \\ y_1 x_1 & \cdots & y_n x_n \\ \vdots & \vdots & \vdots \\ y_1 x_1^{r-1} & \cdots & y_n x_n^{r-1} \end{pmatrix} =$$

$$= \begin{pmatrix} y_1 g_1(x_1) & \dots & y_n g_1(x_n) \\ y_1 g_2(x_1) & \dots & y_n g_2(x_n) \\ \vdots & \vdots & \vdots \\ y_1 g_r(x_1) & \dots & y_n g_r(x_n) \end{pmatrix} \tag{2.20}$$

where $g_i(x) = s_{i,1} + s_{i,2}x + s_{i,3}x^2 + \cdots + s_{i,r}x^{r-1}$ for each $i = 1, \dots, r$.

Definition 2.34 Fix a finite field $\mathbb{F}_{q^m}$ with $m > 1$. Let $\alpha_1, \dots, \alpha_n, w_1, \dots, w_s$ be $n+s$ distinct elements of $\mathbb{F}_{q^m}$, and $z_1, \dots, z_n$ be non-zero elements of $\mathbb{F}_{q^m}$. The *Generalized Srivastava (GS) code* of order $r = st$ and length n is the alternant code $\mathsf{A}_r(\boldsymbol{x}, \boldsymbol{y})$ defined by the parity-check matrix (2.20) having

$$g_{(l-1)t+k}(x) = \frac{\prod_{j=1}^{s} (x - w_j)^t}{(x - w_l)^k} \qquad \text{for } l = 1, \dots, s \text{ and } k = 1, \dots, t$$

$$y_i = \frac{z_i}{\prod_{j=1}^{s} (\alpha_i - w_j)^t} \qquad \text{for } i = 1, \dots, n.$$

This implies

$$y_i g_{(l-1)t+k}(\alpha_i) = \frac{z_i}{(\alpha_i - w_l)^k} \tag{2.21}$$

for $i = 1, \dots, n$, $l = 1, \dots, s$ and $k = 1, \dots, t$.
It is then possible to deduce a standard form for the parity-check matrix of GS codes as

$$H = \begin{pmatrix} H_1 \\ H_2 \\ \vdots \\ H_s \end{pmatrix} \tag{2.22}$$

where each block is

$$H_i = \begin{pmatrix} \dfrac{z_1}{\alpha_1 - w_i} & \dots & \dfrac{z_n}{\alpha_n - w_i} \\ \dfrac{z_1}{(\alpha_1 - w_i)^2} & \dots & \dfrac{z_n}{(\alpha_n - w_i)^2} \\ \vdots & \vdots & \vdots \\ \dfrac{z_1}{(\alpha_1 - w_i)^t} & \dots & \dfrac{z_n}{(\alpha_n - w_i)^t} \end{pmatrix}.$$

The original *Srivastava codes* are the special case $t = 1$ and $z_i = \alpha_i^\nu$ for all $i = 1, \dots, n$ and for a certain power ν.
Since GS codes are alternant codes, the parameters are length $n \leq q^m - s$, dimension $k \geq n - mst$ and minimum distance $d \geq st + 1$.
By analogy with BCH codes, GS codes are called *primitive* if the α_i's are chosen to be all the elements of $\mathbb{F}_{q^m}$ apart from the w_i's. In this case the code length is exactly $n = q^m - s$.

GS codes are a large family of codes that includes other families as a special case. For example, when $m = 1$ these are called *Gabidulin codes.* Moreover, it is easy to prove that every GS code with $t = 1$ is a Goppa code.
We will use this property, together with the fact that GS codes can be decoded with the usual alternant decoding algorithm (Table 2.7), to build a cryptographic scheme in Chapter 4.

2.3 Cryptology and Coding Theory: Hard Problems

In the previous sections we've presented the fundamentals of cryptology and coding theory. The meeting point between the two is the branch commonly known as *code-based cryptography*, and is centered on problems that arise from coding theory, which are hard enough to serve as cryptographic primitives. In this section, we will present the most relevant of those problems and discuss their hardness.

We start with the following, commonly called *general decoding problem (GDP).*

Table 2.8: General Decoding Problem.

Given	An $[n, k]$ linear code $\mathcal{C}$ over $\mathbb{F}_q$ and a vector $\boldsymbol{y} \in \mathbb{F}_q^n$.
Goal	Find $\boldsymbol{x} \in \mathcal{C}$ such that $\mathsf{d}(\boldsymbol{x}, \boldsymbol{y})$ is minimal.

Note that this corresponds to correcting a certain number of errors occurred on the codeword $\boldsymbol{x}$, represented by an error vector $\boldsymbol{e}$, that is $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}$. By Theorem 2.1, a unique solution exists if the weight of $\boldsymbol{e}$ is less than or equal to $w = \lfloor \frac{d-1}{2} \rfloor$, where d is the minimum distance of $\mathcal{C}$.

This problem is well known and was proved to be NP-complete by Berlekamp, McEliece and van Tilborg in [13]. Moreover, GDP is believed to be hard on average, and not just on the worst-case instances (see for example Sendrier [111]).

An alternative and very popular formulation is given in terms of the parity-check matrix, and is known as the *Syndrome Decoding Problem (SDP).* Sometimes, this is also referred to as *computational* syndrome decoding problem. However, note that there is no "gap" between the computational and the decisional versions of SDP: an attacker equipped with a decisional syndrome decoding oracle can in fact solve any instance of SDP with a linear number of queries.

Table 2.9: Syndrome Decoding Problem.

Given	An $[n-k, n]$ parity-check matrix for an $[n, k]$ linear code $\mathcal{C}$ over $\mathbb{F}_q$, a vector $\boldsymbol{s} \in \mathbb{F}_q^{n-k}$ and an integer $w \in \mathbb{N}^+$.
Goal	Find $\boldsymbol{e} \in \mathbb{F}_q^n$ of weight $\leq w$ such that $\boldsymbol{s} = \boldsymbol{H}\boldsymbol{e}^\intercal$.

We now present a very important bound for linear codes:

Definition 2.35 Let $\mathcal{C}$ be an $[n, k]$ linear code over $\mathbb{F}_q$. The *Gilbert-Varshamov (GV) Distance* is the largest integer d_0 such that

$$\sum_{i=0}^{d_0-1} \binom{n}{i} (q-1)^i \leq q^{n-k}. \tag{2.23}$$

It is then clear that, if $w \leq d_0$, we have a unique solution to SDP. Otherwise, multiple solutions exist (see for example Overbeck and Sendrier, [94]). It follows that decoding problems are meaningful only if the weight w is *small.* If the given syndrome is random, then the weight is likely to be close to the GV bound, therefore providing a guarantee for the hardness of the problem. However in practice, as we will see, for cryptographic schemes the weight is much smaller since it has to be within the correction range of the code in use.

McEliece and Previous Work

3.1 Original Proposals

3.1.1 The McEliece Cryptosystem

As the name suggests the scheme is due to Robert J. McEliece and dates back to 1978. The original formulation [80] makes use of binary Goppa codes. According to the author, these are chosen mainly for two reasons: they form a large family, providing a vast number of potential public keys, and there exists an efficient, i.e. polynomial-time, algorithm for decoding these codes (e.g. Patterson's algorithm). The scheme can be easily generalized to codes over $\mathbb{F}_q$.

Table 3.1: The McEliece cryptosystem.

Setup	Fix public system parameters $q, m, n, k, w \in \mathbb{N}$ such that $k \geq n - wm$.
K	$\mathsf{K}_{\mathsf{publ}}$ the set of $k \times n$ matrices over $\mathbb{F}_q$. $\mathsf{K}_{\mathsf{priv}}$ the set of triples formed by a $k \times k$ invertible matrix over $\mathbb{F}_q$, an $n \times n$ permutation matrix over $\mathbb{F}_q$ and a code description[1].
P	The vector space $\mathbb{F}_q^k$.
C	The vector space $\mathbb{F}_q^n$.
KeyGen	Generate at random a polynomial $g \in \mathbb{F}_{q^m}[x]$ and elements $\alpha_1, \dots, \alpha_n \in \mathbb{F}_{q^m}$, then build the Goppa code $\Gamma = \Gamma(\alpha_1, \dots, \alpha_n, g)$ over $\mathbb{F}_q$ and its generator matrix $\hat{G}$. Select at random a $k \times k$ invertible matrix S and an $n \times n$ permutation matrix P. Publish the public key $G = S\hat{G}P \in \mathsf{K}_{\mathsf{publ}}$ and store the private key $(S, P, \Gamma) \in \mathsf{K}_{\mathsf{priv}}$.
Enc	On input a public key $G \in \mathsf{K}_{\mathsf{publ}}$ and a plaintext $\boldsymbol{m} \in \mathsf{P}$, sample a random error vector $\boldsymbol{e}$ of weight w in $\mathbb{F}_q^n$ and return the ciphertext $\psi = \boldsymbol{m}G + \boldsymbol{e} \in \mathsf{C}$.
Dec	On input the private key $(S, P, \Gamma) \in \mathsf{K}_{\mathsf{priv}}$ and a ciphertext $\psi \in \mathsf{C}$, first compute ψP^{-1} then apply the decoding algorithm D_Γ to it. If the decoding succeeds, multiply the output $\hat{\boldsymbol{m}}$ by S^{-1}, and return the resulting plaintext $\phi = \hat{\boldsymbol{m}}S^{-1}$. Otherwise, output $\perp$.

It is easy to see that the decryption process works when the ciphertext is correctly formed. In fact, if $\psi = \boldsymbol{m}G + \boldsymbol{e}$ with $\mathsf{wt}(\boldsymbol{e}) = w$, we have $\psi P^{-1} = \boldsymbol{m}S\hat{G} + \boldsymbol{e}P^{-1}$, and since P is a permutation matrix, the vector $\boldsymbol{e}P^{-1}$ has still weight w. We can consider this as the encoding of $\boldsymbol{m}S$ for the code defined by G. The decoding algorithm will succeed returning $\hat{\boldsymbol{m}} = \boldsymbol{m}S$, from which we easily recover $\boldsymbol{m}$.

There are two computational assumptions underlying the security of the scheme.

Assumption 1 (Indistinguishability) *The matrix G output by KeyGen is computationally indistinguishable from a uniformly chosen matrix of the same size.*

Assumption 2 (Decoding hardness) *Decoding a random linear code with parameters n, k, w is hard.*

[1]For Goppa codes, given by the support $\alpha_1, \dots, \alpha_n$ and the Goppa polynomial g.

Note that Assumption 2 is in fact equivalent to assuming the hardness of GDP. It is immediately clear that the following corollary is true.

Corollary 3.1 *Given that both the above assumptions hold, the McEliece cryptosystem is one-way secure under passive attacks.*

Remark 3.1 In a recent paper [37], Faugère et al. presented a distinguisher for instances of the McEliece cryptosystem that make use of high-rate Goppa codes. While this doesn't itself represent an attack on the scheme, avoiding such choices of Γ would at least preserve the generality of the security argument.

A version of the McEliece cryptosystem that uses the parity-check matrix instead of the generator matrix has been subsequently presented by Niederreiter [88], and has been proved to be completely equivalent in terms of security (Li, Deng and Wang [69]). We present it in the next section.

3.1.2 The Niederreiter Cryptosystem

This cryptosystem was introduced by H. Niederreiter in 1985. The security relies directly upon SDP and hence it is often considered a "dual" version of the original McEliece cryptosystem.

Table 3.2: The Niederreiter cryptosystem.

Setup	Fix public system parameters $q, m, n, k, w \in \mathbb{N}$ such that $k \geq n - wm$.
K	$\mathsf{K}_{\mathsf{publ}}$ the set of $(n-k) \times n$ matrices over $\mathbb{F}_q$. $\mathsf{K}_{\mathsf{priv}}$ the set of triples formed by an $(n-k) \times (n-k)$ invertible matrix over $\mathbb{F}_q$, an $n \times n$ permutation matrix over $\mathbb{F}_q$ and a code description.
P	The set $\mathbb{W}_{q,n,w}$ of words of $\mathbb{F}_q^n$ with Hamming weight w.
C	The vector space $\mathbb{F}_q^{(n-k)}$.
KeyGen	Generate at random a polynomial $g \in \mathbb{F}_{q^m}[x]$ and elements $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_{q^m}$, then build the Goppa code $\Gamma = \Gamma(\alpha_1, \ldots, \alpha_n, g)$ over $\mathbb{F}_q$ and its parity-check matrix $\hat{H}$. Select at random an $(n-k) \times (n-k)$ invertible matrix S and an $n \times n$ permutation matrix P. Publish the public key $H = S\hat{H}P \in \mathsf{K}_{\mathsf{publ}}$ and store the private key $(S, P, \Gamma) \in \mathsf{K}_{\mathsf{priv}}$.
Enc	On input a public key $H \in \mathsf{K}_{\mathsf{publ}}$ and a plaintext $\boldsymbol{e} \in \mathsf{P}$, compute the syndrome of $\boldsymbol{e}$, that is $\boldsymbol{s} = H\boldsymbol{e}^{\mathsf{T}}$ and return the ciphertext $\psi = \boldsymbol{s} \in \mathsf{C}$.
Dec	On input the private key $(S, P, \Gamma) \in \mathsf{K}_{\mathsf{priv}}$ and a ciphertext $\psi \in \mathsf{C}$, first compute $S^{-1}\psi$ then apply the decoding algorithm D_Γ to it. If the decoding succeeds, multiply the output $\hat{\boldsymbol{e}}$ by P^{-1}, and return the resulting plaintext $\phi = P^{-1}\hat{\boldsymbol{e}}^{\mathsf{T}}$. Otherwise, output $\perp$.

Just like before, we can verify the consistency of the decryption process. In fact, we have $S^{-1}\psi = \hat{H}P\boldsymbol{e}^{\mathsf{T}}$ and since P is a permutation matrix, the vector $P\boldsymbol{e}^{\mathsf{T}}$ has still weight w. Decoding and then multiplying by P^{-1} on the left returns the desired plaintext.

The computational assumptions for Niederreiter are almost the same, except for Assumption 1, that changes as follows.

Assumption 3 (Indistinguishability) *The $(n-k)\times n$ matrix H output by KeyGen is computationally indistinguishable from a uniformly chosen matrix of the same size.*

Remark 3.2 Note that the use of matrices S and P, in both schemes, is rather outdated and unpractical; moreover, it can introduce vulnerabilities to the scheme as per the work of Strenzke et al. (for example [122, 123]). A still secure (Biswas and Sendrier, [19]), but much simpler description would be to take the public key G (resp. H) to be just the systematic form of $\hat{G}$ (resp. $\hat{H}$), and the private key to be Γ alone.

3.1.3 Remarks on the McEliece and Niederreiter Cryptosystems

Note that the encryption process for both cryptosystems is very fast. In fact, its complexity is dominated by (McEliece) or exactly equal to (Niederreiter) a matrix-vector multiplication operation. Even intuitively, this is much simpler than, for example, exponentiation such as the case of RSA.
Recent benchmarks[2] suggest that the McEliece encryption process is often even faster than the NTRU cryptosystem [60], which makes of fast encryption its strongest point. Decryption, on the other hand, involves a decoding operation and that increases considerably the complexity time.
The major drawback of the scheme is the large memory requirements, in particular the necessity to store a big public key. This is possibly also the main reason why code-based cryptography has not yet been considered in any practical application. McEliece in the original manuscript sets the parameters as $n = 1024, k = 524, w = 50$. With this setting, the public key size is 524×1024 bits = 67072 bytes.
A first improvement comes already with the Niederreiter scheme, following the suggestion to compute the systematic form of the public key, i.e. $H = (M|I_{n-k})$, and store only the non-trivial part M to save some space. This would require 500×524 bits = 32750 bytes, clearly still too big for most applications.
Several proposals have then been made in the following years, trying to modify McEliece's original framework in order to deal with this issue. Unfortunately, almost all of them turned out to be insecure or inefficient. Niederreiter himself, in the first place, suggests to use generalized Reed-Solomon codes instead of Goppa codes for his scheme [88]. A famous attack due to Sidelnikov and Shestakov [117] was subsequently published in 1992 and proved that the algebraic structure of GRS codes can be easily exploited, de facto excluding the whole class from the possible choices for a coding-theory based scheme. A similar fate occurred to proposals centered on Reed-Muller codes (Sidelnikov [116], cryptanalysed in [84] by Minder and Shokrollahi) and Gabidulin codes (Gabidulin et al. [47, 48], cryptanalysed in [93] by Overbeck).

[2]http://bench.cr.yp.to/results-encrypt.html

3.2 Security Overview

Due to its particular nature, the most successful attacks on the McEliece cryptosystem (and its variants) are classified into two major families: general attacks and key-recovery attacks. It is McEliece himself, in the final section of the original paper, to suggest this classification. Algorithms of the first kind are ciphertext-only attacks, hence trying to recover the plaintext directly from the ciphertext. To do this, the cryptanalyst is faced with the problem of decoding a linear code with an unknown structure. Thus, these algorithms are usually called *decoding attacks.* The second family contains attacks directed on the private key of the cryptosystem. The aim is to reconstruct the private key in order to be able to apply the decryption algorithm. Sometimes (as in Faugère et al., [38]), it is enough to recover an equivalent key, rather than exactly the private key produced by KeyGen. Since all of these algorithms are based on recognizing the structure of the codes in use, they are known as *structural attacks.*

3.2.1 Decoding Attacks and ISD

Attacks in this category evolve from a brute force decoding approach and try to solve the general decoding problem assuming the knowledge of an upper bound for the distance to the next codeword. Despite several speedups and improvements, decoding attacks require exponential time, and therefore still represent only a non-critical threat to McEliece, in the sense that is enough to enlarge the parameter size in order to make them infeasible. As a consequence, decoding attacks are often used as a tool to determine the minimum parameter size required to achieve the desired security level (e.g. $2^{80}, 2^{128}$ or 2^{256} bit operations). The most renowned and highly regarded is undoubtedly the technique known as *Information-Set Decoding (ISD)*, and all the best decoding attacks are derived from it. The technique takes its name from the fundamental notion of information set.

Definition 3.1 Let G be an arbitrary generator matrix for the $[n, k]$ linear code $\mathcal{C}$. Let $I = \{i_1, \dots, i_k\}$ be a subset of $\{1, \dots, n\}$ and denote by G_I the $k \times k$ submatrix of G formed by the columns indexed by I. If G_I is invertible, then $G' = G_I^{-1}G$ and G generate the same code, and for any codeword $\boldsymbol{m}G'$ the I-indexed entries will carry the information symbols. Therefore, the set I is called an *information set.*

The basic information-set decoding works as follows: consider receiving a vector $\boldsymbol{y}$ in $\mathbb{F}_q^n$ which is known to have distance w from a codeword $\boldsymbol{x} = \boldsymbol{m}G$ in $\mathcal{C}$. Let I be an information set and suppose that $\boldsymbol{y}$ and $\boldsymbol{x}$ coincide on the positions indexed by I, i.e., no errors occurred at these positions. It is then possible to recover the error vector (and consequently the plaintext $\boldsymbol{m}$). Let $\boldsymbol{y} = (y_{i_1}, \dots, y_{i_k})$, then $\boldsymbol{x} = \boldsymbol{y}_I G'$ and we obtain the error vector as $\boldsymbol{y} - \boldsymbol{x}$.
The attack in this primordial form was already proposed by McEliece in his original paper. The next step consists then of iterating this procedure until the selected information set is such that there are no error positions in the

corresponding-indexed columns, i.e. we are in the above situation. This was first formalized by Lee and Brickell [66], whose algorithm (in a generalized version) we present in Table 3.3.

Table 3.3: The generalized Lee-Brickell algorithm.

Input	A generator matrix G, a ciphertext $\psi = \boldsymbol{y} \in \mathbb{F}_q^n$ and a parameter $p \in \mathbb{N}$.
Output	An error vector $\boldsymbol{e}$ of weight w.
1.	Choose a random information set I and compute $\boldsymbol{y}_I, G_I$ and G' as above.
2.	Calculate $\boldsymbol{y}' = \boldsymbol{y} - \boldsymbol{y}_I G'$.
3.	For each size-p subset $\{a_1, \dots, a_p\} \subset I$, for each $x_1, \dots, x_p \in \mathbb{F}_q \setminus \{0\}$: Compute the corresponding[3] weighted sum $\hat{\boldsymbol{g}} = \sum_{i=1}^p x_i G'_{a_i}$. Write $\boldsymbol{e} = \boldsymbol{y}' - \hat{\boldsymbol{g}}$. If $\mathsf{wt}(\boldsymbol{e}) = w$ then return $\boldsymbol{e}$.
4.	Go back to Step 1.

The idea is to allow for p errors in the information set, and iterate the procedure by checking every time the weight of the corresponding error vector obtained. The parameter p is usually chosen small to keep the number of possible size-p subsets reasonably small. In the (original) binary case, $p = 2$ is optimal [18].
In an independent work [68], Leon proposed an improvement while looking for minimum weight-words in a code. This improvement can be adapted and applied to Lee-Brickell's algorithm, and consists in further constraining the possible locations for the errors by introducing a size-ℓ window of zeroes outside of the information set. The idea was optimized by Stern [119], resulting in the following algorithm.

Table 3.4: The generalized Stern algorithm.

Input	A generator matrix G, a ciphertext $\psi = \boldsymbol{y} \in \mathbb{F}_q^n$ and parameters $\ell, p \in \mathbb{N}$.
Output	An error vector $\boldsymbol{e}$ of weight w.
1.	Choose a random information set I and compute $\boldsymbol{y}_I, G_I$ and G' as above.
2.	Calculate $\boldsymbol{y}' = \boldsymbol{y} - \boldsymbol{y}_I G'$.
3.	Choose at random a subset $X \subset I$ of size $k/2$ and set $Y = I \setminus X$, then select at random a size-ℓ set Z in $\{1, \dots, n\} \setminus I$.
4.	For any size-p subset $A = \{a_1, \dots, a_p\} \subset X$ form the set U as the union of the sets $\{\boldsymbol{y}' - \hat{\boldsymbol{g}} : \hat{\boldsymbol{g}} = \sum_{i=1}^p x_i G'_{a_i}\}$ over all choices of $x_1, \dots, x_p \in \mathbb{F}_q \setminus \{0\}$.
5.	For any size-p subset $B = \{b_1, \dots, b_p\} \subset Y$ form the set V as the union of the sets $\{\hat{\boldsymbol{h}} : \hat{\boldsymbol{h}} = \sum_{i=1}^p y_i G'_{b_i}\}$ over all choices of $y_1, \dots, y_p \in \mathbb{F}_q \setminus \{0\}$.
6.	For each pair (A, B): Look for collisions, i.e. vectors $\boldsymbol{u} \in U$ and $\boldsymbol{v} \in V$ such that $\boldsymbol{u}_Z = \boldsymbol{v}_Z$, then write $\boldsymbol{e} = \boldsymbol{u} - \boldsymbol{v}$. If $\mathsf{wt}(\boldsymbol{e}) = w$ then return $\boldsymbol{e}$.
7.	Go back to Step 1.

[3]We indicate with G'_j the row of G' where there is a 1 in position j. Note that, by definition, this is unique if j is an element of an information set.

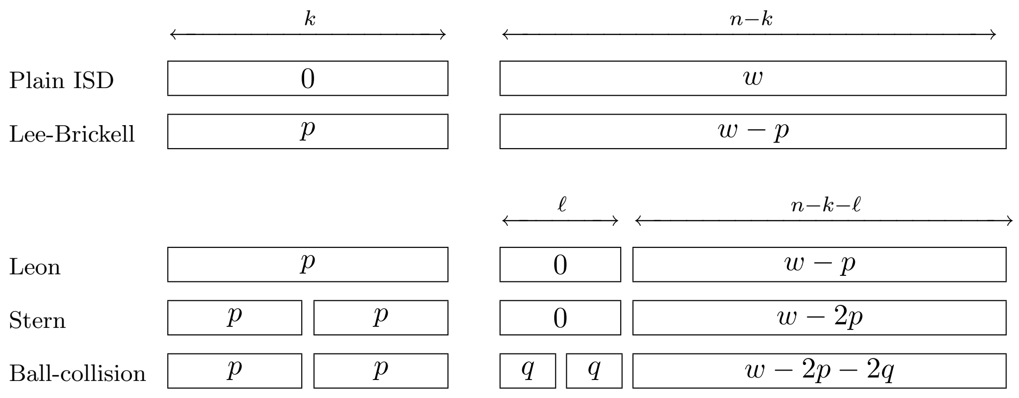
Several other improvements have been proposed and added to Stern's algorithm over the years; we cite in particular [21] and [16]. All of these improvements do not change the general structure of the algorithm, but rather add some technical twists to the process, such as introducing a family of disjoints sets $\{Z_i\}$ instead of the set Z, or reusing existing pivots or additions of vectors in order to obtain a speed-up.
Peters in [99] gives a translation of all the algorithms to the case of codes over $\mathbb{F}_q$ where $q > 2$.
The latest evolution of ISD was presented in 2011 by Bernstein, Lange and Peters with the name "Ball-collision Decoding". A simplified version is presented in Table 3.5; for the complete description of the algorithm, we refer the reader to [17].

Table 3.5: The ball-collision decoding algorithm.

Input	A generator matrix G, a ciphertext $\psi = \boldsymbol{y} \in \mathbb{F}_q^n$ and parameters $\ell_1, \ell_2, p_1, p_2, q_1, q_2, k_1, k_2 \in \mathbb{N}$ such that $p_1 + p_2 + q_1 + q_2 \leq w$ and $k_1 + k_2 = k$.
Output	An error vector $\boldsymbol{e}$ of weight w.
1.	Choose a random information set I and compute $\boldsymbol{y}_I, G_I$ and G' as above.
2.	Calculate $\boldsymbol{y}' = \boldsymbol{y} - \boldsymbol{y}_I G'$.
3.	Partition I into two subsets X, Y of size k_1 and k_2.
4.	Select in $\{1, \dots n\} \setminus I$ two subsets Z_1 and Z_2 of size ℓ_1 and ℓ_2.
5.	Calculate "weighted sums" having respective weights p_1, p_2 for X and Y, and q_1, q_2 for Z_1 and Z_2.
6.	For each choice of the above sets: Look for collisions, i.e. vectors $\boldsymbol{e}$ such that $\boldsymbol{e}_I$ has exactly weight $p_1 + p_2$ and $\boldsymbol{e}_{Z_1 \cup Z_2}$ has exactly weight $q_1 + q_2$. If $\mathsf{wt}(\boldsymbol{e}_{\{1,\dots,n\} \setminus (I \cup Z_1 \cup Z_2)}) = w - p_1 - p_2 - q_1 - q_2$ then return $\boldsymbol{e}$.
7.	Go back to Step 1.

As it is possible to observe, the major contribution comes in that besides fixing some error positions in the information set, now some positions are also fixed in the set Z, here partitioned into $Z_1 \cup Z_2$. This can be thought of as expanding each point of X and Y into balls of Hamming radius q_1 and q_2 (hence the name "ball-collision"); the collisions are then looked for between these balls. Together with the usual tricks of reusing sums and new tricks such as, for example, early aborting, the ball-collision decoding algorithm allows for a further speed-up in the overall cost of attacking a McEliece ciphertext.
Note that even if in the description of the algorithm the parameters can be chosen distinct, for each practical choice they are in fact pairwise coincident, i.e. we have $k_1 = k_2 = k/2$, $\ell_1 = \ell_2$, $p_1 = p_2$ and $q_1 = q_2$. Furthermore, when $q_1 = q_2 = 0$ we reduce to Stern's algorithm (indicated by the authors simply as "collision decoding").

A better understanding of the evolution of the algorithm in its different stages is obtained using a graphical representation, due initially to Overbeck and Sendrier [94] and later expanded by the authors in [17]. We reproduce it below.



To conclude this section, we present a table containing a comparison of the different results obtained to attack the McEliece cryptosystem. The numbers are based on a recent paper by Becker, Joux, May and Meurer [4], presented at Eurocrypt 2012, and include also a previous result from May, Meurer and Thomae [79]. The paper features a further improvement to the algorithm thanks to a twist in a specific step, namely the initial search step.
The running time is measured (asymptotically) as a function of n and R only, that is $T(n,R) = \mathcal{O}(2^{\theta n})$, where $R = k/n$ is the code rate and we define $\theta = f(R) = \lim_{n\to\infty} \frac{1}{n}\log(T(n,R))$.

Table 3.6: Complexity of different general decoding algorithms. The numbers refer to the worst-case scenario where R is close to the Gilbert-Varshamov bound.

	θ
Lee-Brickell	0.05751
Stern	0.05563
Ball-collision	0.05558
MMT	0.05363
BJMM	0.04970

We also have to mention a recent related work by Sendrier [112], called *Decoding One Out of Many*, or simply *DOOM*. As the name suggests, this technique is used when the adversary is in possession of many ciphertexts and is satisfied by decrypting a single one among them. The attack is performed by applying a variant of the collision decoding algorithms that we have just described to a set of say N instances of the decoding problem (i.e. McEliece/Niederreiter ciphertexts). In particular, the author presents a generalized version of [42]. The approach allows for a gain of almost $\sqrt{N}$ operations when N is sufficiently large:

results are given in [112, Table 5] for $N \geq 2^{40}$. Thus, as the author concludes, this attack should be kept in mind when selecting parameters for an application of code-based schemes (e.g. for exchanging session keys) that repeatedly employs the same public key.

3.2.2 Structural Attacks

As opposed to all of the algorithms presented in the previous section, structural attacks target some specific structural weaknesses, sometimes with the help of additional information, and aim to reconstruct the private key, or an equivalent one, in order to decrypt. It is clear that this kind of attack cannot be simply avoided by enlarging the parameters, and most of the time it breaks the cryptosystem completely.
McEliece in [80] already mentions this strategy of attack, although quickly dismissed for the simple reason that there are too many possibilities for S, G and P. This simplistic argument has proven to hold so far, and a structural attack against McEliece in its original form seems hopeless. However, there are many conditions that could quickly alter this conclusion.
It is necessary to choose carefully the family of codes used to generate the keys. For the original McEliece cryptosystem, for example, Goppa codes with a binary generator polynomial produce weak keys. Loidreau and Sendrier in [72] show that these instances are easily recognizable: in fact, the automorphism group of a Goppa code with binary generator polynomial is generated by the Frobenius field automorphism. This results in an attack that makes use of Sendrier's Support Splitting Algorithm (SSA) as in [110].
We have already mentioned that GRS codes, Reed-Muller codes and Gabidulin codes constitute an insecure choice. Other unsuccessful attempts include, for example, concatenated codes [109], elliptic codes [83] and the algebraic-geometric codes proposed by Janwa and Moreno [61], although for the latter only the case of curves with small genus has been cryptanalysed properly.
The general pitfalls to avoid, as summarized in [94], are twofold:

- Families with high performance, like the above cited concatenated codes, turbo-codes or LDPC codes ([1, 86]) are likely to leak some structure due to the high number of low-weight codewords in their duals.
- Families having optimal (as for the GRS codes) or sub-optimal (elliptic codes) combinatorial properties are also dangerous, since minimum-weight codewords are not hard to find and reveal a lot of information about the code structure.

3.2.3 Other Attacks on the General Framework

So far, we have only described attacks that target the OW-CPA security of the McEliece cryptosystem, such as ISD, and mentioned the structural flaws that can compromise its integrity (attacks on the private key). We haven't, instead, analyzed the behavior of the encryption scheme with respect to other security requirements, such as Indistinguishability (Definition 2.10). It is easy to see that both McEliece and Niederreiter, in their original formulations, are vulnerable to this kind of attacks and, in fact, they are not even IND-CPA secure. Consider an adversary $\mathcal{A}$ for McEliece that plays the CPA game as in Definition 2.11. To start, $\mathcal{A}$ is given a public key G; it then chooses two plaintexts $\boldsymbol{m}_0, \boldsymbol{m}_1$, submits them to the encryption oracle and gets back $\psi^* = \mathsf{Enc}_G^{\mathsf{McE}}(\boldsymbol{m}_b)$. To win the game, it is enough for $\mathcal{A}$ to choose a random $b^* \in \{0,1\}$ and encode $\boldsymbol{m}_{b^*}$, then check the weight of $\psi^* - \boldsymbol{m}_{b^*}G$: clearly $b = b^*$ if and only if $\mathsf{wt}(\psi^* - \boldsymbol{m}_{b^*}G) = w$. The attack is trivial for Niederreiter since the scheme is deterministic and obviously can't satisfy an indistinguishability requirement.

We will describe in Section 6.3.2 a simple variant introduced by Nojima, Imai, Kobara and Morozov [89] that achieves IND-CPA security. The variant consists of introducing additional randomness by padding the message with a few random bits, and it is suitable both for McEliece and Niederreiter.
CCA2 security, on the other hand, is a much stronger notion, and it therefore requires a more subtle approach. It is easy to see that both the general McEliece/Niederreiter framework and the IND-CPA variant are vulnerable to a chosen ciphertext attack. Consider an adversary $\mathcal{A}$ for McEliece that plays the CCA2 game as in Definition 2.11. Again, $\mathcal{A}$ is given a public key G; it then chooses two plaintexts $\boldsymbol{m}_0, \boldsymbol{m}_1$, submits them to the encryption oracle and gets back $\psi^* = \mathsf{Enc}_G^{\mathsf{McE}}(\boldsymbol{m}_b)$. At this point, $\mathcal{A}$ can use the decryption oracle in the following way: it flips a random bit of ψ^* and submits the new ciphertext ψ' to the oracle. Since $\psi' \neq \psi^*$, the oracle will accept the query and reply with $\mathsf{Dec}_{\mathsf{sk}}^{\mathsf{McE}}(\psi')$. Now, if the position flipped was part of the support of the error vector generated by $\mathsf{Enc}^{\mathsf{McE}}$, decryption succeeds and $\mathcal{A}$ recovers $\boldsymbol{m}_b$. Otherwise, the oracle outputs $\perp$, in which case $\mathcal{A}$ repeats the process choosing another position. Clearly, this attack works also in the non-binary case, where instead of flipping a bit we are simply changing the value of a specific position. Another, more elegant attack consists of adding a known codeword to the challenge ciphertext, that is, $\psi' = \psi^* + \boldsymbol{c}$ for $\boldsymbol{c} = \boldsymbol{m}'G$. In this way, the decryption oracle will always return a correctly formed plaintext; $\mathcal{A}$ can then recover $\boldsymbol{m}_b$ by subtracting $\boldsymbol{m}'$. It can be shown that a similar attack can be used against Niederreiter.

Note that all of the above attacks allow $\mathcal{A}$ to fully recover the plaintext, thus breaking not only the indistinguishability but also the one-wayness of the cryptosystem (that is, neither McEliece nor Niederreiter are OW-CCA2 secure). Generic constructions that achieve IND-CCA2 security in the random oracle model will be presented in Section 4.5, while IND-CCA2 security in the standard model is the core of Chapter 6.

3.3 New Horizons and Recent Proposals

Since codes with too much evident algebraic structure don't seem to provide a secure choice for McEliece, a new approach is instead being attempted. It consists of introducing just a partial algebraic structure, using clever scrambling techniques to preserve it, while hoping to hide enough of the underlying private code. The core idea is to make use of subfield subcodes as in Section 2.2.3.

3.3.1 Quasi-Cyclic

A first example in this direction was given in 2005 by Gaborit [49] and further pursued by Berger, Cayrel, Gaborit and Otmani [11]. The scheme makes use of the so-called quasi-cyclic codes.

Definition 3.2 Let $N = N_0\ell$ and let π_ℓ be the permutation on $\{0, \dots, N-1\}$ defined by the orbits $\{(0, \dots, \ell-1), (\ell, \dots, 2\ell-1), \dots, ((N_0-1)\ell, \dots, N-1)\}$. We say that a linear code $\mathcal{C}$ of length N is *Quasi-Cyclic* of order ℓ and index N_0 if it is globally invariant under the action of π_ℓ.

We know that cyclic codes admit a generator matrix in circulant form (see Definition 2.23); similarly, a quasi-cyclic code of order ℓ can be described by means of a matrix composed by circulant $\ell \times \ell$ blocks.
The key generation process starts by choosing a Reed-Solomon code in quasi-cyclic form defined over a large alphabet $\mathbb{F}_{q^m}$. This is easy since it's well known that every Reed-Solomon code is in fact a cyclic code [75]; all one needs to do then is to rearrange the support in order to get a quasi-cyclic code. After rearranging and deleting the majority of the blocks (to counter key-recovery attacks tied to the quasi-cyclic structure), the next step consists of transforming the shortened Reed-Solomon code into a quasi-cyclic Generalised Reed-Solomon code. This is accomplished purely by algebraic means by scalar multiplication and matrix multiplication with a diagonal matrix. Finally, the subfield subcode is constructed over $\mathbb{F}_q$ and the resulting block-circulant matrix is the public key. Details of the process are given in Table 3.7.

Table 3.7: The BCGO KeyGen algorithm.

Setup	Fix the public parameters n, k, w, ℓ such that $n = n_0\ell$ and $k \le n - 2mw$. Fix also a finite field $\mathbb{F}_{q^m}$ and a primitive element α, then call $N = q^m - 1$.
1.	Choose a Reed-Solomon code $\mathcal{R}$ of length $N = N_0\ell$ and rearrange the support to get the corresponding quasi-cyclic code $\hat{\mathcal{R}}$ of order ℓ defined by the parity-check matrix $U = (A_0 \vert \dots \vert A_{N_0-1})$.
2.	Select at random n_0 blocks of U and rearrange them in any order to form $U(\boldsymbol{j}) = (A_{j_0} \vert \dots \vert A_{j_{n_0-1}})$.
3.	Let s be an integer between 1 and $\ell - 1$, $\boldsymbol{a}$ be an n_0-tuple of non-zero elements of $\mathbb{F}_{q^m}$ and D the $\ell \times \ell$ diagonal matrix such that $d_{ii} = \beta^{i-1}$, where $\beta = \alpha^{N_0}$. Construct the matrix $U(\boldsymbol{j}, \boldsymbol{a}, s) = (B_0 \vert \dots \vert B_{n_0-1})$, where $B_i = a_i A_{j_i} D^s$.
4.	Compute the trace matrix of $U(\boldsymbol{j}, \boldsymbol{a}, s)$ to obtain the matrix H over $\mathbb{F}_q$. Return the public key $H \in \mathsf{K}_{\mathsf{publ}}$ and the private key $(\boldsymbol{j}, \boldsymbol{a}, s) \in K_{\mathsf{priv}}$.

Thanks to the particular structure of the resulting code, the public key can be expressed in block-circulant form, therefore only the first line of each block needs to be stored. This allows for a considerable reduction in the memory requirements.
We present some sets of parameters for the scheme in the following table.

Table 3.8: Example of parameters for the BCGO scheme (taken from [11, Table 1]).

q^m	ℓ	N_0	w	Name	n	k	q	n_0	Security	Size (bits)
2^{16}	51	1285	50	A_{16}	459	255	2^8	9	80	8160
				B_{16}	510	306		10	90	9792
				C_{16}	612	408		12	100	13056
				D_{16}	765	510		15	120	20400
2^{20}	75	13981	56	A_{20}	450	225	2^{10}	6	80	6750
	93	11275	63	B_{20}	558	279		6	90	8370
	93	11275	54	C_{20}	744	372		8	110	14880

The column "Security", indicates an estimate of the $\log_2$ of the number of binary operations necessary to perform a general decoding attack.
Unfortunately, a much more dangerous attack was presented shortly after by Faugère, Otmani, Perret and Tillich [38], and all of these parameters have been broken in negligible time (ranging from 0.02 to 0.06 seconds). Following the guidelines of [11], the authors of the attack, to which we will refer from now on as FOPT, build a much bigger code (estimated complexity of a general decoding attack of 2^{600}, below) and show that the time necessary to break even this huge set of parameters is not affected if not for a very small factor (0.08 seconds total). Therefore, the scheme has to be considered definitively insecure.

q^m	ℓ	N_0	w	Name	n	k	q	n_0	Security	Size (bits)
2^{16}	255	257	529	QC_{600}	3825	1705	2^8	15	600	113400

3.3.2 Quasi-Dyadic

This scheme was presented by Misoczki and Barreto [85] in 2009 and it features a structure similar to the quasi-cyclic proposal, but using codes in quasi-dyadic form instead.

Definition 3.3 Given a ring R and a vector $\boldsymbol{h} = (h_0, \dots, h_{n-1}) \in \mathsf{R}^n$, the *Dyadic* matrix $\Delta(\boldsymbol{h}) \in \mathsf{R}^{n\times n}$ is the symmetric matrix with components $\Delta_{i,j} = h_{i\oplus j}$, where $\oplus$ stands for bitwise exclusive-or on the binary representations of the indices. The sequence $\boldsymbol{h}$ is called its signature.
If $n = 2^k$, then every $n \times n$ dyadic matrix can be described recursively as

$$\Delta = \begin{pmatrix} A & B \\ B & A \end{pmatrix} \tag{3.1}$$

where each block is a $2^{k-1} \times 2^{k-1}$ dyadic matrix (and where any 1×1 matrix is dyadic).

It is easy to verify that square dyadic matrices of constant dimension n over a fixed ring R form a commutative ring: this is an important feature as we will see later in this section, and again in Chapter 4.

Definition 3.4 A matrix is called *Quasi-Dyadic* of order t if it is a block matrix whose component blocks are $t \times t$ dyadic submatrices.

We denote with $\Delta(t, \boldsymbol{h})$ the matrix $\Delta(\boldsymbol{h})$ truncated to its first t rows. It is clear that, if t divides n, then $\Delta(t, \boldsymbol{h})$ is a quasi-dyadic matrix (of order t). Note, however, that not all quasi-dyadic matrices need to be of the form $\Delta(t, \boldsymbol{h})$. The difference is highlighted in the following table, where every capital letter represents a $t \times t$ dyadic matrix.

Table 3.9: Example of dyadic vs quasi-dyadic matrices. The matrix (a) is $4t \times 4t$ dyadic, its truncation (b) is quasi-dyadic of order $2t$ and (c) is quasi-dyadic of order t.

$$\begin{pmatrix} A & B & C & D \\ B & A & D & C \\ C & D & A & B \\ D & C & B & A \end{pmatrix} \qquad \begin{pmatrix} A & B & C & D \\ B & A & D & C \end{pmatrix} \qquad \begin{pmatrix} A & B & C & D \\ E & F & G & H \end{pmatrix}$$

(a) (b) (c)

We next define a special kind of permutation matrices.

Definition 3.5 Let Π_i be the dyadic matrix $\Delta(\boldsymbol{h})$ whose signature $\boldsymbol{h}$ is the i-th row of the identity matrix. This is called *dyadic permutation* since it is a permutation matrix that preserves the dyadic structure.

In what follows, the main focus will be on dyadic matrices defined over the ring $\mathsf{R} = \mathbb{F}_{q^m}$, the finite field with q^m elements, for a certain prime power q.
The scheme of [85] is based on Goppa codes as in the original McEliece, but these are carefully selected to admit a parity-check matrix in Cauchy form.

Definition 3.6 Given two disjoint sequences $\boldsymbol{v} = (v_1, \dots, v_\ell) \in \mathbb{F}_{q^m}^\ell$ and $\boldsymbol{L} = (L_1, \dots, L_n) \in \mathbb{F}_{q^m}^n$, the *Cauchy matrix* $C(\boldsymbol{v}, \boldsymbol{L})$ is the matrix with components $C_{i,j} = \dfrac{1}{v_i - L_j}$, i.e.

$$C(\boldsymbol{v}, \boldsymbol{L}) = \begin{pmatrix} \dfrac{1}{v_1 - L_1} & \cdots & \dfrac{1}{v_1 - L_n} \\ \vdots & \vdots & \vdots \\ \dfrac{1}{v_\ell - L_1} & \cdots & \dfrac{1}{v_\ell - L_n} \end{pmatrix}. \tag{3.2}$$

Cauchy matrices have the property that all of their submatrices are invertible [108]. Note that in general Cauchy matrices are not necessarily dyadic and vice-versa, but the intersection of these classes is non-empty in characteristic 2.

We know (Tzeng and Zimmermann, [124]) that Goppa codes admit a parity-check matrix in Cauchy form if the generator polynomial is monic and without multiple zeros. In particular, the following theorem holds.

Theorem 3.2 *Let $\varGamma = \varGamma(\alpha_1, \dots, \alpha_n, g)$ be a Goppa code. If the generator polynomial g is monic and separable, i.e. $g(x) = (x - x_0) \dots (x - x_{\ell-1})$, then $\varGamma$ admits a parity-check matrix in Cauchy form $H = C(\boldsymbol{x}, \boldsymbol{\alpha})$.*

The trick to generate a public key in dyadic form is to choose a Goppa code that allows a parity-check matrix that is simultaneously dyadic and Cauchy. Misoczki and Barreto show that this intersection is non-empty in [85, Th. 2].

Theorem 3.3 *Let H be an $n \times n$ matrix over $\mathbb{F}_{q^m}$ such that $H = \Delta(\boldsymbol{h})$ for a certain signature $\boldsymbol{h} \in \mathbb{F}_{q^m}^n$ and $H = C(\boldsymbol{v}, \boldsymbol{L})$ for two disjoint sequences $\boldsymbol{v}, \boldsymbol{L} \in \mathbb{F}_{q^m}^n$. Then $\mathbb{F}_{q^m}$ has characteristic 2, $\boldsymbol{h}$ satisfies*

$$\frac{1}{h_{i \oplus j}} = \frac{1}{h_i} + \frac{1}{h_j} + \frac{1}{h_0} \tag{3.3}$$

and we have $v_{i+1} = 1/h_i + \omega$ and $L_{j+1} = 1/h_j + 1/h_0 + \omega$ for a certain offset $\omega \in \mathbb{F}_{q^m}$.

A method to solve Equation 3.3 is provided in Algorithm 1 of the same paper, and it consists of choosing distinct non-zero h_0 and h_{2^c}, for $0 \leq c \leq \log_2 n$, then assigning

$$h_{i+j} = \frac{1}{\frac{1}{h_i} + \frac{1}{h_j} + \frac{1}{h_0}} \tag{3.4}$$

for all $0 < j < i$ (so that $i + j = i \oplus j$). To make sure that this value is well defined, we choose all the elements of the signature to be distinct. Details are given in the following table.

Table 3.10: Constructing a Goppa code in dyadic form ([85, Algorithm 1]).

Input	An integer $q = 2^c$, an extension degree m and parameters $n \leq q/2$, ℓ.
Output	The support $\alpha_1, \dots, \alpha_n$, generator polynomial g and parity-check matrix H for the Goppa code $\varGamma = \varGamma(\alpha_1, \dots, \alpha_n, g)$ with minimum distance $d = 2\ell+1$ over $\mathbb{F}_q$.
1.	Choose the dyadic signature: i. Set $U = \mathbb{F}_{q^m} \setminus \{0\}$, assign h_0 at random in U, then remove h_0 from U. ii. For each h_i where i is a power of 2, assign h_i at random in U, then compute $h_{i+j} = 1/(1/h_i + 1/h_j + 1/h_0)$ for $j = 1, \dots, i-1$. Remove all the elements just assigned from U. iii. Terminate when length n is reached. The signature is $\boldsymbol{h} = (h_1, \dots, h_n)$.
2.	Assign the offset ω at random in $\mathbb{F}_{q^m}$.
3.	Compute the elements $x_i = 1/h_i + \omega$ for $i = 0, \dots, \ell - 1$ and return the Goppa polynomial $g(x) = \prod_{i=0}^{\ell-1} (x - x_i)$.
4.	Calculate the support $\alpha_{i+1} = 1/h_j + 1/h_0 + \omega$ for $i = 0, \dots, n - 1$.
5.	Return $\alpha_1, \dots, \alpha_n$, g and $H = \Delta(\ell, \boldsymbol{h})$.

The above algorithm is the core of the key generation process. The idea is to start from a fully dyadic code, and then select, permute and scale the columns (block by block) before applying the subfield subcode technique in a similar way as in [11].

Table 3.11: The Misoczki-Barreto KeyGen algorithm.

Setup	Fix a finite field $\mathbb{F}_{q^m} = \mathbb{F}_{2^u}$ where $q = 2^c$, $u = mc$. Choose a code length $n < q^m$, with $n = n_0\ell$ for ℓ such that $m\ell < n$.
1.	Call the algorithm in Table 3.10 to produce a dyadic matrix $H = \Delta(\ell, \boldsymbol{h})$ over $\mathbb{F}_{q^m}$, with $\boldsymbol{h}$ having length $N = N_0\ell > n$.
2.	Partition H into N/ℓ dyadic blocks $(A_0|\dots|A_{N/\ell-1})$ of size $\ell \times \ell$.
3.	Select at random n_0 blocks among the previous, together with n_0 dyadic permutations $\Pi_{j_0}, \dots, \Pi_{j_{n_0-1}}$ and n_0 scale factors $a_0, \dots, a_{n_0-1} \in \mathbb{F}_q$.
4.	Form the matrix $H' = (a_0 A_{i_0} \Pi_{j_0}|\dots|a_{n_0-1} A_{i_{n_0-1}} \Pi_{j_{n_0-1}})$.
5.	Compute the co-trace matrix of H' over the base field $\mathbb{F}_q$ and its systematic form $(M|I_{n-k})$.
6.	Return the public key $M \in \mathsf{K}_{\mathsf{publ}}$ and the private key $H' \in \mathsf{K}_{\mathsf{priv}}$.

Note that all the operations involved in the key generation process preserve the dyadicity of the matrix, including the use of dyadic permutations, the co-trace construction, and the block operations performed during the final Gaussian elimination. In this way, the public key will be composed of dyadic submatrices each of which can be represented compactly by its signature, therefore saving a factor of ℓ in the public key size. Since M is $(n-k) \times k = m\ell \times k$ and is $\ell \times \ell$ block dyadic, it requires only $km\ell/\ell = km$ field elements for storage, equivalent to kmc bits.

Remark 3.3 The algorithm presented by Misoczki and Barreto runs in polynomial time. Since every element of the signature is assigned a value exactly once, the running time is $\mathcal{O}(n)$ steps. The authors in [85] did not give a lower bound for the number of possible *distinct* codes, but only the upper bound $\binom{N/\ell}{n_0} \cdot n_0! \cdot t^{n_0} \cdot \prod_{i=0}^{\lceil \log N \rceil} (q^m - 2^i)$ (due to, respectively, selection, rearrangement, permutations of the blocks and number of signatures generated by the algorithm). It is believed that the algorithm does produce close to this number of codes, but it is too hard to actually state the exact number of *distinct* codes constructible.

Several sets of parameters are proposed in the original paper. We report them in Table 3.12. For the last set of parameters, the paper provides also a comparison with common cryptographic schemes such as RSA, to show that, for relatively similar key sizes, the scheme based on quasi-dyadic codes enjoys much faster encryption/decryption. Unfortunately, almost all of the parameters proposed have been broken by the FOPT attack that we already cited, and that we will present in the next section in detail.

Table 3.12: Example of parameters for the Misoczki-Barreto scheme ([85, Tables 2-5]).

q	m	n	k	ℓ	Security	Size (bits)
2	16	3584	1536	128		24576
2^2	8	3584	1536	256	128	24576
2^4	4	2048	1024	256		16384
2^8	2	1280	768	256		12288
		1024		256	168	
2^8	2	768	512	128	136	8192
		640		64	102	
		8192	4096	256	256	65536
		7168	3072	256	192	49152
2	16	4096	2048	128	128	32768
		3072	2048	64	112	32768
		2560	1536	64	80	24576
		1536	1024	256	256	16384
		1280	768	256	192	12288
2^8	2	768	512	128	128	8192
		640	384	128	112	6144
		512	256	128	80	4096

3.3.3 FOPT

In this section we summarize the structural attack by Faugère, Otmani, Perret and Tillich [38]. It relies on the fundamental property of coding theory $H \cdot G^{\mathsf{T}} = 0$ to build an algebraic system, using then Gröbner bases techniques to solve it.

Table 3.13: The FOPT algorithm.

Input	A $k \times n$ generator matrix $G = \{g_{i,j}\}$ for the subcode $\mathcal{C}\vert_{\mathbb{F}_q}$, G being a matrix formed of $\ell \times \ell$ blocks, with $k = k_0\ell, n = n_0\ell$, over $\mathbb{F}_q = \mathbb{F}_{2^c}$.
Output	A parity-check matrix in alternant form $H = \{y_i x_i^j\}$ for $\mathcal{C}$ over $\mathbb{F}_{q^m}$.
1.	Generate the following system of equations in the unknowns $X = \{X_i\}$ and $Y = \{Y_i\}$: $$\left\{g_{i,0}Y_0X_0^j + \cdots + g_{i,n-1}Y_{n-1}X_{n-1}^j = 0 : 0 \leq i \leq k-1, 0 \leq j \leq \ell-1\right\}. \quad (3.5)$$
2.	Choose $n_{Y'} \geq n - k$ variables Y' from Y, and use the equations to express all other variables in $Y \setminus Y'$ as polynomials in Y'. We call the Y' variables "free" and the remaining "dependent".
3.	Compute the projection of the solutions with respect to the variables Y'.
4.	Having determined the Y', the system will simplify to $$\left\{g'_{i,0}X_0^j + \cdots + g'_{i,n-1}X_{n-1}^j = 0 \ : \ 0 \leq i \leq k-1, 0 \leq j \leq \ell-1\right\}. \quad (3.6)$$
5.	Consider now the subset of the equations having degree equal to a power of two, i.e. $j = 2^l$, for $l = 1, \ldots, \log_2(\ell - 1)$.
6.	Use the Frobenius automorphism to produce a system over $\mathbb{F}_2$, consisting of mcn unknowns and $mc \log_2(\ell-1)k$ equations.
7.	Solve the system for X_i and return H.

The key idea of the algorithm is that the codes in use are part of the alternant family, and therefore it looks for a parity-check matrix that, even if different from the private key, still allows efficient decoding. Observe that for all suitable choices of cryptographic parameters, we have that $\log_2(\ell-1)k > n$, hence the system produced in Step 6 is easily solvable.

The special properties of the structured codes used in the variants presented above are of key importance, as they contribute to considerably reduce the number of unknowns of the system. Some relations, peculiar of each of the two schemes, are in fact derived from these properties, and then used in the context of Step 2 to determine the number of free variables $n_{Y'}$ and simplify the system. These are presented below:

Table 3.14: Properties for the quasi-cyclic (left) and quasi-dyadic (right) schemes.

$$\begin{cases} X_{j\ell+i} = X_{j\ell}\beta^i \\ Y_{j\ell+i} = Y_{j\ell}\beta^{ie} \end{cases}$$

for $0 \le j \le n_0 - 1$, $0 \le i \le \ell - 1$ and an integer $e \in \{0, \dots, \ell-1\}$ picked secretly[4].

$$\begin{cases} Y_{j\ell+i} = Y_{j\ell} \\ X_{j\ell+i} + X_{j\ell} = X_i + X_0 \\ X_{j\ell+(i\oplus i')} = X_{j\ell+i} + X_{j\ell+i'} + X_{j\ell} \end{cases}$$

for $0 \le j \le n_0 - 1$ and $0 \le i, i' \le \ell - 1$.

Now, in some cases this number is very small (e.g. 1 or 2); an exhaustive search thus leads already to a practical attack. Otherwise, the technique used is to find a projection of the solutions with respect to the variables of the block Y', which can be done, as anticipated, by computing a Gröbner basis. This is by far the most expensive part of the algorithm.

Applying the relations in Table 3.14 to the general framework it is possible to deduce the following scenarios for the two schemes, where r such that $rm = n-k$ is the order of the alternant form (in the quasi-dyadic case, we have $r = \ell$):

Table 3.15: System specifications for the quasi-cyclic (left) and quasi-dyadic (right) schemes.

	QC	QD
Unknowns Y_i	$n_0 - 1$	$n_0 - 1$
Unknowns X_i	$n_0 - 1$	$n_0 - 2 + \log_2 \ell$
Linear equations involving only the Y_i	k_0	$n_0 - m$
Non-linear equations containing monomials of the form $Y_i X_i^{\xi}$, for $\xi > 0$	$(r-1)k_0$	$\ell(\ell-1)(n_0 - m)$

[4]For the purpose of the attack, e is assumed to be known, even just via an exhaustive search.

To prove the above values is a matter of a few easy calculations. We report them for the quasi-dyadic case, which is the one we are most interested in.
In this case, the first property in Table 3.14 states that the Y_i of each block are all equal, thus there are $n/\ell = n_0$ distinct variables. We can arbitrarily choose one of them, which explains $n_0 - 1$. Moreover, because of the dyadicity of G, the linear equations in the Y_i are identical, hence redundant, for all the rows of each dyadic block. So we have $k/\ell = (n - m\ell)/\ell = (n_0\ell - m\ell)/\ell = \ell(n_0 - m)/\ell = n_0 - m$ linear equations as claimed.
The other two values are a direct consequence of the second and third properties: in fact, we can fix arbitrarily two variables, say X_0 and X_ℓ and express every other in terms of those two for each block, which means $n_0 + \log_2 \ell - 2$. As for the non-linear equations there are exactly $\ell k - k = (\ell - 1)k$ of them, and since $k = \ell(n_0 - m)$ as we just saw, we obtain the claimed value of $\ell(\ell - 1)(n_0 - m)$ (the dyadicity of G doesn't have an impact when $\xi > 1$ unlike the linear case).

Further analysis has been conducted by the same authors in [39], where the complexity of the attack is studied more carefully. The algebraic system described by (3.5) is seen as an affine bi-linear system with equations of bi-degree $(d_1, d_2) = (2^j, 1)$ and $n_{X'} + n_{Y'}$ unknowns, where $n_{X'}$ is the number of X_i variables obtained after the reduction in Step 2. A theoretical estimate is provided.

Proposition 3.1 *Let $D = \min(n_{X'} + 1, n_{Y'} + 1)$ and denote by $R_{a,b}$ the vector space of bihomogeneous polynomials of bi-degree (a, b) over the polynomial ring R. Then the time complexity (field operations) of computing a Gröbner basis of (3.5) is approximated by*

$$T_{theo} \approx \sum_{\substack{d_1+d_2=D \\ 1\le d_1,d_2\le D-1}} \Big(\dim(R_{d_1,d_2}) - [t_1^{d_1} t_2^{d_2}] HS(t_1, t_2)\Big) \dim(R_{d_1,d_2}) \qquad (3.7)$$

where $[t_1^{d_1} t_2^{d_2}] HS(t_1, t_2)$ stands for the coefficient of the term $t_1^{d_1} t_2^{d_2}$ in the Hilbert bi-series[5] *$HS(t_1, t_2)$.*

The experimental results obtained by running the algorithm on the set of parameters proposed in [11, 85] prove to be reasonably close to this bound, even if this is far from being tight. At the current time, no further analysis has been conducted and the numbers provided by the bound can be interpreted as a good approximation of the overall costs of the algorithm.
While this clearly doesn't fully assess the security of the scheme, it is enough to discard many weak sets of parameters.

A summary is given in [39, Tables 1-2] and presented below.

[5] See [39, Appendix A].

Table 3.16: Summary of the complexities for the FOPT attack.

	Name	q	m	ℓ	n_0	$n_{X'}$	$n_{Y'}$	Security	Time(s)	Ops	T_{theo}
QC	A_{16}	2^8	2	51	9	8	3	80	0.06	$2^{18.9}$	2^{17}
	B_{16}				10	9	3	90	0.03	$2^{17.1}$	2^{18}
	C_{16}				12	11	3	100	0.05	$2^{16.2}$	2^{20}
	D_{16}				15	14	4	120	0.02	$2^{14.7}$	2^{26}
	A_{20}	2^{10}	2	75	6	5	2	80	0.05	$2^{15.8}$	2^{10}
	B_{20}			93	6	5	2	90	0.05	$2^{17.1}$	2^{10}
	C_{20}			93	8	7	2	110	0.02	$2^{14.5}$	2^{11}
	QC_{600}	2^8	2	255	15	14	3	600	0.08	$2^{16.6}$	2^{21}
QD	Table 2	2^2	8	64	56	59	7	128	1776.3	$2^{34.2}$	2^{65}
		2^4	4		32	36	3		0.50	$2^{22.1}$	2^{29}
		2^8	2		12	16	1		0.03	$2^{16.7}$	2^8
	Table 3	2^8	2	64	10	14	1	102	0.03	$2^{15.9}$	2^8
				128	6	11		136	0.02	$2^{15.4}$	2^7
				256	4	10		168	0.11	$2^{19.2}$	2^7
	Table 5	2^8	2	128	4	9	1	80	0.06	$2^{17.7}$	2^6
				128	5	10		112	0.02	$2^{14.5}$	2^7
				128	6	11		128	0.01	$2^{16.6}$	2^7
				256	5	11		192	0.05	$2^{17.5}$	2^7
				256	6	12		256	0.06	$2^{17.8}$	2^7

As it is possible to observe, the attack is very successful except for a single case (line 9, 1776.3 seconds $\simeq$ half an hour). In fact, the complexity clearly increases proportionally to the value $\rho = D - 1 = \min(n_{X'}, n_{Y'})$. For the quasi-dyadic case, this is exactly $\rho = m - 1$ (see Table 3.15), so unlike the other case, it does not depend on the code parameters (length, dimension) but on the field chosen. It is also immediate to notice that the parameters for Table 4 of [85] are missing: for these parameters we have $\rho = 15$ and the authors report that they didn't manage to efficiently solve the system. This suggests that the time necessary for the computation is beyond the range of the machine in use for the tests (Xeon bi-processor 3.2Ghz, with 16 Gb of Ram); however another phenomenon occurs for binary Goppa codes (see Remark 3.4 below).
The authors conclude that any system with $\rho \leq 20$ should be within the scope of the attack.

Remark 3.4 For binary quasi-dyadic Goppa codes the analysis is less accurate. In this case, in fact, it is easy to compute the Gröbner basis for the system, but this is somehow "trivial", i.e. reduced to just one equation, therefore not providing enough information. This is typically due to the fact that only a subset of the equations is used (the ones with bi-degree $(2^j, 1)$). As a result, the variety associated is too big, and the attack cannot be mounted efficiently. To understand how to use all the equations in a more clever way in this case, remains an open problem.

A Quasi-Dyadic Variant of McEliece using Generalized Srivastava Codes

4.1 Introduction

The McEliece cryptosystem (Table 3.1) is one of the main candidates for the post-quantum era. It has a very fast and efficient encryption procedure, and there are no known vulnerabilities against quantum algorithms: in a recent paper by Bernstein [15] it is shown that the speedup in general decoding attacks (e.g. ISD) requires to increase the key sizes by a factor of four. In the previous chapter we have seen that, although the original McEliece has resisted cryptanalysis so far, it has one big flaw: the size of the public key. Our proposal is based on Generalized Srivastava codes (Definition 2.34) and represents a generalization of the scheme of Misoczki and Barreto [85], with the advantage of a better flexibility. By flexibility we mean the following: there is an intersection between the families of Goppa codes and Generalized Srivastava codes (which includes the original Srivastava codes), corresponding to a fixed, particular choice of parameters. In our construction the parameters can instead be chosen in various ways, in order to maximize the reduction in the key size, or to comply with higher levels of security. In particular, we claim a greater resistance to the known structural attacks, while the keys have similar size to the ones presented in [85].
The chapter is organized as follows: Section 4.2 contains a precise description of the construction. Details about security are given in Section 4.4, as well as a choice of parameters and a brief comparison with the Misoczki-Barreto scheme. In Section 4.5 we present the implementation results obtained in a joint work with Pierre-Louis Cayrel and Gerhard Hoffmann [22]. These comprise the implementation on a computer processor and an embedded device, both for a "plain" McEliece scheme, and for an IND-CCA2 secure variant of it. Finally, we conclude in Section 4.6.

4.2 Construction

Our proposal is to use GS codes (Definition 2.34) instead of Goppa codes in the context of the quasi-dyadic scheme presented in Table 3.3.2. Note that GS codes are also alternant codes, hence it is possible to use the efficient alternant decoding algorithm (Table 2.7). According to Sarwate [107, Cor. 2] the complexity of decoding is $\mathcal{O}(n \log^2 n)$, which is the same as for Goppa codes; thus GS codes are another suitable choice for a McEliece-type cryptosystem.
Recall the special form for the parity-check matrix of a GS code given in (2.22). Now, it is evident that an equivalent parity-check matrix (by a row permutation) is given by

$$\hat{H} = \begin{pmatrix} \hat{H}_1 \\ \hat{H}_2 \\ \vdots \\ \hat{H}_t \end{pmatrix} \tag{4.1}$$

where each block is

$$\hat{H}_i = \begin{pmatrix} \dfrac{z_1}{(\alpha_1 - w_1)^i} & \cdots & \dfrac{z_n}{(\alpha_n - w_1)^i} \\ \dfrac{z_1}{(\alpha_1 - w_2)^i} & \cdots & \dfrac{z_n}{(\alpha_n - w_2)^i} \\ \vdots & \vdots & \vdots \\ \dfrac{z_1}{(\alpha_1 - w_s)^i} & \cdots & \dfrac{z_n}{(\alpha_n - w_s)^i} \end{pmatrix}. \tag{4.2}$$

Our idea is to start from a Goppa code in dyadic form, as output by the algorithm in Table 3.10, and to apply some operations to transform it into a GS code while preserving the quasi-dyadic structure. The above equivalent form for the parity-check matrix, in fact, suggests that it is enough to take successive powers of the first block, and then multiply by a diagonal matrix. In our key generation process we use an updated version of the algorithm, introduced by Barreto et al. in [2] as Algorithm 2. The main idea is to generate a signature of the maximum possible length $q^m - \ell$ and then discard the block(s) containing undefined entries.

Table 4.1: The new KeyGen algorithm.

Setup	Fix a finite field $\mathbb{F}_{q^m} = \mathbb{F}_{2^u}$ where $q = 2^c$, $u = mc$. Choose a code length $n < q^m$, with $n = n_0 s$ and s being a power of 2. The parameters s, t are chosen such that $mst < n$. More details about the choice of s and t will be given later.
1.	Call the algorithm in Table 3.10 to produce a dyadic matrix $H = \Delta(s, \boldsymbol{h})$ over $\mathbb{F}_{q^m}$, with $\boldsymbol{h}$ having length n
2.	Set $\hat{H}_1 = H$ with $w_i = v_i$, $\alpha_j = L_j$. Since we are in characteristic 2, we have: $$v_i - L_j = v_i + L_j = w_i + \alpha_j = \alpha_j + w_i = \alpha_j - w_i.$$ for all $i = 1, \dots, s$, $j = 1, \dots, n$. Note that this block is dyadic (of order s) as it defines a GS code with $t = 1$, equivalent to a Goppa code.
3.	Form the remaining blocks by consecutive powers, up to the power of t. This means $\hat{H}_2$ is obtained by squaring each element of $\hat{H}_1$, $\hat{H}_3$ is obtained by cubing, and so on.
4.	Pick the z_i uniformly at random with the following restriction: $$z_{is+j} = z_{is} \text{ for } i = 0, \dots, n_0 - 1, \; j = 1, \dots, s.$$
5.	Compute the matrix $H' = \hat{H} \cdot \mathrm{Diag}(z_i)$ and its co-trace matrix over the base field $\mathbb{F}_q$ in its systematic form $(M \vert I_{n-k})$, having $k = n - mst$ with high probability (see Section 4.3.1).
6.	Return the public key $M \in \mathsf{K}_{\mathsf{publ}}$ and the private key $H' \in \mathsf{K}_{\mathsf{priv}}$.

Note that, in addition to all the other operations, we also choose the z_i to be equal s-wise in order to preserve the dyadicity. Since M is $(n-k)\times k = mst \times k$ and is $s \times s$ block dyadic, it requires only $kmst/s = kmt$ field elements for storage, equivalent to $kmtc$ bits.

Remark 4.1 As pointed out above, we decode GS codes by means of the alternant decoding algorithm, starting from a parity-check matrix $H(\boldsymbol{x}, \boldsymbol{y})$ as in (2.13). Recall that there is a 1-1 correspondence between the roots of the error locator polynomial $\Lambda(z)$ and the error positions: in fact, there is an error in position i if and only if $\Lambda(1/x_i) = 0$. Of course, if one of the x_i's is equal to 0, it is not possible to find the root, and to detect the error.
Now, the generation of the error vector is random, hence we can assume the probability of having an error in position i to be around $st/2n$; since the codes give the best performance when mst is close to $n/2$, we can estimate this probability as $1/4m$, which is reasonably low for any non-trivial choice of m; however, we still argue that the code is not fully decodable and we now explain how to adapt the key generation algorithm to ensure that all the x_i's are non-zero.
As part of the key generation algorithm we assign to each x_i the value L_i, hence it is enough to restrict the possible choices for ω to the set $\{\alpha \in \mathbb{F}_{q^m} : \alpha \neq 1/h_i + 1/h_0,\ i = 0, \ldots, n-1\}$. In doing so, we considerably restrict the possible choices for ω but we ensure that the decoding algorithm works properly.

4.3 Correctness of Key Generation

4.3.1 Full-rank Matrices

We give an estimate of the expected probability of having an invertible submatrix after the co-trace operation defined in Step 5 of the key generation algorithm. The aim of this section is just to provide a techincal explanation of why row reduction to the systematic form is actually possible, and happens with high probability; therefore, it may be skipped by the reader.

We start by considering random matrices as a general case.

Lemma 4.1 *Let M be a random $n \times n$ matrix over the finite field $\mathbb{F}_q$. Then the probability that M is non-singular is:*

$$p = \frac{\prod_{i=1}^{n} \left(q^n - q^{i-1}\right)}{q^{n^2}}.$$

Proof A matrix M is non-singular if and only if its rows are linearly independent vectors. The choices for the first row are $q^n - 1$, while for each row after the first, we have to be sure that it is not in the span of the previous vectors; hence for the i-th row we have only $q^n - q^{i-1}$ choices. This gives $(q^n-1)(q^n-q)\ldots(q^n-q^{n-1})$ choices over the total q^{n^2}, which is what we wanted to prove. $\triangle$

Now, we take into account the special form of our matrix. Since it is dyadic, the number of choices for the row vectors is restricted, since every time we choose a row, the following $s-1$ are uniquely determined according to the dyadic form (permutations). Practically speaking, we are considering an $r \times r$ quasi-dyadic matrix, where $r = mst = r_0 s$, and we are choosing only r_0 row vectors.

However now, in each choice, we must also ensure that the set of s rows produced is by itself linearly independent. Since each of those is composed by r_0 square blocks of side s, we first focus on a single block.

Lemma 4.2 *Let $D = \Delta(\boldsymbol{h})$ be an $s \times s$ matrix over the finite field $\mathbb{F}_{q^m}$ $(q = 2^\lambda)$ given by the signature $\boldsymbol{h} = (h_0, \dots, h_{s-1})$, with s being a power of 2. Then:*

$$D \text{ is singular} \iff \sum_{i=0}^{s-1} h_i = 0.$$

Proof Since s is a power of 2, say 2^j, we know D is of the following form:

$$D = \begin{pmatrix} A & B \\ B & A \end{pmatrix}$$

where A, B are dyadic submatrices of dimension 2^{j-1} defined, respectively, by $\boldsymbol{h}_A = h_0, \dots, h_{s/2-1}$ and $\boldsymbol{h}_B = h_{s/2}, \dots, h_{s-1}$. All we need is to consider the determinant of D. Recall from Section 3.3.2 that dyadic matrices form a commutative ring, hence in particular A and B commute. We can then invoke a generalization of Silvester [118] (see Section 4.3.2) and claim that $\det D = \det(A^2+B^2)$. Applying the argument recursively (and remembering that we are in characteristic 2) we arrive at the conclusion that $\det D = (h_0 + \cdots + h_{s-1})^{2^j}$. Now, D is singular $\iff \det D = 0 \iff (h_0 + \cdots + h_{s-1})^{2^j} = 0 \iff h_0 + \cdots + h_{s-1} = 0$, which terminates the proof. $\triangle$

Thanks to Lemma 4.2 it is now easy to give a description of how to select the first row. We call a row vector v *good* if the set of s vectors consisting of v and its dyadic rearrangements is linearly independent, and we call v *bad* if it is not good. Now, for every choice of $s-1$ field elements, the sum will still be a field element; hence, for each block we have q^{s-1} signatures that sum to 0, and overall $(q^{s-1})^{r_0}$ bad sequences. It is then sufficient to subtract this number from the total possible choices q^r, and we obtain that the number of good choices for the first row vector is:

$$q^r - (q^{s-1})^{r_0} = q^r - q^{r_0(s-1)} = q^r - q^{r-r_0} = q^{r-r_0}(q^{r_0} - 1).$$

Let's call $\mathcal{G}$ the set of all good rows. As a last precaution, we need to determine how many linear combinations of the rows in a size-s set produce a row which is still in $\mathcal{G}$, so that we can exclude them at the moment of choosing the next one.

This is easy for the first set.

Lemma 4.3 *Let $v^{(1)}, \dots, v^{(s)}$ be the first s row vectors of a quasi-dyadic matrix, and suppose the first row is good. Then for every $v = \sum_{i=1}^{s} a_i v^{(i)}$:*

$$v \in \mathcal{G} \Longleftrightarrow \sum_{i=1}^{s} a_i \neq 0.$$

Proof Let's analyze, without loss of generality, the first block and write:

$$\begin{gathered} v_1 + v_2 + \cdots + v_s = \\ = (a_1 v_1^{(1)} + a_2 v_1^{(2)} + \cdots + a_s v_1^{(s)}) + \cdots + (a_1 v_s^{(1)} + a_2 v_s^{(2)} + \cdots + a_s v_s^{(s)}) = \\ = (a_1 v_1^{(1)} + a_1 v_2^{(1)} + \cdots + a_1 v_s^{(1)}) + \cdots + (a_s v_1^{(s)} + a_s v_2^{(s)} + \cdots + a_s v_s^{(s)}) = \\ = a_1 \sum_{i=1}^{s} v_i^{(1)} + a_2 \sum_{i=1}^{s} v_i^{(2)} + \cdots + a_s \sum_{i=1}^{s} v_i^{(s)}. \end{gathered}$$

Now, each of these sums is exactly the sum of the elements of each row, which because of the dyadicity is constant, say equal to α, and by hypothesis different from 0; hence we can write $\alpha(a_1 + \cdots + a_s) = 0 \Longleftrightarrow a_1 + \cdots + a_s = 0$, which terminates the proof. $\triangle$

According to Lemma 4.3 then, $q^{s-1}(q-1)$ linear combinations of the rows in the first set produce a row in $\mathcal{G}$. Unfortunately the same reasoning doesn't work when we consider the next sets, as the rows in the next set will sum in principle to a different element (say β, γ etc.). Hence, we can just obtain a lower bound, by excluding all the q^s linear combinations. However, it is reasonable to think that very few linear combinations produce a bad row, so our lower bound is not far from the real value.

Theorem 4.1 *Let H be an $r \times n$ parity-check matrix over $\mathbb{F}_q$ as in Step 5, with $r = mst = r_0 s$. Then the row-reduction to the systematic form for H succeeds with probability at least:*

$$p = \prod_{i=0}^{r_0-1} \Big(1 - \frac{1}{q^{r_0}} - \frac{1}{q^{(r_0-i)s}}\Big).$$

Proof Follows directly from our last argument: we get $p = \dfrac{\prod_{i=0}^{r_0-1} \Big(q^r - q^{r-r_0} - q^{is}\Big)}{q^{r_0 r}}$. This is a product of r_0 terms and since $q^{r_0 r} = (q^r)^{r_0}$ we can divide each term by q^r and obtain the conclusion. $\triangle$

Experimental results suggest this number looks roughly like $(q-1)/q$.

4.3.2 Determinant of Block Matrices

We state the following result, which we will need to prove Lemma 4.2:

Lemma 4.4 *Let D be an $n \times n$ block-symmetric matrix over a finite field $\mathbb{F}$ of characteristic 2, i.e. D is in the form:*

$$D = \begin{pmatrix} A & B \\ B & A \end{pmatrix}.$$

where A and B are themselves block-symmetric matrices of dimension $n/2$.
If A and B commute, then $\det D = \det(A^2 + B^2)$.

Proof We know from [118] that $\det \begin{pmatrix} A & B \\ \mathbf{0} & C \end{pmatrix} = \det \begin{pmatrix} A & \mathbf{0} \\ B & C \end{pmatrix} = \det A \det C$.

Now, consider the following product $M = \begin{pmatrix} A & B \\ B & A \end{pmatrix} \begin{pmatrix} A & \mathbf{0} \\ B & \mathbf{I} \end{pmatrix}$.

Since A and B are both symmetric and commute, we have that $A = A^\intercal$, $B = B^\intercal$ and $AB = (AB)^\intercal$, hence we can rewrite the product as:

$$M = \begin{pmatrix} A & B \\ B^\intercal & A \end{pmatrix} \begin{pmatrix} A^\intercal & \mathbf{0} \\ B & \mathbf{I} \end{pmatrix} = \begin{pmatrix} A^2 + B^2 & B \\ B^\intercal A^\intercal + AB & A \end{pmatrix} =$$

$$= \begin{pmatrix} A^2 + B^2 & B \\ (AB)^\intercal + AB & A \end{pmatrix} = \begin{pmatrix} A^2 + B^2 & B \\ \mathbf{0} & A \end{pmatrix}.$$

Looking at determinants, and applying the hypothesis, we read:

$$\det M = \det D \det A = \det(A^2 + B^2) \det A$$

which implies in particular $(\det D + \det(A^2 + B^2)) \det A = 0$ and the result follows immediately if we assume $\det A \neq 0$. However, we don't even need this assumption if we use the following trick: instead of working over $\mathbb{F}$, let's do our calculations over the corresponding polynomial ring $\mathbb{F}[x]$ by defining $A_x = A + x\mathbf{I}$ and $D_x = \begin{pmatrix} A & B \\ B & A_x \end{pmatrix}$.

We obtain $(\det D_x + \det(AA_x + B^2)) \det A_x = 0$ but now this time we are considering a product of polynomials and $\det A_x = \det(A + x\mathbf{I})$ is certainly not the zero polynomial, hence the left-hand side must be.
Thus $\det D_x = \det(AA_x + B^2)$ follows, from which it is enough to put $x = 0$ to get our result. $\triangle$

4.4 Security

4.4.1 Cryptanalysis

It is clear that, since GS codes also belong to the class of alternant codes, the main security issue is the FOPT attack (Table 3.13). As we will see, this can be applied to our proposal directly, with the system properties (Table 3.14) holding in a similar way.
Despite the absence of a precise criterion for assessing the security, it makes sense to compare the different security levels for the Misoczki-Barreto scheme and for our scheme. In fact, we can think of a Goppa code or a GS code with the same parameters $[n, k, d]$ having, respectively, $k = n - m\ell = n - mst \Longrightarrow \ell = st$. If $t = 1$ then our scheme is exactly the same as [85]. For $t > 1$, however, the system parameters change, as $n = n_0\ell = n_0's$ having $n_0' > n_0$. We now focus our attention on the linear part of the system: just like before, it is possible to prove that all the Y_i in a block are equal.

Proposition 4.1 *Let Y_i be the set of unknowns defined in (3.5). Then:*

$$Y_{is+j} = Y_{is} \text{ for } i = 0, \ldots, n_0 - 1, j = 1, \ldots, s.$$

Proof Recall from Definition 2.34 the specifications for the particular alternant form of GS codes. Now, we want to prove that $y_{is+j} = y_{is}$ for $i = 0, \ldots, n_0 - 1$, $j = 1, \ldots, s$. Let's then fix a specific i (i.e. choose a block) and consider in particular $y_{is+j^*} = y_{is}$, for any $j^* \in \{1, \ldots, s\}$.

If we can prove that $\prod_{j=1}^{s}(\alpha_{is+j^*} - w_j) = \prod_{j=1}^{s}(\alpha_{is} - w_j)$, then obviously

$$\prod_{j=1}^{s}(\alpha_{is+j^*} - w_j)^t = \prod_{j=1}^{s}(\alpha_{is} - w_j)^t \Longrightarrow \frac{1}{\prod_{j=1}^{s}(\alpha_{is+j^*} - w_j)^t} = \frac{1}{\prod_{j=1}^{s}(\alpha_{is} - w_j)^t}.$$

We know that $z_{is+j} = z_{is}$ for $i = 0, \ldots, n_0 - 1, j = 1, \ldots, s$ by construction.

Hence $\dfrac{z_{is+j}}{\prod_{j=1}^{s}(\alpha_{is+j^*} - w_j)^t} = \dfrac{z_{is}}{\prod_{j=1}^{s}(\alpha_{is} - w_j)^t}$, and this means $y_{is+j} = y_{is}$.

Since this does not depend on the choice of i, it is then true for all i, and we obtain our result.

It remains to prove $\prod_{j=1}^{s}(\alpha_{is+j^*} - w_j) = \prod_{j=1}^{s}(\alpha_{is} - w_j)$.

Now, remember that, by means of the algorithm, the support was built as $w_{i+1} = v_{i+1} = 1/h_i + \omega$ and $\alpha_{j+1} = L_{j+1} = 1/h_j + 1/h_0 + \omega$, so our expression becomes

$$\prod_{j=1}^{s}(1/h_{is+j^*-1}+1/h_0-1/h_{j-1})=\prod_{j=1}^{s}(1/h_{is-1}+1/h_0-1/h_{j-1})$$

or, without loss of generality, rearranging and since we are in characteristic 2,

$$\prod_{j=1}^{s}(1/h_0+1/h_{is+j^*}+1/h_j)=\prod_{j=1}^{s}(1/h_0+1/h_{is}+1/h_j).$$

Let $k_1 = is + j^*$ and $k_2 = is$; then, remembering equation (3.3), we can rewrite:

$$\prod_{j=1}^{s}(1/h_0+1/h_{k_1}+1/h_j)=\prod_{j=1}^{s}(1/h_0+1/h_{k_2}+1/h_j) \iff$$

$$\iff \prod_{j=1}^{s}(1/h_{k_1\oplus j})=\prod_{j=1}^{s}(1/h_{k_2\oplus j}) \iff \frac{1}{\prod_{j=1}^{s}h_{k_1\oplus j}}=\frac{1}{\prod_{j=1}^{s}h_{k_2\oplus j}} \iff$$

$$\iff \prod_{j=1}^{s}h_{k_1\oplus j}=\prod_{j=1}^{s}h_{k_2\oplus j},$$

which is true since k_1 and k_2 belong to the same block (the matrix is $s\times s$ dyadic). Essentially, this corresponds to multiplying together the elements of a string of length s (substring of a row) on two different rows of the same block; since each block is dyadic, any two rows are a permutation of each other, and the product of the elements is therefore constant. Hence the equality holds, and this terminates the proof. $\triangle$

Proposition 4.1 tells us that there are $n_0'-1$ distinct variables (since, like before, we can arbitrarily fix one of them). Now, the dimension of the blocks is smaller (as $s<\ell$), so we will have more equations, but the numbers are not increasing at the same rate. In fact $k/s=(n-mst)/s=(n_0's-mst)/s=s(n_0'-mt)/s=n_0'-mt$. We will then have the following values for the linear part of the system.

Table 4.2: System specifications for our scheme (linear part).

	GS-QD
Unknowns Y_i	$n_0'-1$
Linear equations involving only the Y_i	$n_0'-mt$

The solution space will therefore have dimension $mt-1$. This is a major improvement since now the security does not rely only on m; we can instead increase t so that we are not forced to use a big extension field, which gives large and unpractical keys, while making the attack less effective.

4.4.2 Parameters

In the following tables we give various sets of parameters in order to better illustrate the features of our scheme. The column “Size” refers to the public key size, expressed in bytes, while the column “ISD cost” refers to the estimated complexity of decoding attacks[1]($\log_2$ of binary operations). We also include experimental results about resistance to the attack just presented (column “FOPT cost”); these are obtained by using the upper bound provided by equation (3.7). We remark that the resulting numbers are just a theoretical upper bound that gives the approximate cost of computing a Gröbner basis with the indicated dimensions and variables, but nevertheless are useful to give an idea of the expected cost of the attack against that specific set of parameters. The numbers obtained by the theorem match with the costs obtained for the attacks successfully mounted against the codes of [11] and [85]. It also seems to emerge why the authors indicate 20 as a safe threshold, since all the parameters that produce a number of free variables greater than 20 generate a complexity superior to 2^{128}. Table 4.3 highlights the differences in performance and security according to the choice of m and t when keeping fixed the other parameters. Note that the first line ($t = 1$) represents a Goppa code.

Table 4.3: Example of parameters for GS codes over the base field $\mathbb{F}_{2^2}$, for a fixed number ($mt - 1 = 23$) of free variables.

m	n	k	s	t	Errors	Size (bytes)	ISD cost	FOPT cost
24	12288	6144	2^8	1	128	36864	128	150
12	6144	3072	2^7	2	128	18432	128	150
8	4096	2560	2^6	3	96	15360	128	160

Here we chose to keep constant this particular number of free variables mainly because $mt = 24$ gives a lot of possibilities for factoring (i.e. a lot of different choices for m and t) and the resulting amount 23 is well above the threshold of 20 indicated in [39].
It is also possible to observe that choosing an odd value for t gives better results even with a smaller number of errors introduced (e.g. compare line 2 and 3). That is because while the product st decreases (and consequently the numbers of correctable errors), so do the code minimum requirements for size (n) and dimension (k). This allows a tighter choice of parameters and overall works better for our purposes.

Table 4.4: GS codes over the base field $\mathbb{F}_{2^2}$ with fixed length $n = 1920$ and extension degree $m = 6$.

k	s	t	Errors	Size (bytes)	ISD cost	FOPT cost
960	2^5	5	80	7200	90	186
768	2^6	3	96	3456	80	105

[1]To compute this number we refer to [99] and use the corresponding script provided by Christiane Peters in http://www2.mat.dtu.dk/people/C.Peters/isdfq.html.

From Table 4.4 it is evident that a bigger t allows the construction of a code with better performance, but results in a much bigger key. It is also clear how deeply all the parameters are intertwined, at the same time contributing to the flexibility of the scheme: the first code, for instance, generates a much greater complexity against the structural attack, while achieving an even smaller key size than any of the codes in Table 4.3. However, the security against general decoding attacks decreases considerably.
Keeping all of this in mind, we give in Table 4.5 a sample of some smaller codes with the aim to minimize the public key size.

Table 4.5: Sets of parameters for smaller GS codes, obtained by choosing larger base fields and increasing t, while lowering the extension degree.

Base Field	m	n	k	s	t	Errors	Size (bytes)	ISD cost	FOPT cost
$\mathbb{F}_{2^5}$	2	992	416	2^5	9	144	4680	128	105
$\mathbb{F}_{2^4}$	3	768	432	2^4	7	56	4536	80	132
$\mathbb{F}_{2^5}$	2	512	256	2^4	2^3	64	2560	80	96

In an updated version of [85], the authors remove all the insecure parameters and keep only the set referring to binary quasi-dyadic codes. We present them again below for a comparison.

Table 4.6: Quasi-Dyadic Goppa codes ([85, Table 2]) with base field $\mathbb{F}_2$ and extension degree $m = 16$.

n	k	ℓ	Size (bytes)	ISD cost
8192	4096	256	8192	256
6912	2816	256	5632	192
4092	2048	128	4096	128
3584	1536	128	3072	112
2304	1280	64	2560	80

Note that we decided not to include the column "FOPT cost" in this case. This is because, as we argued in Remark 3.4, the FOPT algorithm doesn't, to date, lead to an attack against binary quasi-dyadic Goppa codes. We remark that it still makes sense considering a comparison, in the eventuality that some future work might succeed in completing the attack for the binary case.

For all the above codes, the level of security ($m - 1 = 15$) against FOPT is the same of the last code in Table 4.5, but only one has the same key size (2560 bytes), whereas the others are all considerably larger. If our main concern is resistance against structural attacks rather than general decoding attacks, it is then evident that we have an advantage.
An example is the codes in Table 4.5, line 1 and Table 4.6, line 3. For the same security level of 2^{128} we have a solution space of dimension $mt - 1 = 17$ for the former as opposed to 15 for the latter.

We remark that until a precise complexity analysis for the structural attacks is given, we should obey the condition obtained from the experimental results presented in [39], thus keeping the dimension of the solution space for the Y_i strictly greater than 20.

Remark 4.2 The special structure of the blocks $\hat{H}_i$ as described in (4.2) might suggest the possibility of a modification of the FOPT attack to exploit the extra structure coming from the powering process when $t > 1$. In particular, the coefficients z_i could be treated as an additional set of unknowns $\{Z_i\}$. This, however, would imply changing completely the algebraic system to solve, since the matrix given in (4.1) is **not** in alternant form. We remark that the FOPT attack is aimed generally at codes that are part of the alternant family, to the point that it could as well be directed against the original McEliece. This possibility is in fact mentioned by the authors in [38], and immediately discarded since solving the system in this case would prove infeasible. The success of FOPT depends on the additional structure coming from the quasi-cyclic or quasi-dyadic properties, rather than the properties of the code itself.
To date, such a modification hasn't been proposed.

4.5 Implementation

4.5.1 The Fujisaki-Okamoto Conversion

There are standard ways to obtain an IND-CCA2 secure encryption scheme from one that only has OW-CPA, for example the Fujisaki-Okamoto transform [46], introduced in 1999. The construction achieves CCA2-security in the random oracle model by integrating an asymmetric encryption scheme with a symmetric scheme, and therefore it is also known as *Hybrid Encryption.*
The IND-CPA security is obtained directly if the asymmetric scheme is One-Way secure and the symmetric scheme is Find-Guess secure. The IND-CCA2 security requires an additional property of the asymmetric encryption scheme called γ-uniformity. We define it here.

Definition 4.1 Let $\mathcal{E}$ be a PKE scheme as defined in Table 2.3 and let's call R the set where the randomness is chosen for the (probabilistic) encryption algorithm. For given $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{K}$, $\phi \in \mathsf{P}$ and a string $\boldsymbol{y}$, we define

$$\gamma(\phi, \boldsymbol{y}) = \mathsf{Pr}[r \overset{\$}{\leftarrow} \mathsf{R} : \boldsymbol{y} = \mathsf{Enc}_{\mathsf{pk}}(\phi, r)] \tag{4.3}$$

where the notation $\mathsf{Enc}_{\mathsf{pk}}(\phi, r)$ makes explicit the role of the randomness r. We say that $\mathcal{E}$ is γ-*uniform* if, for any $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{K}$, any $\phi \in \mathsf{P}$ and any string $\boldsymbol{y}$, we have $\gamma(\phi, \boldsymbol{y}) \leq \gamma$ for a certain $\gamma \in \mathbb{R}$.

Table 4.7: The Fujisaki-Okamoto conversion. $\mathcal{H}_1$ and $\mathcal{H}_2$ are hash functions.

Encryption of ϕ	Decryption of ψ
$\eta \xleftarrow{\$} \mathsf{P}^{\mathsf{PKE}}$	$\psi = (\psi_1 \| \psi_2)$
$r = \mathcal{H}_1(\eta, \phi)$	$\hat{\eta} = \mathsf{Dec}^{\mathsf{PKE}}_{\mathsf{sk}}(\psi_1)$ (return $\perp$ if decryption fails)
$\psi_1 = \mathsf{Enc}^{\mathsf{PKE}}_{\mathsf{pk}}(\eta, r)$	$\hat{\phi} = \mathsf{Dec}^{\mathsf{SE}}_{\mathcal{H}_2(\hat{\eta})}(\psi_2)$ (return $\perp$ if decryption fails)
$\psi_2 = \mathsf{Enc}^{\mathsf{SE}}_{\mathcal{H}_2(\eta)}(\phi)$	$\hat{r} = \mathcal{H}_1(\hat{\eta}, \hat{\phi})$
	if $\mathsf{Enc}^{\mathsf{PKE}}_{\mathsf{pk}}(\hat{\eta}, \hat{r}) = \psi_1$ return $\phi = \hat{\phi}$
return $\psi = (\psi_1 \| \psi_2)$	else return $\perp$

In a successive paper [64], Kobara and Imai proposed three alternative constructions in a similar fashion, tailored specifically for the McEliece cryptosystem rather than a general OWE encryption scheme. The biggest contribution of the new constructions is that the amount of overhead data (i.e. difference between the bit-length of the ciphertext and the bit-length of the plaintext) is considerably reduced. While this is certainly an important issue for some applications, in the common cryptographic practice it will never constitute a serious concern. In fact, the aim of public key cryptography is not to encrypt a whole, large plaintext, but rather to encrypt just a small (e.g. 128 or 256 bits) key for a more efficient symmetric scheme, that will be then used to encrypt the message. From a computational point of view the Kobara-Imai encryption process seems to be more expensive; in fact, the whole construction is rather complex.

Table 4.8: The Kobara-Imai hybrid "Conversion γ" for the McEliece (McE) public-key encryption scheme. $\mathcal{H}$ is a hash function, *Gen* a random number generator, *Conv* a constant weight encoding function and *Const* a (predetermined) public constant.

Encryption of ϕ	Decryption of ψ
$r \xleftarrow{\$} \{0,1\}^*$	$\psi = (\boldsymbol{y}_5 \| \boldsymbol{y}')$
$\boldsymbol{y}_1 = Gen(\boldsymbol{r}) \oplus (\phi \| Const)$	$\boldsymbol{y}_3 = \mathsf{Dec}^{\mathsf{McE}}_{\mathsf{sk}}(\boldsymbol{y}')$
$\boldsymbol{y}_2 = \boldsymbol{r} \oplus \mathcal{H}(\boldsymbol{y}_1)$	$\boldsymbol{z} = \boldsymbol{y}_3 G \oplus \boldsymbol{y}'$
$(\boldsymbol{y}_5 \| \boldsymbol{y}_4 \| \boldsymbol{y}_3) = (\boldsymbol{y}_2 \| \boldsymbol{y}_1)$	$\boldsymbol{y}_4 = Conv^{-1}(\boldsymbol{z})$
$\boldsymbol{z} = Conv(\boldsymbol{y}_4)$	$(\boldsymbol{y}_2 \| \boldsymbol{y}_1) = (\boldsymbol{y}_5 \| \boldsymbol{y}_4 \| \boldsymbol{y}_3)$
	$\boldsymbol{r} = \boldsymbol{y}_2 \oplus \mathcal{H}(\boldsymbol{y}_1)$
	$(\hat{x} \| Const') = \boldsymbol{y}_1 \oplus Gen(r)$
	if $Const' = Const$ return $\phi = \hat{\phi}$
return $\psi = (\boldsymbol{y}_5 \| \mathsf{Enc}^{\mathsf{McE}}_{G}(\boldsymbol{y}_3, \boldsymbol{z}))$	else return $\perp$

Note that the Fujisaki-Okamoto decryption process includes an encoding operation in the final check. This makes decryption slower. The cost of the process, though, is still dominated by the decoding operation rather than the matrix-vector multiplication. Moreover, as we already remarked, we argue that the distinctive feature of the McEliece scheme is the fast encryption process, and the Fujisaki-Okamoto conversion preserves fast encryption better than the Kobara-Imai approach.

4.5.2 Applying Fujisaki-Okamoto to McEliece

We give here a new way to use McEliece together with the Fujisaki-Okamoto transform. We remark that, although this work appears in [22], it is solely due to the author.
Previous approaches always needed a constant weight encoding function to convert $\mathcal{H}_1(\eta, \phi)$ into an error vector. Our idea is to swap the message and the error in the McEliece scheme, with a technique similar to the one used by Micciancio in [82]. This means that we interpret $\mathsf{Enc}_G^{\mathsf{McE}}(\boldsymbol{m}, \boldsymbol{e}) = \boldsymbol{e}G + \boldsymbol{m}$, encoding the message in the error vector rather than in the codeword. This is possible because, unlike other PKE schemes, the decryption process of McEliece, consisting mainly of decoding, returns both $\boldsymbol{m}$ and $\boldsymbol{e}$, allowing to recover, in addition to the plaintext, also the randomness used. With this simple trick, we avoid having to use a (inconvenient) constant weight encoding function and we simplify the encryption process considerably.
For simplicity we take the symmetric encryption scheme to be the one-time pad with an ephemeral key generated as $\mathcal{H}_2(\eta)$ where $\mathcal{H}_2$ is a random oracle with arbitrary length output. This symmetric encryption scheme satisfies the Find-Guess security property. In practice, one might use a block cipher in CBC mode.

Table 4.9: The Fujisaki-Okamoto transform applied to McEliece. $\mathbb{W}_{q,n,w}$, the set of words of length n and weight w over $\mathbb{F}_q$, is the usual space R for the McEliece PKE scheme.

Encryption of ϕ	Decryption of ψ
$\eta \xleftarrow{\$} \mathbb{W}_{q,n,w}$	$\psi = (\psi_1 \| \psi_2)$
$\boldsymbol{r} = \mathcal{H}_1(\eta \| \phi)$	$\hat{\eta} = \mathsf{Dec}_{\mathsf{sk}}^{\mathsf{McE}}(\psi_1)$ (return $\perp$ if decoding fails)
$\psi_1 = \boldsymbol{r}G + \eta$	$\hat{\phi} = \mathcal{H}_2(\hat{\eta}) \oplus \psi_2$
$\psi_2 = \mathcal{H}_2(\eta) \oplus \phi$	$\hat{\boldsymbol{r}} = \mathcal{H}_1(\hat{\eta} \| \hat{\phi})$
	if $\hat{\boldsymbol{r}}G + \hat{\eta} = \psi_1$ return $\phi = \hat{\phi}$
return $\psi = (\psi_1 \| \psi_2)$	else return $\perp$

The following lemma is fundamental to prove that our scheme enjoys the γ-uniformity required by the conversion.

Lemma 4.5 *The McEliece encryption scheme is γ-uniform for $\gamma = \dfrac{1}{q^k}$.*

Proof Let G be a public key for McEliece, that is a generator matrix for the code $\mathcal{C}$; in our setting, $\boldsymbol{y}$ is a generic string in $\mathbb{F}_q^n$. Then clearly:

$$\gamma(\eta, \boldsymbol{y}) = \mathsf{Pr}[\boldsymbol{r} \overset{\$}{\leftarrow} \mathbb{F}_q^k : \boldsymbol{y} = \boldsymbol{r}G + \eta] = \begin{cases} 0 & \text{if } \boldsymbol{y} - \eta \notin \mathcal{C} \\ \\ \dfrac{1}{q^k} & \text{if } \boldsymbol{y} - \eta \in \mathcal{C} \end{cases} \tag{4.4}$$

and that concludes the proof. △

Theorem 4.2 *If the assumptions of indistinguishability and decoding hardness of the McEliece PKE hold, the encryption scheme described in Table 4.9 is IND-CCA2 secure in the random oracle model.*

Proof The scheme enjoys one-way security because of the computational assumptions in the hypothesis. Moreover, Lemma 4.5 provides the γ-uniformity as required. Finally, the symmetric scheme used (one-time pad) satisfies the Find-Guess security property. It is then possible to apply [46, Th. 12]. △

4.5.3 Results

We now report on some implementation results, published in [22]. The implementation was initially done in C++ by my colleagues P.-L. Cayrel and G. Hoffmann, and is based on the library *SBCrypt* (Syndrome-Based Cryptography Library) by Barreto, Misoczki and Villas Boas. The code was subsequently converted to run on an embedded device, namely the microcontroller ATxmega256A3 from the AVR XMEGA family. It has 264 Kbytes of Flash memory, 16 Kbytes of SRAM memory and is running at a clock frequency of 32 MHz. The test results for the C++ code have been executed on an Intel(R) Core(TM) 2 Duo CPU E8400@3.00GHz running Ubuntu/Linux 2.6.32, where the source has been compiled with gcc 4.4.3. Similar results have been obtained using the Intel compiler icpc/icc. As for the embedded microcontroller, the code has been simulated on AVR Studio 5.0[2].

A key feature of our scheme is that we are able to make use of exponential/antilog tables to perform finite field arithmetic; these are simply tables containing the logarithmic representation of the elements of the finite field in question (see for example [75, Ch. 4, §5]). This is possible for all the codes in Table 4.5 as the extension fields are small enough to fit completely in the available memory, and it is therefore one of the main reasons to choose GS codes over Goppa codes.
As for the hash functions $\mathcal{H}_1$ and $\mathcal{H}_2$, we opted for the Keccak family[3] , recently selected by NIST as the winner of the SHA-3 competition, with assigned output length equal to k, in the first instance, or equal to the plaintext length (128 bits

[2] *www.atmel.com/avrstudio*
[3] *http://keccak.noekeon.org/.*

in our case), in the second. Its flexibility also allows for using it as a stream cipher. For details on how to use it for randomly sampling error vectors of weight w, we refer again to [22].

McEliece based on GS codes

We have measured two different operations: the encoding step $\boldsymbol{m}G + \boldsymbol{e}$ for $\boldsymbol{m} \in \mathbb{F}_q^k$ and the decoding of a ciphertext $\psi \in \mathbb{F}_q^n$. Results are presented below for three sets of parameters that we call, respectively, codes $\mathfrak{A}, \mathfrak{B}$ and $\mathfrak{C}$.

Table 4.10: Profiling results for McEliece using GS codes. The timings are expressed in milliseconds (ms).

Name	Base Field	m	n	k	s	t	Errors	Encoding	Decoding
$\mathfrak{A}$	$\mathbb{F}_{2^5}$	2	992	416	2^5	9	144	0.287	5.486
$\mathfrak{B}$	$\mathbb{F}_{2^4}$	3	768	432	2^4	7	56	0.179	1.578
$\mathfrak{C}$	$\mathbb{F}_{2^5}$	2	512	256	2^4	2^3	64	0.093	1.234

It is easy to see that the decoding process dominates the runtime.

The next table reports the results obtained when running the same operations on the microcontroller, for the last two codes. The costs displayed are in clock cycles; for a conversion to the standard time units, it is enough to keep in mind that the device runs at 32MHz, hence we have 32 million cycles per second.

Table 4.11: Details of the costs of encryption and decryption for codes $\mathfrak{B}$ and $\mathfrak{C}$.

Operation	**Code $\mathfrak{B}$**	**Code $\mathfrak{C}$**
Generate error vector $\boldsymbol{e}$	313,114	316,568
Load the plaintext $\boldsymbol{m}$	4,313	2,553
Encode $\boldsymbol{m}G$	3,418,292	1,603,854
Add $\boldsymbol{e}$	8,818	5,944
Encoding total	3,744,537	1,928,919

Operation	**Code $\mathfrak{B}$**	**Code $\mathfrak{C}$**
Compute syndrome $H\boldsymbol{e}^{\mathsf{T}}$	6,910,742	5,440,245
Solve key equation	955,597	1,192,400
Compute error positions	2,061,066	1,571,689
Compute error values	611,898	794,463
Correct the errors	8,641	5,121
Decoding total	10,547,944	9,003,918

CCA2-McEliece based on GS codes

We now consider the implementation of the full CCA2-secure variant using the Fujisaki-Okamoto transformation. The performances of the scheme are given in Table 4.12 and Table 4.13, respectively for the C++ code and for the microcontroller.

Table 4.12: Profiling results for CCA2-McEliece using GS codes.

Name	Base Field	m	n	k	s	t	Errors	Encryption	Decryption
$\mathfrak{A}$	$\mathbb{F}_{2^5}$	2	992	416	2^5	9	144	0.323	5.914
$\mathfrak{B}$	$\mathbb{F}_{2^4}$	3	768	432	2^4	7	56	0.213	1.814
$\mathfrak{C}$	$\mathbb{F}_{2^5}$	2	512	256	2^4	2^3	64	0.114	1.382

Table 4.13: Details of the costs of encryption and decryption for CCA2-McEliece.

Operation	**Code $\mathfrak{B}$**	**Code $\mathfrak{C}$**
Generate error vector η	322,109	321,812
Load the plaintext x	1,019	1,019
Hash $r = \mathcal{H}(\eta, x)$	282,285	281,497
Encode rG	3,426,700	1,591,031
Add η	1,103	1,314
Hash $\mathcal{K}(\eta)$	137,704	137,720
Pad $\mathcal{K}(\eta) \oplus x$	1,814	1,811
Encryption total	4,171,734	2,336,204

Operation	**Code $\mathfrak{B}$**	**Code $\mathfrak{C}$**
Compute syndrome $H\psi_1^\mathsf{T}$	7,029,985	5,425,696
Solve key equation	954,522	1,202,032
Compute error positions	2,031,514	1,561,946
Compute error values	611,944	794,524
Correct the errors	1,108	5,112
Hash $\mathcal{K}(\hat{\eta})$	147,822	144,768
Pad $\mathcal{K}(\hat{\eta}) \oplus \psi_2$	1,585	1,586
Hash $\hat{r} = \mathcal{H}(\hat{\eta}, \hat{x})$	282,066	282,278
Encode $\hat{r}G$	3,426,721	1,591,049
Add $\hat{\eta}$	1,113	1,273
Check equality	9,207	6,135
Decryption total	14,497,587	11,016,399

Comparing the results in Table 4.10 and Table 4.12 (as well as Table 4.11 and Table 4.13), we see that indeed the computational overhead to get CCA2 security is quite low.

For further clarification, the comparison of the total timings is reported in Tables 4.14 and 4.15.

Table 4.14: Summary of the timings (ms) for the C++ code.

Code	Encoding	CCA2 Encryption	Decoding	CCA2 Decryption
$\mathfrak{A}$	0.287	0.323	5.486	5.914
$\mathfrak{B}$	0.179	0.213	1.578	1.814
$\mathfrak{C}$	0.093	0.114	1.234	1.382

Table 4.15: Summary of the timings (clock cycles) for the embedded device.

Code	Encoding	CCA2 Encryption	Decoding	CCA2 Decryption
$\mathfrak{B}$	3,744,537	4,171,734	10,547,944	14,497,587
$\mathfrak{C}$	1,928,919	2,336,204	9,003,918	11,016,399

4.6 Conclusions

We have given a detailed description of a construction based on Quasi-Dyadic Generalized Srivastava codes. This is a generalization of [85], and is suitable as a trapdoor for a McEliece or Niederreiter scheme. The public keys are considerably smaller than the original McEliece proposal, and the construction easily gives codes secure against general decoding attacks.
Thanks to the introduction of the parameter t we are able to modulate our scheme in a much more flexible way, allowing us to consider codes over smaller extension fields without losing in security; moreover, the parameter t balances both the ratio (extension degree)/(number of free variables), and the reduction in the public key size, as this depends solely on s, which grows or shrinks according to t (for a fixed dimension and error-correction capacity). The result of this is a flexible and practical scheme which produces very small keys and resists all the attacks presented so far.
The choice of a base field other than $\mathbb{F}_2$, though actually increasing the public key size, looks like a better choice for the construction. Unlike the case of Goppa codes, GS codes do not benefit from an increased error-correction capacity in the binary case, so there is no particular reason to choose binary over non-binary. Instead, choosing a bigger base field allows us to further reduce the extension degree to values for which the scheme would otherwise be infeasible.

An independent work proposing a CCA2-secure scheme based on quasi-dyadic Goppa codes has been recently presented at PQCrypto 2011 by Stefan Heyse [58]. The performance indicated for encryption and decryption on the embedded device are slower than our results (the simulator program is the same, AVR Studio, although in a slightly older version). Part of the reason is due to the use of a constant weight encoding function (more than three times as costly as hashing) that we avoid thanks to the particular configuration of our scheme. However, the major difference comes from the fact that our vector-matrix multiplication, despite performing operations over non-binary fields, is at least two times faster, and this is the dominating part in the encryption process and is also a very high cost in the decryption process. This is a direct consequence of the structure of the scheme. In fact, the construction in [58] makes use of binary Goppa codes, which for security reasons [38] need to be defined over the extension field $\mathbb{F}_{2^{16}}$: this is too big to fit the corresponding log/antilog tables on the flash memory of the device. The result is that, in order to avoid using additional, external memory, the tables for $\mathbb{F}_{2^8}$ are represented instead, and operations are performed using tower field arithmetic, which is much slower. For example, a multiplication over a tower $\mathbb{F}_{(2^8)^2}$ is equivalent to performing 5 multiplications over $\mathbb{F}_{2^8}$.

Another disadvantage of [58] is that the public key G is computed as $S\hat{G}$ like in the original McEliece (P is supposed to be implicit in the support of the code), and the scramble matrix S occupies a great amount of memory (131,072 bytes, see [58, Table 3]). This is completely redundant, as the reduction to the systematic form is enough to mask the trapdoor and provide one-way security, as shown in [19].
On the other hand, the length of the encrypted plaintext is about 10 times the length of our plaintext (1288 bits, as opposed to 128 bits); however, we stress again that, in a "real-world" scenario, public-key encryption would only be used for encrypting a small amount of data. So if a large number of bits needs to be encrypted, then a PKE scheme would be used to exchange a small key (usually 128 or 256 bits) and then the plaintext would be encrypted with a symmetric encryption scheme.
If we follow this approach in our case, the timings that we obtain strongly support our claim. The latest benchmark speed indicated for AES-128 is about 16 cycles per byte[4]. Hence, if we want to encrypt, for a comparison, a plaintext of length 1288 bits = 161 bytes, it would take only 2,576 clock cycles; even on an embedded device, this number is very small compared to the rest of the encryption process. In total, our encryption process ranges from around 1.5 to 2.7 times faster than the scheme proposed in [58].

Table 4.16: Cost of encrypting a plaintext of length 1288 bits.

Code	**Cost (clock cycles)**
Goppa + Kobara-Imai	6,358,952
Code $\mathfrak{B}$	4,174,310
Code $\mathfrak{C}$	2,338,780

A similar argument holds for decryption.
Finally, we would like to highlight that we are using Keccak for both our hash functions and as a random number generator; the flexibility that it provides is evident. Other SHA-3 competitors like the function Blue Midnight Wish (BMW) used in [58] have been proved to be faster [44], but do not reach the same level of security, and for this have been discarded: although, as noted in the announcement of the finalists, "none of these candidates was clearly broken", several attacks have been presented[5].

[4]http://www.cryptopp.com/benchmarks.html
[5]http://ehash.iaik.tugraz.at/wiki/Blue_Midnight_Wish

Design of an Efficient Code-Based KEM

5.1 Introduction

A *Hybrid Encryption* scheme is a cryptographic protocol that features both a public-key encryption scheme and a symmetric encryption scheme, the former with the task of encrypting a key for the latter, in charge of encrypting the actual body of the message. The first component is therefore known as *Key Encapsulation Mechanism (KEM)* while the second is called *Data Encapsulation Mechanism (DEM)*. Key feature is that the two parts are independent of one another. The framework was first introduced in a seminal work by Cramer and Shoup [30], along with the corresponding notions of security and an example of a scheme based on the DDH assumptions. In a successive work [115], Shoup presents a proposal for an ISO standard on public-key encryption including many different schemes based on the RSA assumptions (RSA-OAEP, RSA-KEM), elliptic curves (ECIES) and Diffie-Hellman (PSEC, ACE). Other schemes based on integer factorization such as EPOC or HIME are also mentioned. This work follows up a suggestion from Bernstein [14] and stems from the RSA-KEM scheme (also known as "Simple RSA" in earlier versions of the paper) and as far as we know is the first proposal for a KEM based on the coding theory assumptions. The chapter is organized as follows: in the next section we introduce all the definitions and notions of security for KEMs and DEMs, plus other cryptographic tools that we will need for our scheme, such as KDFs and MACs. In Section 5.3 we present the construction, prove its security and give a hint on how to realize an efficient DEM to associate. Finally, we conclude in Section 5.4.

5.2 Preliminaries

5.2.1 Encapsulation Mechanisms and the Hybrid Framework

A key encapsulation mechanism is essentially a public-key encryption scheme, with the exception that the encryption algorithm takes no input apart from the public key, and returns a pair (K, ψ_0). The string K has fixed length ℓ_K, specified by the KEM, and ψ_0 is an "encryption" of K in the sense that $\mathsf{Dec}_{\mathsf{sk}}(\psi_0) = K$. Formally, a KEM consists of the following three algorithms.

Table 5.1: Key Encapsulation Mechanism.

KeyGen	A probabilistic key generation algorithm that takes as input a security parameter 1^λ and outputs a public key pk and a private key sk.
Enc	A probabilistic encryption algorithm that receives as input a public key pk and returns a key/ciphertext pair (K, ψ_0).
Dec	A deterministic decryption algorithm that receives as input a private key sk and a ciphertext ψ_0 and outputs either a key K or the failure symbol $\perp$.

A KEM is required to be *sound* for at least all but a negligible portion of public key/private key pairs, that is, if $\mathsf{Enc}_{\mathsf{pk}}(\ ) = (K, \psi_0)$ then $\mathsf{Dec}_{\mathsf{sk}}(\psi_0) = K$ with overwhelming probability.

The data encapsulation mechanism is a (possibly labeled) symmetric encryption scheme that uses as a key the string K output by the KEM. In what follows we only discuss, for simplicity, un-labeled DEMs.
Formally, a DEM consists of the following two algorithms.

Table 5.2: Data Encapsulation Mechanism.

Enc	A deterministic encryption algorithm that receives as input a key K and a plaintext ϕ and returns a ciphertext ψ_1.
Dec	A deterministic decryption algorithm that receives as input a key K and a ciphertext ψ_1 and outputs either a plaintext ϕ or the failure symbol $\perp$.

We require that the key K used in Enc and Dec has the same length ℓ_K as in the KEM. In this case, the mechanisms are said to be *compatible*, and can be composed in the canonical way as follows.

Table 5.3: Hybrid Encryption scheme.

K	$\mathsf{K}_{\mathsf{publ}}$ the *public key space.* $\mathsf{K}_{\mathsf{priv}}$ the *private key space.*
P	The set of messages to be encrypted, or *plaintext space.*
C	The set of the messages transmitted over the channel, or *ciphertext space.*
KeyGen	A probabilistic key generation algorithm[1]that takes as input a security parameter 1^λ and outputs a public key $\mathsf{pk} \in \mathsf{K}_{\mathsf{publ}}$ and a private key $\mathsf{sk} \in \mathsf{K}_{\mathsf{priv}}$.
Enc	A probabilistic encryption algorithm that receives as input a public key $\mathsf{pk} \in \mathsf{K}_{\mathsf{publ}}$ and a plaintext $\phi \in \mathsf{P}$. The algorithm invokes $\mathsf{Enc}_{\mathsf{pk}}^{\mathsf{KEM}}(\ )$ and obtains a key/ciphertext pair (K, ψ_0), then runs $\mathsf{Enc}_K^{\mathsf{DEM}}(\phi)$ and gets a ciphertext ψ_1. Finally, it outputs the ciphertext $\psi = (\psi_0 \| \psi_1)$.
Dec	A deterministic decryption algorithm that receives as input a private key $\mathsf{sk} \in \mathsf{K}_{\mathsf{priv}}$ and a ciphertext $\psi \in \mathsf{C}$. The algorithm parses ψ as $(\psi_0 \| \psi_1)$, then decrypts the left part by running $\mathsf{Dec}_{\mathsf{sk}}^{\mathsf{KEM}}(\psi_0)$; it either gets $\perp$ or a key K. In the first case, the algorithm returns $\perp$, otherwise it runs $\mathsf{Dec}_K^{\mathsf{DEM}}(\psi_1)$ and returns either the resulting plaintext ϕ or the failure symbol $\perp$.

The security notions are similar to their corresponding ones for PKE and SE schemes (see Section 2.1.3). We present them below.

Definition 5.1 The adaptive chosen ciphertext attack game for a KEM proceeds as follows:

1. Query a key generation oracle to obtain a public key pk.
2. Make a sequence of calls to a decryption oracle, submitting any string ψ_0 of the proper length[2]. The oracle will respond with $\mathsf{Dec}_{\mathsf{sk}}^{\mathsf{KEM}}(\psi_0)$.

[1]Note that this coincides with $\mathsf{KeyGen}^{\mathsf{KEM}}$.

[2]The adversary is free to choose this string in any arbitrary way, and not necessarily using the encryption algorithm.

3. Query an encryption oracle. The oracle runs $\mathsf{Enc}^{\mathsf{KEM}}_{\mathsf{pk}}$ to generate a pair $(\tilde{K}, \tilde{\psi_0})$, then chooses a random $b \in \{0,1\}$ and replies with the "challenge" ciphertext $(K^*, \tilde{\psi_0})$ where $K^* = \tilde{K}$ if $b = 1$ or K^* is a random string of length ℓ_K otherwise.

4. Keep performing decryption queries. If the submitted ciphertext is ψ_0^*, the oracle will return $\perp$.

5. Output $b^* \in \{0,1\}$.

We say that a KEM is secure if the advantage $\mathsf{Adv}_{\mathsf{KEM}}$ of any adversary $\mathcal{A}$ in the above CCA2 attack model is negligible.

Definition 5.2 The attack game for a DEM proceeds as follows:

1. Receive as input a key K.

2. Choose two plaintexts ϕ_0, ϕ_1 and submit them to an encryption oracle. The oracle will choose a random $b \in \{0,1\}$ and reply with the "challenge" ciphertext $\psi_1^* = \mathsf{Enc}^{\mathsf{DEM}}_K(\phi_b)$.

3. Make a sequence of calls to a decryption oracle, submitting any pair (L, ψ_1). The oracle will respond with $\mathsf{Dec}^{\mathsf{DEM}}_K(L, \psi_1)$. If the submitted ciphertext is (L^*, ψ_1^*), the oracle will return $\perp$.

4. Output $b^* \in \{0,1\}$.

We say that a DEM is secure if the advantage $\mathsf{Adv}_{\mathsf{DEM}}$ of any adversary $\mathcal{A}$ in the above attack model is negligible.

It is then easy to prove that, given an adversary $\mathcal{A}$ for the hybrid scheme (HY), there exist an adversary $\mathcal{A}_1$ for KEM and an adversary $\mathcal{A}_2$ for DEM running in roughly the same time as $\mathcal{A}$, such that for any choice of the security parameter λ we have $\mathsf{Adv}_{\mathsf{HY}}(\mathcal{A}, \lambda) \leq \mathsf{Adv}_{\mathsf{KEM}}(\mathcal{A}_1, \lambda) + \mathsf{Adv}_{\mathsf{DEM}}(\mathcal{A}_2, \lambda)$. See Cramer and Shoup [30, Th. 5] for a complete proof.

5.2.2 Other Cryptographic Tools

In this section we introduce other cryptographic tools that we need for our construction. We start with key derivation functions.

Definition 5.3 A *Key Derivation Function (KDF)* is a function that takes as input a string $\boldsymbol{x}$ of arbitrary length and an integer $\ell \geq 0$ and outputs a bit string of length ℓ.

A KDF is modelled as a random oracle, and it satisfies the *entropy smoothing* property, that is, if $\boldsymbol{x}$ is chosen at random from a high entropy distribution, the output of KDF should be computationally indistinguishable from a random length-ℓ bit string.
Intuitively, a good choice for a KDF could be a hash function with a variable (arbitrary) length output, such as Keccak (see previous chapter).

Definition 5.4 A *Message Authentication Code (MAC)* is an algorithm that produces a short piece of information (*tag*) used to authenticate a message. A MAC is defined by a function Ev that takes as input a key K and an arbitrary string T and returns a tag to be appended to the message, that is, a string τ of fixed length ℓ_M.

Informally, a MAC is similar to a signature scheme, with the difference that the scheme makes use of private keys both for evaluation and verification; in this sense, it could be seen as a "symmetric encryption equivalent" of a signature scheme.
The usual desired security requirement is existential unforgeability under chosen message attacks (see Section 2.1.4).

5.3 The Hybrid Encryption Scheme

5.3.1 The KEM Construction

The key encapsulation mechanism we present in this section follows the original Niederreiter approach (Table 3.2) and is thus based on the hardness of the Syndrome Decoding problem (Table 2.9). Note that, compared to the original Niederreiter scheme, a slight modification is introduced in the decryption process. As we will see later, this is necessary for the proof of security. Although unusual, this particular formulation still satisfies the requirements of a KEM.

Table 5.4: The Niederreiter KEM.

Setup	Fix public system parameters $q, n, r, w \in \mathbb{N}$.
KeyGen	Generate a random parity-check matrix $\hat{H}$ for an $[n, n-r]$ linear code over $\mathbb{F}_q$ with an efficient decoding algorithm given by the code description Γ, a $r \times r$ random invertible matrix S and an $n \times n$ permutation matrix P. Publish the public key $H = S\hat{H}P$ and store the private key (S, P, Γ).
Enc	On input a public key H choose a random $\boldsymbol{e} \in \mathbb{W}_{q,n,w}$, then compute $K = \mathsf{KDF}(\boldsymbol{e}, \ell_K)$, $\psi_0 = H\boldsymbol{e}^\mathsf{T}$ and return the key/ciphertext pair (K, ψ_0).
Dec	On input a private key (S, P, Γ) and a ciphertext ψ_0, first compute $\psi_0' = S^{-1}\psi_0$ then apply the decoding algorithm $\mathcal{D}_\Gamma$ to ψ_0'. If the decoding succeeds, multiply the output by P^{-1}, and recover $\boldsymbol{e}$, then compute $K = \mathsf{KDF}(\boldsymbol{e}, \ell_K)$ and return K. Otherwise, set K to be a string of length ℓ_K determined as a pseudorandom function[3] of ψ_0'. Return K.

If the ciphertext is correctly formed, the decoding will always succeed, hence the KEM is perfectly sound. Furthermore, we will see in Section 5.3.2 that, even if with this formulation $\mathsf{Dec}^{\mathsf{KEM}}$ never fails, there is no integrity loss in the context of the hybrid encryption scheme thanks to the integrity check given by the MAC.

We prove the security of the KEM in the following theorem.

[3]A natural suggestion is for example to set $K = \mathsf{KDF}(\psi_0', \ell_K)$.

Theorem 5.1 *Let $\mathcal{A}$ be an adversary in the random oracle model for the Niederreiter KEM as in Definition 5.1. Then there exists an adversary $\mathcal{A}'$ for SDP such that $\mathsf{Adv}_{\mathsf{KEM}}(\mathcal{A},\lambda) \leq \mathsf{Adv}_{\mathsf{SDP}}(\mathcal{A}',\lambda) + n_{\mathsf{Dec}}/N$, where n_{Dec} is the total number of decryption queries performed and $N = |\mathbb{W}_{q,n,w}|$. The running time of $\mathcal{A}'$ will be approximately equal to the running time of $\mathcal{A}$ plus n_{KDF} matrix-vector multiplications, where n_{KDF} is the number of random oracle queries performed, and some table lookups.*

Proof We replace KDF with a random oracle $\mathcal{H}$ mapping words in $\mathbb{W}_{q,n,w}$ to bit strings of length ℓ_K. To prove our claim, we proceed as follows. Let's call G_0 the original attack game played by $\mathcal{A}$, and S_0 the event that $\mathcal{A}$ succeeds in game G_0. We define a new game G_1 which is identical to G_0 except that the game is halted if the challenge ciphertext $\psi_0^* = \boldsymbol{H}\boldsymbol{e}^{*\mathsf{T}}$ obtained when querying the encryption oracle had been previously submitted to the decryption oracle: we call this event F_1. Since the number of valid ciphertexts is N, we have $\mathsf{Pr}[\mathsf{F}_1] \leq n_{\mathsf{Dec}}/N$. It follows that $\Big|\mathsf{Pr}[\mathsf{S}_0] - \mathsf{Pr}[\mathsf{S}_1]\Big| \leq n_{\mathsf{Dec}}/N$, where S_1 is the event that $\mathcal{A}$ succeeds in game G_1. Next, we define game G_2 which is identical to G_1 except that we generate the challenge ciphertext ψ_0^* at the beginning of the game, and we halt if $\mathcal{A}$ ever queries $\mathcal{H}$ at $\boldsymbol{e}^*$: we call this event F_2. By construction, since $\mathcal{H}(\boldsymbol{e}^*)$ is undefined, it is not possible to tell whether $K^* = K$, thus we have $\mathsf{Pr}[\mathsf{S}_2] = 1/2$, where S_2 is the event that $\mathcal{A}$ succeeds in game G_2. We obtain that $\Big|\mathsf{Pr}[\mathsf{S}_1] - \mathsf{Pr}[\mathsf{S}_2]\Big| \leq \mathsf{Pr}[\mathsf{F}_2]$ and we just need to bound $\mathsf{Pr}[\mathsf{F}_2]$.
We now construct an adversary $\mathcal{A}'$ against SDP. $\mathcal{A}'$ interacts with $\mathcal{A}$ and is able to simulate the random oracle and the decryption oracle with the help of two tables T_1 and T_2, initially empty, as described below.

Key Generation: On input the instance $(H, \boldsymbol{s}^*, w)$ of SDP, return $\mathsf{pk} = H$.

Challenge queries: When asked for the challenge ciphertext:

1. Generate a random string K^* of length ℓ_K.
2. Set $\psi_0^* = \boldsymbol{s}^*$.
3. Return the pair (K^*, ψ_0^*).

Random oracle queries: On input $\boldsymbol{e} \in \mathbb{W}_{q,n,w}$ to the random oracle:

1. Look up $\boldsymbol{e}$ in T_1. If $(\boldsymbol{e}, \boldsymbol{s}, K)$ is in T_1 for some $\boldsymbol{s}$ and K, return K.
2. Compute $\boldsymbol{s} = \boldsymbol{H}\boldsymbol{e}^{\mathsf{T}}$.
3. If $\boldsymbol{s} = \boldsymbol{s}^*$ then $\mathcal{A}'$ outputs $\boldsymbol{e}$ and the game ends.
4. Look up $\boldsymbol{s}$ in T_2. If $(\boldsymbol{s}, K)$ is in T_2 for some K (i.e. the decryption oracle has been evaluated at $\boldsymbol{s}$), return K.
5. Set K to be a random string of length ℓ_K and place the triple $(\boldsymbol{e}, \boldsymbol{s}, K)$ in table T_1.
6. Return K.

Decryption queries: Upon a decryption query $\psi_0 = \boldsymbol{s} \in \mathbb{F}_q^r$:

1. Look up $\boldsymbol{s}$ in T_2. If $(\boldsymbol{s}, K)$ is in T_2 for some K, return K.
2. Look up $\boldsymbol{s}$ in T_1. If $(\boldsymbol{e}, \boldsymbol{s}, K)$ is in T_1 for some $\boldsymbol{e}$ and K (i.e. the random oracle has been evaluated at $\boldsymbol{e}$ such that $\boldsymbol{s} = H\boldsymbol{e}^{\mathsf{T}}$), return K.
3. Generate a random string K of length ℓ_K and place the pair $(\boldsymbol{s}, K)$ in T_2.
4. Return K.

Note that, in both random oracle and decryption queries, we added Step 1 to guarantee the integrity of the simulation, that is, if the same value is queried more than once, the same output is returned.

A fundamental issue is that it is impossible for the simulator to determine if a word is decodable or not. If the decryption algorithm returned $\perp$ if and only if a word was not decodable, then it would be impossible to simulate decryption properly. We have resolved this problem by insisting that the KEM decryption algorithm always outputs a hash value. With this formulation, the simulation is flawless and $\mathcal{A}'$ outputs a solution to the SDP instance with probability equal to $\mathsf{Pr}[\mathsf{F}_2]$. $\triangle$

5.3.2 A Standard DEM

For completeness, we show how to construct a DEM in a standard way by means of a SE scheme and a one-time MAC.

Table 5.5: Standard DEM.

Enc	On input a key K and a plaintext ϕ, parse K as $(K_1\|\|K_2)$ then compute $\psi' = \mathsf{Enc}^{\mathsf{SE}}_{K_1}(\phi)$, set $T = \psi'$ and evaluate $\tau = \mathsf{Ev}(K_2, T)$. Return the ciphertext $\psi_1 = (\psi'\|\|\tau)$.
Dec	On input a key K and a ciphertext ψ_1, parse[4] ψ_1 as $(\psi'\|\|\tau)$ then parse K as $(K_1\|\|K_2)$, set $T = \psi'$ and apply the MAC algorithm to obtain $\tau' = \mathsf{Ev}(K_2, T)$. If $\tau' \neq \tau$ the verification fails, hence return $\perp$. Otherwise, compute $\phi = \mathsf{Dec}^{\mathsf{SE}}_{K_1}(\psi')$ and return the plaintext ϕ.

It is easy to prove that if the underlying components are secure, so is the resulting DEM. In particular it is possible to prove [30, Th. 4] that, for any DEM adversary $\mathcal{A}$, we have $\mathsf{Adv}_{\mathsf{DEM}}(\mathcal{A}, \lambda) \leq \mathsf{Adv}_{\mathsf{FG}}(\mathcal{A}_1, \lambda) + \mathsf{Adv}_{\mathsf{MAC}}(\mathcal{A}_2, \lambda)$, where $\mathcal{A}_1$ and $\mathcal{A}_2$ are, respectively, a find-guess adversary for SE and a one-time existential forgery adversary for MAC, both running in about the same time of $\mathcal{A}$.

[4]Note that this step may fail if, for example, ψ_1 is too short.

5.4 Conclusions and Future Work

We have introduced a key encapsulation method based on the Niederreiter cryptosystem. This is the first KEM based directly on a coding theory problem and it enjoys a simple construction and a tight security proof. Future work includes investigating practical applications of the KEM, with the aim of an implementation. This could potentially make use of an algebraic variant aimed to reduce the public key size, in a similar way as described in the previous chapter. The implementation work is still in progress at the current time and we chose therefore to not include it in this thesis.

On a CCA2-secure Variant of McEliece in the Standard Model

6.1 Introduction

As we saw in the previous chapters, it is possible to produce CCA2-secure code-based schemes in the random oracle model, but it is of interest to study systems that are secure in the standard model.
Rosen and Segev in [106] gave a general approach for CCA2 security in the standard model incorporating tools such as lossy trapdoor functions (a very powerful tool introduced by Peikert and Waters in [97]) and one-time signature schemes. This general protocol can be applied directly to many different hard problems such as Quadratic Residuosity, Composite Residuosity, the d-linear Assumption and the Syndrome Decoding Problem, as shown in [45]. Dowsley et al. [36] have tried to apply the Rosen-Segev approach to the McEliece framework. To do this, a new structure called k-repetition PKE is introduced, as well as a number of differences in the key generation, encryption and decryption processes. It is claimed that the scheme has IND-CCA2 security in the standard model.
In this chapter we make some observations on the ambiguity of the description of the scheme of [36], provide a correct formulation and proof of security, and then show how to get a CCA2-secure cryptosystem based on the McEliece assumptions using the original Rosen-Segev approach.
The chapter is structured as follows: in the next section, we recall the original Rosen-Segev scheme. Section 6.3 features two existing proposals for a scheme based on coding theory: the first makes use of the Niederreiter cryptosystem [88], while the second is a summary of [36]. In Section 6.4 we propose an alternative scheme to realize the Rosen-Segev protocol with McEliece. We conclude in Section 6.5.

6.2 The Rosen-Segev Scheme

6.2.1 Computable Functions and Correlated Products

We define here the notion of security under correlated products for a collection of functions. Recall from Definition 2.1 the notion of a collection of efficiently computable functions. We define a k-wise product as follows:

Definition 6.1 Let $\mathcal{F} = (\mathsf{G},\mathsf{F})$ be a collection of efficiently computable functions and k be an integer. The *k-wise product* $\mathcal{F}_\mathsf{k}$ is a pair of algorithms $(\mathsf{G}_\mathsf{k}, \mathsf{F}_\mathsf{k})$ such that:

- G_k is a generation algorithm that independently samples k functions from $\mathcal{F}$ by invoking k times the algorithm G and returns a tuple $(f_1, \dots, f_\mathsf{k})$.

- F_k is an evaluation algorithm that receives as input a sequence of functions $(f_1, \dots, f_\mathsf{k})$ and a sequence of points $(x_1, \dots, x_\mathsf{k})$ and invokes F to evaluate each function on the corresponding point, i.e.

$$\mathsf{F}_\mathsf{k}(f_1, \dots, f_\mathsf{k}, x_1, \dots, x_\mathsf{k}) = (\mathsf{F}(f_1, x_1), \dots, \mathsf{F}(f_\mathsf{k}, x_\mathsf{k})). \qquad (6.1)$$

Let's now also recall the definition of trapdoor one-way function (Definition 2.2). We may think to extend the notion to the case where the input is given according to a certain distribution, that is, there exists a correlation between the points $x_1, \dots, x_\mathsf{k}$.

Definition 6.2 Let $\mathcal{F} = (\mathsf{G}, \mathsf{F})$ be a collection of efficiently computable functions with domain D and $\mathcal{C}_\mathsf{k}$ be a distribution of points in $D_1 \times \cdots \times D_\mathsf{k}$. We say that $\mathcal{F}$ is *secure under a $\mathcal{C}_\mathsf{k}$-correlated product* if $\mathcal{F}_\mathsf{k}$ is one-way with respect to the input distribution $\mathcal{C}_\mathsf{k}$.

In the special case where the input distribution $\mathcal{C}_\mathsf{k}$ is exactly the uniform k-repetition distribution (that is, k copies of the same input $x \in D$) we simply speak about *one-wayness under k-correlated inputs*. Rosen and Segev in [106] showed that a collection of lossy trapdoor functions for an appropriate choice of parameters can be used to construct a collection of functions that is one-way under k-correlated inputs. Their work is summarized in the next section.

6.2.2 The Rosen-Segev Encryption Scheme

The computational assumption underlying the scheme is that there exists a collection of functions $\mathcal{F} = (\mathsf{G}, \mathsf{F})$ which is secure under k-correlated inputs. The scheme makes use of a strongly-unforgeable signature scheme and of a hard-core predicate h for the collection $\mathcal{F}_\mathsf{k}$.

$\mathsf{KeyGen}^{\mathsf{RS}}$: Invoke G for $2\mathsf{k}$ times independently and obtain the descriptions of functions $(f_1^0, f_1^1, \dots, f_\mathsf{k}^0, f_\mathsf{k}^1)$ and the corresponding trapdoors $(\mathsf{td}_1^0, \mathsf{td}_1^1, \dots, \mathsf{td}_\mathsf{k}^0, \mathsf{td}_\mathsf{k}^1)$. The former is distributed as the public key pk, while the latter is the private key sk.

$\mathsf{Enc}^{\mathsf{RS}}$: To encrypt a plaintext $m \in \{0,1\}$ with the public key pk, sample a key from a strongly-unforgeable one-time signature scheme, say $(\mathsf{vk}, \mathsf{sgk})$ and a random $x \in \{0,1\}^N$. Write vk_i for the i-th bit of vk and let h be a hard-core predicate, then:

1. Evaluate $c_i = \mathsf{F}(f_i^{\mathsf{vk}_i}, x)$ for $i = 1, \dots, \mathsf{k}$.
2. Set $y = m \oplus h(f_1^{\mathsf{vk}_1}, \dots, f_\mathsf{k}^{\mathsf{vk}_\mathsf{k}}, x)$.
3. Compute $\sigma = \mathsf{Sign}_{\mathsf{sgk}}^{\mathsf{SS}}(c_1, \dots, c_\mathsf{k}, y)$.

It is assumed that $\mathsf{vk} \in \{0,1\}^\mathsf{k}$: if not, it is enough to apply a universal one-way hash function to obtain the desired length.
Finally, output the ciphertext $\psi = (\mathsf{vk}, c_1, \dots, c_\mathsf{k}, y, \sigma)$.

$\mathsf{Dec}^{\mathsf{RS}}$: Upon reception of a ciphertext ψ:

1. Verify the signature; if $\mathsf{Ver}_{\mathsf{vk}}^{\mathsf{SS}}((c_1, \dots, c_\mathsf{k}, y), \sigma) = 0$ output $\perp$.
2. Otherwise compute $x_i = \mathsf{F}^{-1}(\mathsf{td}_i^{\mathsf{vk}_i}, c_i)$ for $i = 1, \dots, \mathsf{k}$.
3. If $x_1 = \cdots = x_\mathsf{k}$ then set $m = y \oplus h(f_1^{\mathsf{vk}_1}, \dots, f_\mathsf{k}^{\mathsf{vk}_\mathsf{k}}, x_1)$ and return the plaintext m, otherwise output $\perp$.

The security of the scheme comes from the next theorem, proved in [106].

Theorem 6.1 *Assuming that $\mathcal{F}$ is secure under* k*-correlated inputs, and that the signature scheme is one-time strongly unforgeable, the above encryption scheme is IND-CCA2-secure.*

The proof consists of a standard argument, divided in two parts. The first part shows that if an adversary exists that can to break the CCA2 security of the scheme, it can be converted to an adversary able to forge the signature scheme. In the second part, assuming that the forgery doesn't occur, an adversary is built that contradicts the security of the hard-core predicate. We don't present the proof here, but we refer the reader to [106] for more details.

6.3 Two Previous Proposals

If we describe the McEliece encryption as a function $f_G(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}G + \boldsymbol{y}$ then clearly this is not secure under correlated inputs: in fact, given two evaluations $f_{G_1}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}G_1 + \boldsymbol{y}$ and $f_{G_2}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}G_2 + \boldsymbol{y}$ we could sum the outputs together and, since the error vector cancels out (we assume we are in the binary case like in the original McEliece scheme), we get $\boldsymbol{x}(G_1 + G_2)$ from which it is easy to recover x. The problem is that, since we are defining a function, there is no randomness anymore, whereas McEliece requires a random error vector in order to be secure under k-correlated inputs. A mapping that incorporates a random element would in fact give a different result for multiple encryptions of the same plaintext and so won't have a unique image.

We now present two alternative schemes that have been proposed to deal with the matter.

6.3.1 Syndrome Decoding

This construction was presented by Freeman, Goldreich, Kiltz, Rosen and Segev [45] and is based on the Niederreiter cryptosystem (Table 3.2). The Niederreiter trapdoor function can be efficiently described in the above fashion as the family $\mathcal{N} = (\mathsf{G},\mathsf{F})$ where G and F are defined as follows:

Generation: On input n, k the algorithm G generates a random parity-check matrix $\hat{H}$ for an $[n, k]$ linear code over $\mathbb{F}_q$ with an efficient decoding algorithm given by the code description $\mathit{\Gamma}$, an $(n-k) \times (n-k)$ random invertible matrix S and an $n \times n$ permutation matrix P, then publishes the public key $H = S\hat{H}P$ and the private key $(S, P, \mathit{\Gamma})$.

Evaluation: On input $H, \boldsymbol{e}$, where $\boldsymbol{e}$ is a string of fixed weight w in $\mathbb{F}_q^n$, the algorithm F computes $\psi = H\boldsymbol{e}^{\mathsf{T}}$ and returns the ciphertext ψ.

It is possible to invert F using the trapdoor: on input $(S, P, \mathit{\Gamma})$ and ψ, multiply ψ by S^{-1}, decode to obtain $P\boldsymbol{e}^{\mathsf{T}}$ and retrieve $\boldsymbol{e}^{\mathsf{T}}$ by multiplying by P^{-1}.

The function is proved to be one-way under k-correlated inputs in [45, Th. 6.2], provided that k is chosen such that the indistinguishability and decoding hardness assumptions still hold for n and $(n-k)\mathsf{k}$; it is intended to be used in the general Rosen-Segev framework.

6.3.2 k-repetition PKE

Dowsley, Müller-Quade and Nascimento [36] propose a scheme that resembles the Rosen-Segev protocol trying to apply it to the McEliece cryptosystem. Despite the authors claim that this is the "direct translation" of [106], clearly this is not the case. Among other differences, the scheme doesn't rely on a collection of functions but instead defines a structure called *k-repetition Public-Key Encryption* (PKE_{k}). This is essentially an application of k samples of the PKE to the same input, in which the decryption algorithm also includes a verification step on the k outputs. The encryption step produces a signature directly on the McEliece ciphertexts instead of introducing a random vector x as in the original scheme; therefore an IND-CPA secure variant of McEliece's cryptosystem (Nojima et al. [89]) is necessary to achieve CCA2 security. We briefly recall it below.

Table 6.1: The Randomized McEliece cryptosystem.

Setup	Fix public system parameters $q, m, n, k, w \in \mathbb{N}$ such that $k \geq n - wm$, $k = k_1 + k_2$.
K	$\mathsf{K}_{\mathsf{publ}}$ the set of $k \times n$ matrices over $\mathbb{F}_q$. $\mathsf{K}_{\mathsf{priv}}$ the set of triples formed by a $k \times k$ invertible matrix over $\mathbb{F}_q$, an $n \times n$ permutation matrix over $\mathbb{F}_q$ and a code description.
P	The vector space $\mathbb{F}_q^{k_1}$.
R	The vector space $\mathbb{F}_q^{k_2}$.
C	The vector space $\mathbb{F}_q^{n}$.
KeyGen	Generate at random a polynomial $g \in \mathbb{F}_{q^m}[x]$ and elements $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_{q^m}$, then build the Goppa code $\Gamma = \Gamma(\alpha_1, \ldots, \alpha_n, g)$ over $\mathbb{F}_q$ and its generator matrix $\hat{G}$. Select at random a $k \times k$ invertible matrix S and an $n \times n$ permutation matrix P. Publish the public key $G = S\hat{G}P \in \mathsf{K}_{\mathsf{publ}}$ and store the private key $(S, P, \Gamma) \in \mathsf{K}_{\mathsf{priv}}$.
Enc	On input a public key $G \in \mathsf{K}_{\mathsf{publ}}$, a plaintext $\boldsymbol{m} \in \mathsf{P}$ and a randomness $\boldsymbol{r} \in \mathsf{P}$, sample a random error vector $\boldsymbol{e}$ of weight w in $\mathbb{F}_q^n$ and return the ciphertext $\psi = (\boldsymbol{r}\|\boldsymbol{m})G + \boldsymbol{e} \in \mathsf{C}$.
Dec	On input the private key $(S, P, \Gamma) \in \mathsf{K}_{\mathsf{priv}}$ and a ciphertext $\psi \in \mathsf{C}$, first compute ψP^{-1} then apply the decoding algorithm D_Γ to it. If the decoding succeeds, multiply the output by S^{-1}, parse it as $(\boldsymbol{r}\|\boldsymbol{m})$ and return the plaintext $\phi = \boldsymbol{m}$. Otherwise, output $\perp$.

We now present the scheme described in [36]. Note that, in the paper, this is presented as a general scheme, applicable to any IND-CPA secure PKE which is secure and verifiable under k-correlated inputs.

$\mathsf{KeyGen}^{\mathsf{DMQN}}$: Invoke $\mathsf{KeyGen}^{\mathsf{PKE}}$ for $2\mathsf{k}$ times independently and obtain the collection of public keys $(\mathsf{pk}_1^0, \mathsf{pk}_1^1, \ldots, \mathsf{pk}_\mathsf{k}^0, \mathsf{pk}_\mathsf{k}^1)$ and the corresponding private keys $(\mathsf{sk}_1^0, \mathsf{sk}_1^1, \ldots, \mathsf{sk}_\mathsf{k}^0, \mathsf{sk}_\mathsf{k}^1)$, then run the key generation algorithm for the signature scheme to obtain a key $(\mathsf{vk}^*, \mathsf{sgk}^*)$.
Publish the public key $\mathsf{pk} = (\mathsf{pk}_1^0, \mathsf{pk}_1^1, \ldots, \mathsf{pk}_\mathsf{k}^0, \mathsf{pk}_\mathsf{k}^1)$ and choose the private key accordingly to vk^*, i.e. $\mathsf{sk} = (\mathsf{vk}^*, \mathsf{sk}_1^{1-\mathsf{vk}_1^*}, \ldots, \mathsf{sk}_\mathsf{k}^{1-\mathsf{vk}_\mathsf{k}^*})$.

$\mathsf{Enc}^{\mathsf{DMQN}}$: To encrypt a plaintext m with the public key pk, sample another, different key $(\mathsf{vk}, \mathsf{sgk})$ from the signature scheme, then:

1. Evaluate $c_i = \mathsf{Enc}^{\mathsf{PKE}}_{\mathsf{pk}_i^{\mathsf{vk}_i}}(m)$ for $i = 1, \ldots, \mathsf{k}$.

2. Compute $\sigma = \mathsf{Sign}^{\mathsf{SS}}_{\mathsf{sgk}}(c_1, \ldots, c_\mathsf{k})$.

3. Output the ciphertext $\psi = (\mathsf{vk}, c_1, \ldots, c_\mathsf{k}, \sigma)$.

$\mathsf{Dec}^{\mathsf{DMQN}}$: Upon reception of a ciphertext ψ:

1. If $\mathsf{vk} = \mathsf{vk}^*$ or $\mathsf{Ver}^{\mathsf{SS}}_{\mathsf{vk}}((c_1, \ldots, c_\mathsf{k}), \sigma) = 0$ output $\perp$.

2. Otherwise compute $m = \mathsf{Dec}^{\mathsf{PKE}}_{\mathsf{sk}_i^{\mathsf{vk}_i}}(c_i)$ for some i such that $\mathsf{vk}_i \neq \mathsf{vk}_i^*$.

3. Verify that c_i is a possible output of $\mathsf{Enc}^{\mathsf{PKE}}_{\mathsf{pk}_i^{\mathsf{vk}_i}}(m)$ for all $i = 1, \ldots, t$. If the verification is successful return the plaintext m, otherwise output $\perp$.

Since we know that $\mathsf{vk} \neq \mathsf{vk}^*$, there is at least one position in which they differ, hence the decryption process is well defined.

Remark 6.1 Note that, even though the encryption process is not deterministic, for McEliece encryption it is still possible to perform the check in the last step of $\mathsf{Dec}^{\mathsf{DMQN}}$. It is in fact enough to check the Hamming weight of $c_i - \boldsymbol{m}G_i$ where G_i is the generator matrix corresponding to the public key $\mathsf{pk}_i^{\mathsf{vk}_i}$. This is not clearly stated by the authors along with the description of the general scheme, but it is mentioned later on in [36, Theorem 3] for the particular case of the randomized McEliece.

Remark 6.2 Clearly, the above specification of the scheme is ambiguous. In fact, even assuming that the underlying encryption scheme is IND-CPA secure, the encryption step is described simply as $\mathsf{Enc}^{\mathsf{PKE}}_{\mathsf{pk}_i^{\mathsf{vk}_i}}(m)$ for $i = 1, \ldots, \mathsf{k}$, without indicating explicitly the role of the randomness. In [36, Section 4] some remarks are made about the security and it is suggested to use the randomized McEliece scheme from [89] (see Table 6.1); however, precise details on how this should be instantiated are missing. One could in general think at the k encryptions as $c_i = \mathsf{Enc}^{\mathsf{PKE}}_{\mathsf{pk}_i^{\mathsf{vk}_i}}(\boldsymbol{m}, \boldsymbol{r}_i) = (\boldsymbol{r}_i || \boldsymbol{m})G_i + \boldsymbol{e}_i$. In this case, since we check the Hamming weight of $c_i - (\boldsymbol{r}_i || \boldsymbol{m})G_i$, the check would obviously fail unless $\boldsymbol{r}_1 = \cdots = \boldsymbol{r}_\mathsf{k} = \boldsymbol{r}$.

Remark 6.3 The KeyGen algorithm is slightly different from the Rosen-Segev case. In particular, $2\mathsf{k}$ keys are generated, then a random verification key vk^* is chosen and half of the private keys (the ones corresponding to vk^*) are discarded. This also implies that decryption only works when $\mathsf{vk} \neq \mathsf{vk}^*$. This technique is used in the context of the proof of Theorem 6.1, specifically in the second part while constructing an efficient distinguisher for the hard-core predicate. While, as we will see in the following, this is necessary for the proof (both for the original paper and for the proposed scheme), it is certainly a redundant requirement in the KeyGen process.

In light of the previous observations, a more correct description of the scheme would then be:

$\mathsf{KeyGen}^{\mathsf{DMQN}}$: Invoke $\mathsf{KeyGen}^{\mathsf{PKE}}$ for $2\mathsf{k}$ times independently and obtain the collection of public keys $(\mathsf{pk}_1^0, \mathsf{pk}_1^1, \dots, \mathsf{pk}_\mathsf{k}^0, \mathsf{pk}_\mathsf{k}^1)$ and the corresponding private keys $(\mathsf{sk}_1^0, \mathsf{sk}_1^1, \dots, \mathsf{sk}_\mathsf{k}^0, \mathsf{sk}_\mathsf{k}^1)$. The former is distributed as the public key pk, while the latter is the private key sk.

$\mathsf{Enc}^{\mathsf{DMQN}}$: To encrypt a plaintext m with the public key pk, sample a key $(\mathsf{vk}, \mathsf{sgk})$ from the signature scheme *and a randomness* r, then:

1. Evaluate $c_i = \mathsf{Enc}^{\mathsf{PKE}}_{\mathsf{pk}_i^{\mathsf{vk}_i}}(m, r)^1$ for $i = 1, \dots, \mathsf{k}$.
2. Compute $\sigma = \mathsf{Sign}^{\mathsf{SS}}_{\mathsf{sgk}}(c_1, \dots, c_\mathsf{k})$.
3. Output the ciphertext $\psi = (\mathsf{vk}, c_1, \dots, c_\mathsf{k}, \sigma)$.

$\mathsf{Dec}^{\mathsf{DMQN}}$: Upon reception of a ciphertext ψ:

1. If $\mathsf{Ver}^{\mathsf{SS}}_{\mathsf{vk}}((c_1, \dots, c_\mathsf{k}), \sigma) = 0$ output $\perp$.
2. Otherwise compute $(m, r) = \mathsf{Dec}^{\mathsf{PKE}}_{\mathsf{sk}_i^{\mathsf{vk}_i}}(c_i)$ for some i.
3. Verify that c_i is a possible output of $\mathsf{Enc}^{\mathsf{PKE}}_{\mathsf{pk}_i^{\mathsf{vk}_i}}(m, r)$ for all $i = 1, \dots, t$. If the verification is successful return the plaintext m, otherwise output $\perp$.

The construction is proved to be CCA2-secure in [36, Th. 1]. We now reproduce a more careful proof of security.

Theorem 6.2 ([36]) *Assuming that PKE_k is IND-CPA secure and verifiable under k-correlated inputs, and that the signature scheme is one-time strongly unforgeable, the above encryption scheme is IND-CCA2-secure.*

Let $\mathcal{A}$ be an IND-CCA2 adversary. During the attack game, $\mathcal{A}$ submits m_0, m_1 and gets back the challenge ciphertext $\psi^* = (\mathsf{vk}^*, c_1^*, \dots, c_\mathsf{k}^*, \sigma^*)$. Indicate with Forge the event that, for one of $\mathcal{A}$'s decryption queries $\psi = (\mathsf{vk}, c_1, \dots, c_\mathsf{k}, \sigma)$, it holds $\mathsf{vk} = \mathsf{vk}^*$ and $\mathsf{Ver}^{\mathsf{SS}}_{\mathsf{vk}}((c_1, \dots, c_\mathsf{k}), \sigma) = 1$. The theorem is proved by means of the two following lemmas.

[1]Note that the randomness we are expliciting here is the one necessary to realize the IND-CPA security of PKE, hence Enc is still a randomized algorithm. In particular, for the McEliece instantiation we would have $c_i = (\boldsymbol{r}||\boldsymbol{m})G_i + \boldsymbol{e}_i$.

Lemma 6.1 *Pr[Forge] is negligible.*

Proof Assume that there exists an adversary $\mathcal{A}$ for which $\mathsf{Pr}[\mathsf{Forge}]$ is not negligible. We build an adversary $\mathcal{A}'$ that breaks the security of the one-time strongly unforgeable scheme. $\mathcal{A}'$ works as follows:

Key Generation: Invoke $\mathsf{KeyGen}^{\mathsf{DMQN}}$ as above and return pk to $\mathcal{A}$.

Decryption queries: Upon a decryption query $\psi = (\mathsf{vk}, c_1, \dots, c_{\mathsf{k}}, \sigma)$:

1. If $\mathsf{vk} = \mathsf{vk}^*$ and $\mathsf{Ver}^{\mathsf{SS}}_{\mathsf{vk}}((c_1, \dots, c_{\mathsf{k}}), \sigma) = 1$ output $\perp$ and halt.
2. Otherwise, decrypt using $\mathsf{Dec}^{\mathsf{DMQN}}$ and return the resulting plaintext m.

Challenge queries: Upon a challenge query m_0, m_1:

1. Choose random $b \in \{0, 1\}$.
2. Use $\mathsf{Enc}^{\mathsf{DMQN}}$ to compute $c_i^* = \mathsf{Enc}^{\mathsf{PKE}}_{\mathsf{pk}_i^{\mathsf{vk}_i^*}}(m_b, r)$ for $i = 1, \dots, \mathsf{k}$.
3. Obtain[2] the signature σ^* on $(c_1^*, \dots, c_{\mathsf{k}}^*)$ with respect to vk^*.
4. Return the challenge ciphertext $\psi^* = (\mathsf{vk}^*, c_1^*, \dots, c_{\mathsf{k}}^*, \sigma^*)$.

Note that, if Forge doesn't occur, the simulation of the CCA2 interaction is perfect. Therefore, the probability that $\mathcal{A}'$ breaks the security of the one-time signature scheme is exactly $\mathsf{Pr}[\mathsf{Forge}]$. The one-time strong unforgeability implies that this probability is negligible. $\triangle$

Lemma 6.2 $\left|\mathit{Pr}[b = b^* \wedge \neg \mathit{Forge}] - \frac{1}{2}\right|$ *is negligible.*

Proof Assume that there exists an adversary $\mathcal{A}$ for which $\left|\mathsf{Pr}[b = b^* \wedge \neg \mathsf{Forge}] - \frac{1}{2}\right|$ is not negligible. We build an adversary $\mathcal{A}'$ that breaks the IND-CPA security of PKE_{k}. $\mathcal{A}'$ works as follows:

Key Generation: On input the public key $(\mathsf{pk}_1, \dots, \mathsf{pk}_{\mathsf{k}})$ for PKE_{k}:

1. Execute $\mathsf{KeyGen}^{\mathsf{SS}}$ and obtain a key $(\mathsf{vk}^*, \mathsf{sgk}^*)$.
2. Set $\mathsf{pk}_i^{\mathsf{vk}^*} = \mathsf{pk}_i$ for $i = 1, \dots, \mathsf{k}$.
3. Run $\mathsf{KeyGen}^{\mathsf{PKE}}$ for k times and denote the resulting public keys by $(\mathsf{pk}_1^{1-\mathsf{vk}_1^*}, \dots, \mathsf{pk}_{\mathsf{k}}^{1-\mathsf{vk}_{\mathsf{k}}^*})$ and private keys by $(\mathsf{sk}_1^{1-\mathsf{vk}_1^*}, \dots, \mathsf{sk}_{\mathsf{k}}^{1-\mathsf{vk}_{\mathsf{k}}^*})$.
4. Return the public key $\mathsf{pk} = (\mathsf{pk}_1^0, \mathsf{pk}_1^1, \dots, \mathsf{pk}_{\mathsf{k}}^0, \mathsf{pk}_{\mathsf{k}}^1)$ to $\mathcal{A}$.

[2]Remember that in the one-time strong unforgeability game the adversary is allowed to ask to a signing oracle for the signature on one message.

Decryption queries: Upon a decryption query from $\mathcal{A}$:

1. If Forge occurs output $\perp$ and halt.
2. If $\mathsf{Ver}^{\mathsf{SS}}_{\mathsf{vk}}((c_1, \ldots, c_{\mathsf{k}}), \sigma) = 0$ output $\perp$ and halt.
3. Otherwise, there will be some i such that $\mathsf{vk}_i \neq \mathsf{vk}^*_i$. Decrypt using $\mathsf{Dec}^{\mathsf{PKE}}$ with the key $\mathsf{sk}_i^{\mathsf{vk}_i}$ previously generated and check all the other encryptions, then return either the resulting plaintext m or $\perp$ if the check fails.

Challenge queries: Upon a challenge query m_0, m_1:

1. Send m_0, m_1 to the challenge oracle for the IND-CPA game of $\mathcal{A}'$ and obtain the corresponding challenge ciphertext $(c^*_1, \ldots, c^*_{\mathsf{k}})$.
2. Sign $(c^*_1, \ldots, c^*_{\mathsf{k}})$ using sgk^* to get the signature σ^*.
3. Return the challenge ciphertext $\psi^* = (\mathsf{vk}^*, c^*_1, \ldots, c^*_{\mathsf{k}}, \sigma^*)$.

Output: When $\mathcal{A}$ outputs b^* also $\mathcal{A}'$ outputs b^*.

As long as Forge doesn't occur, it is clear that the IND-CPA advantage of $\mathcal{A}'$ against PKE_{k} is the same as the IND-CCA2 advantage of $\mathcal{A}$ against the above scheme. Since we are assuming the IND-CPA security of PKE_{k}, we have the IND-CCA2 security as desired. $\triangle$

Remark 6.4 It is clear that, as already mentioned by the authors in [89], the IND-CPA security of the "randomized McEliece" scheme is not absolute, but depends on the choice of the sizes of the message $\boldsymbol{m}$ and randomness $\boldsymbol{r}$ in the encryption procedure $(\boldsymbol{r}||\boldsymbol{m})G + \boldsymbol{e}$. In the context of a IND-CPA attack game, in fact, this ciphertext is subject to general decoding attacks with partial information about the plaintext. As illustrated in [89, Table 1], if the randomness $\boldsymbol{r}$ is not large enough, the IND-CPA security of the scheme can be easily broken.

6.4 A Direct Translation

We now explain how to realize the Rosen-Segev scheme using McEliece[3]. The construction arises naturally if we want to be as close as possible to the original McEliece formulation. We hence follow the usual approach of the McEliece cryptosystem, that is to choose a different random error vector every time we call the evaluation algorithm; this implies that we are not using functions anymore. The construction is proved to be secure under k-correlated inputs in Theorem 6.3.

It proceeds as follows:

[3]A similar work has been done by Peikert in [96] for the case of LWE-based lattice encryption.

Describe McEliece as a pair $\mathsf{McE} = (\mathsf{G}, \mathsf{F})$ composed by two algorithms: G is a generation algorithm that samples a description, and F is an evaluation algorithm that provides the evaluation on a given input.

Generation: on input n, k the algorithm G generates a random generator matrix $\hat{G}$ for an $[n, k]$ linear code over $\mathbb{F}_q$ with an efficient decoding algorithm given by the code description Γ, a $k \times k$ random invertible matrix S and an $n \times n$ permutation matrix P, then publishes the public key $G = S\hat{G}P$ and the private key (S, P, Γ).

Evaluation: on input $G, \boldsymbol{m}$ the algorithm F generates a random error vector $\boldsymbol{e}$ of fixed weight w in $\mathbb{F}_q^n$, computes $\psi = \boldsymbol{m}G + \boldsymbol{e}$ and outputs the ciphertext ψ.

It is possible to invert F using the trapdoor: on input (S, P, Γ) and ψ, multiply ψ by P^{-1}, decode to obtain $\boldsymbol{m}S$ and retrieve $\boldsymbol{m}$ by multiplying by S^{-1}.

We claim that, for a certain choice of parameters, this encryption process is secure under k-correlated inputs. This is proved in the following theorem, which closely follows the proof of [45, Th. 6.2]. First, we need a lemma:

Lemma 6.3 *If the indistinguishability assumption (Assumption 2 of Section 3.1.1) holds for parameters $\hat{n}, k$ and $\hat{w}$, then the ensembles $\{(G, \boldsymbol{m}G + \boldsymbol{e}) : G \in \mathbb{F}_q^{k \times \hat{n}}, \boldsymbol{m} \in \mathbb{F}_q^k, \boldsymbol{e} \in \mathbb{W}_{q,\hat{n},\hat{w}}\}$ and $\{(G, \boldsymbol{y}) : G \in \mathbb{F}_q^{k \times \hat{n}}, \boldsymbol{y} \xleftarrow{\$} \mathbb{F}_q^{\hat{n}}\}$ are computationally indistinguishable.*

Proof An equivalent lemma was proved by Fischer and Stern in [43] for the syndrome decoding (Niederreiter) case. We know [69] that the two formulations are equivalent; in particular, any adversary able to distinguish the above ensembles can be used to build an adversary for the Niederreiter case. Consider then the problem of distinguishing the ensembles $\{(H, H\boldsymbol{e}^\intercal) : H \in \mathbb{F}_q^{(\hat{n}-k) \times \hat{n}}, \boldsymbol{e} \in \mathbb{W}_{q,\hat{n},\hat{w}}\}$ and $\{(H, \boldsymbol{y}) : H \in \mathbb{F}_q^{(\hat{n}-k) \times \hat{n}}, \boldsymbol{y} \xleftarrow{\$} \mathbb{F}_q^{\hat{n}-k}\}$ as in [43] and suppose $\mathcal{A}$ is a probabilistic polynomial-time algorithm that is able to distinguish the ensembles of Lemma 6.3. In particular, say $\mathcal{A}$ outputs 1 if the challenge ensemble is of the form $(G, \boldsymbol{m}G + \boldsymbol{e})$ and 0 otherwise. We show how to construct an adversary $\mathcal{A}'$ that solves the above problem.

Let $(H, \boldsymbol{z})$ be the input to $\mathcal{A}'$, where $\boldsymbol{z}$ is either $H\boldsymbol{e}^\intercal$ for a certain error vector $\boldsymbol{e} \in \mathbb{W}_{q,\hat{n},\hat{w}}$ or a random vector of $\mathbb{F}_q^{\hat{n}-k}$. By linear algebra, is easy to find a vector $\boldsymbol{x} \in \mathbb{F}_q^{\hat{n}}$ with $\mathsf{wt}(\boldsymbol{x}) \geq \hat{w}$ such that $\boldsymbol{z} = H\boldsymbol{x}^\intercal$. It is then enough to choose $\boldsymbol{x}$ uniformly at random in the corresponding coset, and submit $(\tilde{G}, \boldsymbol{x})$ to $\mathcal{A}$, where $\tilde{G}$ is the generator matrix associated to H. Now, if $\boldsymbol{z} = H\boldsymbol{e}^\intercal$ we can write $\boldsymbol{x} = \tilde{\boldsymbol{m}}\tilde{G} + \boldsymbol{e}$; in this case, in fact, we have $H\boldsymbol{x}^\intercal = \boldsymbol{z} = H\boldsymbol{e}^\intercal \Longrightarrow H(\boldsymbol{x} - \boldsymbol{e})^\intercal = 0$ and clearly this implies that $(\boldsymbol{x} - \boldsymbol{e})^\intercal$ is a codeword. Then $\mathcal{A}$ will output 1 and so will $\mathcal{A}'$. Otherwise, $\mathcal{A}$ will output 0 and so will $\mathcal{A}'$. In both cases, $\mathcal{A}'$ is able to distinguish correctly and this terminates the proof. $\triangle$

We then state an assumption regarding the computational indistinguishability of some distributions.

Assumption 4 *Let $U_1, \dots, U_{\mathsf{k}}$ be k uniform $k \times n$ matrices and F be the evaluation algorithm defined above. Then the distributions $(U_1, \dots, U_{\mathsf{k}}, \mathsf{F}(U_1, \boldsymbol{m}), \dots, \mathsf{F}(U_{\mathsf{k}}, \boldsymbol{m}))$ and $(U, \mathsf{F}(U, \boldsymbol{m}))$ are computationally indistinguishable*[4].

Note that in the latter distribution the error vector used has length $n\mathsf{k}$ and weight $w\mathsf{k}$. A formal argument is provided in Remark 6.5.

We are now ready to state the theorem.

Theorem 6.3 *Fix an integer k. If the parameters n, k, w are chosen such that decoding a random linear code with parameters $n\mathsf{k}, k$ and $w\mathsf{k}$ is hard and Assumption 4 holds, then the above encryption process is secure under k-correlated inputs.*

Proof Let $\mathcal{A}$ be an adversary for the one-wayness under k-correlated inputs. We define the advantage of $\mathcal{A}$ to be

$$\mathsf{Adv}(\mathcal{A}, \lambda) = \mathsf{Pr}[\mathcal{A}(G_1, \dots, G_{\mathsf{k}}, \mathsf{F}(G_1, \boldsymbol{m}), \dots, \mathsf{F}(G_{\mathsf{k}}, \boldsymbol{m})) = \boldsymbol{m}]$$

where $G_1, \dots, G_{\mathsf{k}}$ are k independent public keys generated by G.
We assume the indistinguishability assumption holds: we can then exchange all the matrices G_i with uniform matrices U_i with a negligible advantage for the attacker. Now, let's define the $k \times n\mathsf{k}$ matrix U by concatenating the rows of the matrices U_i, i.e. $U = (U_1|\dots|U_{\mathsf{k}})$. By Assumption 4, the distributions $(U_1, \dots, U_{\mathsf{k}}, \mathsf{F}(U_1, \boldsymbol{m}), \dots, \mathsf{F}(U_{\mathsf{k}}, \boldsymbol{m}))$ and $(U, \mathsf{F}(U, \boldsymbol{m}))$ are interchangeable without a significant advantage for the attacker. We now invoke Lemma 6.3 with $\hat{n} = n\mathsf{k}$ and $\hat{w} = w\mathsf{k}$. Hence

$$\mathsf{Adv}(\mathcal{A}, \lambda) = \mathsf{Pr}[\mathcal{A}(U, \mathsf{F}(U, \boldsymbol{m})) = \boldsymbol{m}] - \mathsf{Pr}[\mathcal{A}(U, \boldsymbol{y}) = \boldsymbol{m}] \in \text{negl}(n)$$

and since this last one is of course negligible, we conclude the proof. $\triangle$

Remark 6.5 Similarly to the case of the IND-CPA security of the McEliece variant (as pointed out in Remark 6.4), the security we are trying to achieve is not absolute, but depends on a suitable choice of parameters. Assumption 4 consists of replacing the vector $(\boldsymbol{m}U_1 + \boldsymbol{e}_1||\dots||\boldsymbol{m}U_{\mathsf{k}} + \boldsymbol{e}_{\mathsf{k}})$ with the vector $\boldsymbol{m}U + \boldsymbol{e}$, where $U = (U_1|\dots|U_{\mathsf{k}})$ and $\boldsymbol{e}$ is a random error vector of weight $w\mathsf{k}$; in other words, we would like to argue that $\boldsymbol{e}' = (\boldsymbol{e}_1||\dots||\boldsymbol{e}_{\mathsf{k}})$ is computationally indistinguishable from $\boldsymbol{e}$. Note that $\mathsf{wt}(\boldsymbol{e}') = \mathsf{wt}(\boldsymbol{e})$ but while the distribution of the error positions on $\boldsymbol{e}$ is truly pseudorandom, $\boldsymbol{e}'$ is formed by k blocks of weight w each. It is plausible that the number of vectors of this kind (that we denote $\#_{\boldsymbol{e}'}$) is not too small compared to the total of error vectors with same length and weight. We can use the well-known bound $\binom{n}{w} \approx 2^{nh_2(w)} + \varepsilon$, where $h_2 : \mathbb{R} \to \mathbb{R}$ is the usual binary entropy function defined by $h_2(x) = -x \log_2 x - (1-x)\log_2(1-x)$ and ε is a small approximation error. We then have the following estimate:

[4]For a formal definition, see for example Yao [126].

$$\frac{\#_{\boldsymbol{e}'}}{|\mathbb{W}_{q,n\mathsf{k},w\mathsf{k}}|} = \frac{\binom{n}{w}^{\mathsf{k}}}{\binom{n\mathsf{k}}{w\mathsf{k}}} \approx \frac{(2^{nh_2(w)}+\varepsilon)^{\mathsf{k}}}{2^{n\mathsf{k}h_2(w\mathsf{k})}+\varepsilon}. \tag{6.2}$$

so that as ε approaches 0, the above ratio approaches 1 as desired.

One can then implement the Rosen-Segev scheme using this choice of F and G. For completeness we present the details below.

$\mathsf{KeyGen}^{\mathsf{NEW}}$: Invoke G for $2\mathsf{k}$ times independently and obtain the collections of public keys $\mathsf{pk} = (\mathsf{pk}_1^0, \mathsf{pk}_1^1, \dots, \mathsf{pk}_\mathsf{k}^0, \mathsf{pk}_\mathsf{k}^1)$ and private keys $\mathsf{sk} = (\mathsf{sk}_1^0, \mathsf{sk}_1^1, \dots, \mathsf{sk}_\mathsf{k}^0, \mathsf{sk}_\mathsf{k}^1)$, where $\mathsf{pk}_j^i = G_j^i$ and $\mathsf{sk}_j^i = (S, P, \Gamma)_j^i$ as above.
$\mathsf{Enc}^{\mathsf{NEW}}$: To encrypt a plaintext m with the public key pk, sample a key $(\mathsf{vk}, \mathsf{sgk})$ and a random $x \in \{0,1\}^k$, then:

1. Evaluate $c_i = \mathsf{F}(\mathsf{pk}_i^{\mathsf{vk}_i}, x)$ for $i = 1, \dots, \mathsf{k}$.
2. Set $y = m \oplus h(\mathsf{pk}_1^{\mathsf{vk}_1}, \dots, \mathsf{pk}_\mathsf{k}^{\mathsf{vk}_\mathsf{k}}, x)$.
3. Compute $\sigma = \mathsf{Sign}_{\mathsf{sgk}}^{\mathsf{SS}}(c_1, \dots, c_\mathsf{k}, y)$.

where vk_i represents the i-th bit of vk. As in [106] we can assume m to be a single bit, in which case h describes a hard-core predicate for McEliece; the protocol extends easily to multiple bits plaintexts.
Finally, output the ciphertext $\psi = (\mathsf{vk}, c_1, \dots, c_\mathsf{k}, y, \sigma)$.

$\mathsf{Dec}^{\mathsf{NEW}}$: Upon reception of a ciphertext ψ:

1. Verify the signature; if $\mathsf{Ver}_{\mathsf{vk}}^{\mathsf{SS}}((c_1, \dots, c_\mathsf{k}, y), \sigma) = 0$ output $\perp$.
2. Otherwise compute[5] $x_i = \mathsf{F}^{-1}(\mathsf{sk}_i^{\mathsf{vk}_i}, c_i)$ for $i = 1, \dots, \mathsf{k}$.
3. If $x_1 = \cdots = x_\mathsf{k}$ then set $m = y \oplus h(\mathsf{pk}_1^{\mathsf{vk}_1}, \dots, \mathsf{pk}_\mathsf{k}^{\mathsf{vk}_\mathsf{k}}, x_1)$ and return the plaintext m, otherwise output $\perp$.

The security is assessed in the following corollary:

Corollary 6.4 *The above encryption scheme is IND-CCA2 secure in the standard model.*

Proof By Theorem 6.3, the collection of McEliece encryption schemes McE is k-correlation secure. Then this is analogous to Theorem 6.1, noting that the same argument applies when $\mathcal{F} = \mathsf{McE}$, i.e. f describes a randomized algorithm rather than a function. The proof uses the same steps as in Theorem 6.2, with the exception that in our case Lemma 6.2 is proved by constructing an adversary $\mathcal{A}'$ that works as a predictor for the hard-core predicate h. $\triangle$

[5]By analogy with the Rosen-Segev scheme. Clearly in practice it would be much more efficient, rather than decoding k ciphertexts, to just decode one and then re-encode and test as in [36, Th. 3].

6.5 Conclusions

The scheme of Dowsley et al. [36] is a first proposal to translate the Rosen-Segev protocol to the McEliece framework. However, the construction is ambiguous, as we have shown in Section 6.3.2. Another criticism of the Dowsley, Müller-Quade, Nascimento idea is the strange and unnecessary "forgetting" of half the private keys, and forbidding ciphertexts to feature the verification key vk^*. The original Rosen-Segev scheme has no such requirements.
We therefore present a construction that successfully deals with the problem, providing a choice of algorithms F and G that can be used directly into the Rosen-Segev scheme preserving the original framework.

Signatures

7.1 Introduction

Digital signatures (Section 2.1.4) are a very important cryptographic protocol in the modern world. Among the most popular there are schemes based on the RSA assumptions, discrete logarithm (DSA) and elliptic curves (ECDSA), all included in the FIPS standard 186-3 [70]. On the other hand, many schemes based on coding theory have been proposed over the years, either following a "direct" approach like CFS (Courtois, Finiasz and Sendrier [29]) and KKS (Kabatianskii, Krouk and Smeets [62]), or converting a zero-knowledge identification scheme with the help of the Fiat-Shamir transform [41]. Code-based identification schemes are usually built via a 3-pass protocol (Véron [125]) or, more recently, a 5-pass protocol (Cayrel, Véron and El Yousfi [24]), in turn relying on the work of Stern [120, 121]. Unfortunately, all of the above are highly inefficient in practical situations, due mainly to a huge public key, a large signature and a slow signing algorithm. This usually comes from having to repeat the protocol many times in order to guarantee correctness or security. In this chapter we present the state-of-art work in coding signatures, and point out the main difficulties in designing a secure and efficient scheme based on coding theory.
The chapter is organized as follows: in the next section we present the classical proposals for code-base schemes that we have already mentioned, including CFS, KKS and the identification schemes by Stern, Véron and Cayrel et al. In Section 7.3 we illustrate some recent proposals for lattice signatures, and we discuss the feasibility or unfeasibility of such an approach for code-based schemes. We conclude in Section 7.4.

7.2 Existing Schemes

7.2.1 CFS

A natural approach for code-based signatures would be to follow the usual *hash and sign* framework that is the base of the very famous Full Domain Hash (FDH) signature scheme (Bellare and Rogaway [7, 8]). This is a very efficient signature scheme based on the RSA assumptions, and it has been proved to be existentially unforgeable under adaptive chosen-message attacks in the random oracle model, hence achieving the maximum desirable level of security (see for example Coron [28]). The framework in its simplest version makes use of a trapdoor one-way function f and a hash function $\mathcal{H}$ whose output is an element of the domain of f; the hash function is modelled as a random oracle. The key feature is the use of the one-way function in a "reverse" way compared to how it would be used in the related cryptosystem. The procedure is as follows: first, the message that is to be signed is hashed, then the trapdoor is applied to it. The signature is $\sigma = f^{-1}(\mathcal{H}(\mu))$. The verification is public and can be performed with the public key; this is applied directly to the received signature, and the verifier then computes himself the hash value and checks that $f(\sigma) = \mathcal{H}(\mu)$. For clarification, we present the basic RSA-FDH version below.

Table 7.1: The FDH Signature Scheme.

K	$\mathsf{K}_{\mathsf{sign}}$ the group $\mathbb{Z}^*_{\varphi(N)}$. $\mathsf{K}_{\mathsf{ver}}$ the group $\mathbb{Z}^*_{\varphi(N)}$.
M	The set of binary strings (of arbitrary length) $\{0,1\}^*$.
Σ	The ring $\mathbb{Z}_N$.
KeyGen	Fix an RSA modulus N and a hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_N$. Choose at random an encryption exponent $e \in \mathbb{Z}^*_{\varphi(N)}$ and compute its inverse d modulo $\varphi(N)$. Return the signing key $d \in \mathsf{K}_{\mathsf{sign}}$ and the verification key $e \in \mathsf{K}_{\mathsf{ver}}$.
Sign	On input a signing key $d \in \mathsf{K}_{\mathsf{sign}}$ and a message $\boldsymbol{\mu} \in \mathsf{M}$, compute $y = \mathcal{H}(\boldsymbol{\mu})$ and return the signature $\sigma = y^d \pmod N \in \Sigma$.
Ver	On input a verification key $e \in \mathsf{K}_{\mathsf{ver}}$, a message $\boldsymbol{\mu} \in \mathsf{M}$ and a signature $\sigma \in \Sigma$, compute $y = \mathcal{H}(\boldsymbol{\mu})$ and $y' = \sigma^e \pmod N$, then output 1 if $y = y'$, else return 0.

Unfortunately, it is easy to see that this approach can't be applied directly to the coding theory setting. Consider without loss of generality the Niederreiter trapdoor function (a similar argument can be given for McEliece) for an $[n, k]$ linear code over $\mathbb{F}_q$, and suppose $\mathcal{H}$ is a random oracle mapping bit strings to words of $\mathbb{F}_q^r$, where $r = n - k$. In general, a randomly chosen syndrome does not decode uniquely, so the signing algorithm fails. The idea of Courtois, Finiasz and Sendrier in [29] is to sequentially add an integer c to the input[1] of $\mathcal{H}$ and to test if $\mathcal{H}(\boldsymbol{\mu}, c)$ is a decodable syndrome, iterating the procedure until such a syndrome is found. This can be done more efficiently, for instance by *random counter sampling* in the set $\{0, \ldots, 2^r - 1\}$ as pointed out by Dallot [32].
It is clear that in general the process is not efficient: with the original McEliece parameters $n = 1024, k = 524, w = 50, q = 2$ there are exactly 2^{500} distinct syndromes, of which only $\sum_{i=1}^{w} \binom{n}{i} \approx 2^{284}$ are decodable. Thus an average of 2^{216} decoding attempts would be needed, which is obviously not plausible. Even if the parameters are adjusted like suggested by the authors ($m = 16, w = 9$ with $n = 2^m, r = mw$, for a security level of 2^{80}), the scheme is far from practical: in order to sign it is necessary to repeat the algorithm in average 9! times, with the additional disadvantage of a very big public key (1152 Kbytes). On the other hand, the verification is intuitively very fast (similarly to the Niederreiter encryption process, it is just a matrix-vector multiplication) and the signature size can be considerably shortened, thanks to an indexing trick, to reach a size comparable to other schemes; however, the two above flaws are so limiting that the disadvantage of CFS is still too great in many applications.

Remark 7.1 Note that the use of McEliece in this context would be even worse, since the above mentioned indexing trick relies on the fact that the signature is a vector of very low weight, and couldn't be applied in this case. The signature

[1]This can be realized, for example, by concatenating μ and the bit string corresponding to the integer c.

would therefore have at least size $k = n - mw$, which is of course too big for any suitable set of parameters.

Table 7.2: The CFS Signature Scheme.

Setup	Fix public system parameters $q, n, r, w \in \mathbb{N}$.
K	$\mathsf{K}_{\mathsf{sign}}$ the set of triples formed by a $k \times k$ invertible matrix over $\mathbb{F}_q$, an $n \times n$ permutation matrix over $\mathbb{F}_q$ and a code description.
	$\mathsf{K}_{\mathsf{ver}}$ the set of $k \times n$ matrices over $\mathbb{F}_q$.
M	The set of binary strings (of arbitrary length) $\{0,1\}^*$.
Σ	The vector space $\mathbb{F}_q^r$.
KeyGen	Generate a random parity-check matrix $\hat{H}$ for an $[n, n-r]$ linear code over $\mathbb{F}_q$ with an efficient decoding algorithm given by the code description Γ, a $r \times r$ random invertible matrix S and an $n \times n$ permutation matrix P. Return the signing key $(S, P, \Gamma) \in \mathsf{K}_{\mathsf{sign}}$ and the verification key $H = S\hat{H}P \in \mathsf{K}_{\mathsf{ver}}$.
Sign	On input a signing key $(S, P, \Gamma) \in \mathsf{K}_{\mathsf{sign}}$ and a message $\boldsymbol{\mu} \in \mathsf{M}$, find c such that $\boldsymbol{y} = \mathcal{H}(\boldsymbol{\mu}, c)$ is a decodable syndrome, decode $\boldsymbol{y}$ to recover $\boldsymbol{e} \in \mathbb{W}_{q,n,w}$ and return the signature $(\boldsymbol{e}, c) \in \Sigma$.
Ver	On input a verification key $H \in \mathsf{K}_{\mathsf{ver}}$, a message $\boldsymbol{\mu} \in \mathsf{M}$ and a signature $\sigma = (\boldsymbol{e}, c) \in \Sigma$, compute $\boldsymbol{y} = \mathcal{H}(\boldsymbol{\mu}, c)$ and $\boldsymbol{y}' = H\boldsymbol{e}^{\mathsf{T}}$, then output 1 if $y = y'$ and $\mathsf{wt}(\boldsymbol{e}) \leq w$, else return 0.

To give a better picture, we now present an extract of [29, Table 6], including three variants of the above CFS set of parameters ($m = 16, w = 9$). These are simple trade-off techniques optimized for fast verification (CFS1), short signature (CFS3) or halfway between the two (CFS2); see [29, Section 5.3] for more details. In the table below, "Data Size" specifies the instance of the scheme, for example the size of the RSA modulus in FDH, the syndrome length in CFS etc.

Table 7.3: Comparison of some of the most popular signature schemes. The signature size is given in bits, the public key size in kilobytes and all the timings are measured on a machine running at 1GHz.

Hard Problem	Factoring	Disc. Log	Ell. Curves	Syndrome Decoding		
Scheme	RSA-FDH	DSA	ECDSA	CFS1	CFS2	CFS3
Data Size (bits)	1024	160/1024	160	144	144	144
Security	2^{80}	2^{80}	2^{80}	2^{80}	2^{80}	2^{80}
Signature (bits)	1024	320	321	132	119	81
Public Key (Kb)	0.2	0.1	0.1	1152	1152	1152
Signing	9 ms	1.5 ms	5 ms	10-30s	10-30s	10-30s
Verification	9 ms	2 ms	6 ms	$< 1\ \mu$s	< 1 ms	≈ 1s

A Generalized Birthday Attack (GBA) due to Bleichenbacher was presented by Finiasz and Sendrier in [42], with the result that the original set of parameters ($m = 16, w = 9$) can't be considered secure anymore. The authors propose instead ($m = 21, w = 10$) or ($m = 15, w = 12$). Barreto, Cayrel, Misoczki and Niebuhr in [2] presented a variant of CFS using the quasi-dyadic framework (Section 3.3.2). The public key size is considerably reduced and, among the various trade-offs between key size and signing complexity, intermediate choices seem the more appropriate: for example ($m = 15, w = 12$) results in a public key of 169 Kbytes, although in average $2^{29.8}$ repetitions are needed to sign, which is still very high.

Remark 7.2 Note that all the above "CFS-friendly" codes have a very high rate. Thus, if Goppa codes are used, the scheme is likely to be susceptible to the distinguishing attack already mentioned in Remark 3.1. Note also that there is currently no known distinguisher for GS codes, that could thus be considered a safer choice.

7.2.2 KKS

A diametrically opposed approach was introduced in 1997 by Kabatianskii, Krouk and Smeets [62]. Their scheme, in fact, realizes signatures *without decoding*. The approach seems promising since it avoids the main problem of CFS-like signatures, that is, that a randomly generated syndrome is in general not decodable. Moreover, as decoding is not involved, the scheme in principle does not need to use special families of codes with an efficient decoding algorithm. However, some issues are still arising from the construction, as we will see later. The basic scheme is presented below.

Table 7.4: The KKS Signature Scheme.

Setup	Fix public system parameters $q, N, n, r, k, w_1, w_2 \in \mathbb{N}$.
K	$\mathsf{K}_{\mathsf{sign}}$ the set of pairs formed by a set of integers of cardinality n, and a $k \times n$ matrix over $\mathbb{F}_q$.
	$\mathsf{K}_{\mathsf{ver}}$ the set of pairs formed by an $r \times k$ matrix and an $r \times N$ matrix, both over $\mathbb{F}_q$.
M	The vector space $\mathbb{F}_q^k$.
Σ	The vector space $\mathbb{F}_q^N$.
KeyGen	Generate a random parity-check matrix H for an $[N, N-r]$ linear code $\hat{\mathcal{C}}$ over $\mathbb{F}_q$ and a generator matrix G for an $[n, k, w_1]$ linear code $\mathcal{C}$ over $\mathbb{F}_q$ such that $\mathsf{wt}(\boldsymbol{x}) \leq w_2$ for all $\boldsymbol{x} \in \mathcal{C}$, then choose a size-$n$ subset $J \subset \{1, \dots, N\}$. Return the signing key $(J, G) \in \mathsf{K}_{\mathsf{sign}}$ and the verification key $(F, H) \in \mathsf{K}_{\mathsf{ver}}$ where $F = H_J G^\mathsf{T}$ and H_J is the $r \times n$ submatrix indexed by J.
Sign	On input a signing key $(J, G = \{g_{i,j}\}) \in \mathsf{K}_{\mathsf{sign}}$ and a message $\boldsymbol{\mu} \in \mathsf{M}$, form G^* by setting $g_{i,j} = 0 \, \forall j \notin J$, then compute $\boldsymbol{\sigma} = \boldsymbol{\mu} G^*$ and return the signature $\boldsymbol{\sigma} \in \Sigma$.
Ver	On input a verification key $(F, H) \in \mathsf{K}_{\mathsf{ver}}$, a message $\boldsymbol{\mu} \in \mathsf{M}$ and a signature $\boldsymbol{\sigma} \in \Sigma$, output 1 if $w_1 \leq \mathsf{wt}(\boldsymbol{\sigma}) \leq w_2$ and $F\boldsymbol{\mu}^\mathsf{T} = H\boldsymbol{\sigma}^\mathsf{T}$, else return 0.

It is easy to see that the verification step works for correctly formed signatures since $HG^{*T} = H_J G^\mathsf{T}$.

The authors' first proposal is to choose $\mathcal{C}$ to be an *equidistant* code, i.e. a code whose codewords are all at the same distance from each other. In this case we would have $w_1 = w_2 = q^{k-1}$ and $N \geq n = \frac{q^k-1}{q-1}$. Unfortunately, this straightforward approach isn't applicable in practice. For the binary case, for example, the number of distinct signatures is exactly 2^k, hence for any desirable security level ($k = 128$ or 256) the value of N is too large. Three alternatives are given in [62].

First Variant

The first variant consists of choosing $\mathcal{C}$ to be the dual of a binary BCH code (see Definition 2.24). The bound on the weight of the codewords is guaranteed by the following lemma.

Lemma 7.1 (Carlitz-Uchiyama Bound) *Let $\mathcal{C}$ be the dual of a binary BCH code of length $n = 2^m - 1$ and designated distance $\delta = 2s + 1$. Then for any $\boldsymbol{x} \in \mathcal{C}$:*

$$\left| \mathsf{wt}(\boldsymbol{x}) - \frac{n+1}{2} \right| \leq (s-1)\sqrt{n+1}. \tag{7.1}$$

In addition, the scheme features also an invertible $k \times k$ matrix A in order to mask the structure of G, so the matrix F is now defined as $H_J(AG)^\intercal$. The following choice of parameters is suggested: $m = 10, s = 6, k = ms = 60, n = 2^m - 1 = 1023, w_1 = 352, w_2 = 672, r = 2808$ and $N = 3000$.

Second Variant

In the second variant $\mathcal{C}$ is chosen as a random binary code. The bound is in this case satisfied with a large probability, which is estimated in the following proposition ([62, Prop. 3]).

Proposition 7.1 *Let $\mathcal{C}$ be a randomly chosen $[n, k]$ binary code given by a generator matrix in systematic form, and fix $\delta \in \mathbb{R}$. Let h_2 be the binary entropy function as defined in Remark 6.5. Then the probability that $\mathsf{wt}(\boldsymbol{x}) \in [\frac{n}{2}(1-\delta), \frac{n}{2}(1+\delta)]$ for every non-zero codeword $\boldsymbol{x} \in \mathcal{C}$ is at least*

$$1 - 2^{-r+nh_2(\delta)+1}. \tag{7.2}$$

This can be easily generalized to the q-ary case by replacing 2 with q and h_2 with h_q in (7.2), where $h_q(x) = x\log_q(q-1) - x\log_q x - (1-x)\log_2(1-x)$ for all $x \in \mathbb{R}$. For example with the proposed parameters $k = 160, n = 900, w_1 = 90, w_2 = 110, r = 1100$ and $N = 2000$ this probability is at least $1 - 2^{-749}$.

Third Variant

The idea of this variant is to construct a the code $\mathcal{C}$ starting from smaller codes for which is known that the codewords have low weight. In particular, $\mathcal{C}$ is formed as direct product of P distinct $[n^*, k^*, w_1^*]$ linear codes $\mathcal{C}_i$ over $\mathbb{F}_q$, each having codewords with weight less than w_2^*. We will then have $n = Pn^*, k = Pk^*, w_1 = Pw_1^*, w_2 = Pw_2^*$. The construction also makes use of P invertible $k^* \times k^*$

matrices $A_1, \dots, A_P$ and P non-zero elements $\beta_1, \dots, \beta_P$ of the field $\mathbb{F}_{q^{k^*}}$. To each of these elements is associated a matrix M_{β_i} representing the linear map $x \mapsto x\beta_i$. The public key matrix F is then defined as $F = (F_1, \dots, F_Q)$ for a certain integer Q, where for $j = 1, \dots, P$ we call F_j the $r \times k^*$ matrix given by

$$F_j = \sum_{i=1}^{P} H_{J_i}(M_{\beta_i^{j-1}} A_i G_i)^\mathsf{T}, \tag{7.3}$$

where $J_1, \dots, J_P$ are disjoint size-n^* subsets of $\{1, \dots, n\}$ and G_i is the generator matrix of the code $\mathcal{C}_i$. Suggested parameters are $Q = 14, P = 12, k^* = 4, n^* = 15$ with the $\mathcal{C}_i$ being all equal to a binary equidistant code having $w_1^* = w_2^* = 8$.

Further discussion on KKS is given by Cayrel, Otmani and Vergnaud in [23], along with a new set of parameters for the second variant. We report the numbers in the next table; the public key size is expressed in kilobytes.

Table 7.5: Comparison of parameters for the KKS variants with binary codes (q=2).

Variant	N	r	n	k	w_1	w_2	Public Key (Kb)
First	3000	2808	1023	60	352	672	86.4
Second	1250	990	280	60	50	230	38.7
Second (Cayrel et al.)	2000	1100	1000	160	90	110	142.3
Third	1100	765	180	48	96	96	35.8

Remark 7.3 The parameters proposed by Cayrel et al. are tailored to provide a level of security of 2^{80} against the general decoding attack of Canteaut and Chabaud [21], which the other proposals fail to achieve. As we know from Section 3.2.1, there exist more recent general decoding attacks that make also these new parameters insecure.

It is easy to notice that, despite a reasonable signature size (few hundred bits), the public key size is still very large. However, the real concern is the security of the scheme; in fact, most of the original proposals can be broken after recovering just a few signatures. This is because every message/signature pair reveals some information on the secret support J (on average half of the positions); the attack, described in detail in [23], succeeds to recover J with a workfactor of approximately 2^{80} operations with 13 and 20 signatures for the first two variants and just 5 signatures for the third variant. There is a slight improvement with the new parameters by Cayrel et al. (about 40 signatures), but this is clearly still too vulnerable. The scheme seems therefore to be suitable only as a one-time signature. In particular, Barreto, Misoczki and Simplício in [3] proposed a variant that achieves one-time existential unforgeability against chosen message attack. This is accomplished by simply using the basic KKS framework together with a hash function $\mathcal{H}$ and with the addition of an error vector in the signature, in the following way: the signer samples a random error vector $\boldsymbol{e}$ of weight n, computes the hash value $\boldsymbol{h} = \mathcal{H}(\boldsymbol{\mu}, H\boldsymbol{e}^\mathsf{T})$ and returns the signature $\boldsymbol{\sigma} = \boldsymbol{h}G^* + \boldsymbol{e}$. The verifier then checks that $\mathsf{wt}(\boldsymbol{\sigma}) \leq 2n$ and that $\boldsymbol{h} = \mathcal{H}(\boldsymbol{\mu}, H\boldsymbol{\sigma}^\mathsf{T} + F\boldsymbol{h}^\mathsf{T})$. Some parameters are presented below.

Table 7.6: Example of parameters for the KKS variant of [3].

N	r	n	k	w_1	w_2	Public Key (Kb)
11626	5813	320	160	133	187	8363.3
16294	8147	448	224	192	256	16427.3
18586	9293	512	256	222	290	21374.3
27994	13997	768	384	342	426	48487.2
37274	18637	1024	512	464	560	85964.1

Recently, an attack by Otmani and Tillich [91] managed to break all the parameters proposed in the literature, including the above one-time scheme, without even needing to know a single message/signature pair. The attack exploits the fact that, even if H and G are chosen at random, the matrix $\tilde{H} = (H|F)$ describes a code that does not behave as a random code. In particular:

- The left and the right part are related by the equation $F = H_J G^{\mathsf{T}}$.
- There are many low-weight codewords.
- The support of the codewords is limited to a very small subset of positions (of size $w_2 + k$ or $n + k$ in the one-time variant).
- Part of the support is already known to the attacker (the rightmost k positions).

Now, it is clear that low-weight codewords of the code described by $\tilde{H}$ are valid message/signature pairs for the scheme. The idea is then to use general decoding algorithms like ISD to look for low-weight codewords. It turns out that these algorithms work better than usual, because of the above properties; all the parameters proposed are broken with timings that range from a few milliseconds (Table 7.5, last row) to approximately 6 minutes (Table 7.6, last row). In particular, if I is the information set chosen in an iteration of the algorithm and $I' = I \cap J$, we expect to have $k/|I'| \approx \rho/R$, where ρ and R are the transmission rates of, respectively, $\mathcal{C}$ and $\hat{\mathcal{C}}$. This value is very close to 1 for the original KKS parameters and is exactly 1 (since both rates are equal to 1/2) for the proposal by Barreto et al., so the number of required iterations is very small (less than two in the last case); this explains the fast timings of the attack. On the other hand, it is the evident that in order to avoid the attack ρ would need to be significantly smaller than R, leading to even more impractical sets of parameters.

7.2.3 Identification Schemes and Fiat-Shamir

In modern cryptography, a *Zero-Knowledge Identification Scheme* is a protocol that allows one party, called the Prover, to prove to another party, the Verifier, that he possesses secret information, without revealing to the verifier what that secret information is. The paradigm works as follows: suppose that the prover $\mathcal{P}$ wants to prove to the verifier $\mathcal{V}$ the knowledge of some secret information s; $\mathcal{V}$ is equipped with a public key pk and the public data D. To start, $\mathcal{P}$ chooses some random data y and commits to it by sending $Y = f(y)$ to $\mathcal{V}$, where f

is usually a trapdoor one-way function or a hash function. $\mathcal{V}$ then chooses a random challenge c and sends it to $\mathcal{P}$. After receiving c, $\mathcal{P}$ computes a response z as a function of s, c and y and transmits z. Finally, $\mathcal{V}$ checks that z is correctly formed with the help of pk and D.

Table 7.7: Paradigm of 3-pass Zero-Knowledge Identification Scheme.

Public Data	D.
Private Key	s.
Public Key	pk.

PROVER		VERIFIER
Choose random y and compute $Y = f(y)$.	$\xrightarrow{Y}$	
	$\xleftarrow{c}$	Choose random challenge c.
Compute the response $z = z(s, c, y)$.	$\xrightarrow{z}$	Verify z using pk and D.

A classical example is the Feige-Fiat-Shamir identification scheme [40], based on the Quadratic Residuosity (QR) hard problem, which we describe below.

Table 7.8: Feige-Fiat-Shamir Identification Scheme.

Public Data	An RSA modulus $N = pq$.
Private Key	$s_1, \ldots, s_k$ with $(s_i, N) = 1$.
Public Key	$v_1, \ldots, v_k$ with $v_i = s_i^2 \pmod{N}$.

PROVER		VERIFIER
Choose $y \overset{\$}{\leftarrow} \mathbb{Z}$ and $r \overset{\$}{\leftarrow} \{-1, 1\}$, then set $Y = ry^2 \pmod{N}$.	$\xrightarrow{Y}$	
	$\xleftarrow{c}$	$c = (c_1, \ldots, c_k)$ with $c_i \overset{\$}{\leftarrow} \{0, 1\}$.
Compute $z = ys_1^{c_1} s_2^{c_2} \ldots s_k^{c_k} \pmod{N}$.	$\xrightarrow{z}$	Accept if $z^2 = \pm Y v_1^{c_1} v_2^{c_2} \ldots v_k^{c_k} \pmod{N}$.

Security is assessed with regard to two different types of adversaries. A *zero-knowledge attacker* tries to extract information the protocol, in order to recover the secret s; in this sense, even an honest verifier is considered as an adversary for the scheme. An *impersonator*, instead, tries to replace the prover and to produce a response that is accepted as valid without the knowledge of s. Both adversaries are allowed to have access not only to the public key and the public data, but also to the information exchanged during any number of interactions between the prover and the verifier.
A correctly designed zero-knowledge identification scheme always features a *zero-knowledge proof* that deals with the first kind of attacks, while for the second kind, it should be at least computationally hard for an impersonator to produce a valid response. For example, in the above scheme, the authors provide a zero-

knowledge proof that relies upon the hardness of QR. As for forgeries, note that an impersonator would succeed if able to predict the k bits of c in advance. If that is the case, in fact, to pass the verification it would be enough to choose a random y, commit $Y = y^2 v_1^{-c_1} v_2^{-c_2} \dots v_k^{-c_k} \pmod N$ and, after receiving c, reply with $z = y$. However, the probability of guessing k bits is $\frac{1}{2^k}$, so the scheme is secure for large enough k.

Identification schemes are of particular interest because it is possible to convert them into efficient signature schemes via the very famous Fiat-Shamir transform [41]. The signer simply runs the protocol, where, for the purpose of generating the challenge, the verifier is replaced by a random oracle $\mathcal{H}$. The signature is then accepted according to the validity of the response in the identification scheme.

Table 7.9: The Fiat-Shamir Signature Scheme.

Setup	Select a zero-knowledge identification scheme $\mathcal{I}$.
Sign	On input the private key of $\mathcal{I}$ and a message μ, commit Y, set $c = \mathcal{H}(Y, \mu)$, compute a response z and return the signature $\sigma = (Y, z)$.
Ver	On input the public key of $\mathcal{I}$, a message μ and a signature σ, set $c = \mathcal{H}(Y, \mu)$ then output 1 if z is accepted in $\mathcal{I}$, else return 0.

The first code-based identification scheme relies on the hardness of syndrome decoding, and was introduced in 1993 by Stern [120]. This is still a 3-pass protocol like the Feige-Fiat-Shamir scheme, but it follows a slightly different framework that makes use of multiple commitments. The scheme was then converted by Véron [125] to an equivalent protocol that relies on GDP.

Table 7.10: Stern Identification Scheme.

Public Data	The parameters $n, k, w \in \mathbb{N}$, an $(n-k) \times n$ parity-check matrix H over $\mathbb{F}_2$ and a hash function $\mathcal{H}$.
Private Key	$\boldsymbol{s} \in \mathbb{W}_{2,n,w}$.
Public Key	$\boldsymbol{S} = H\boldsymbol{s}^{\mathsf{T}}$.

PROVER		VERIFIER
Choose $\boldsymbol{y} \xleftarrow{\$} \mathbb{F}_2^n$ and a permutation $\pi \xleftarrow{\$} \mathrm{Sym}(n)$, then set $c_1 = \mathcal{H}(\pi, H\boldsymbol{y}^{\mathsf{T}})$, $c_2 = \mathcal{H}(\pi(\boldsymbol{y})), c_3 = \mathcal{H}(\pi(\boldsymbol{y}+\boldsymbol{s}))$.	$\xrightarrow{c_1, c_2, c_3}$	
	$\xleftarrow{b}$	$b \xleftarrow{\$} \{0, 1, 2\}$.
If $b = 0$ set $z = (\boldsymbol{y}, \pi)$.		Accept if c_1 and c_2 are correct.
If $b = 1$ set $z = (\boldsymbol{y}+\boldsymbol{s}, \pi)$.	$\xrightarrow{z}$	Accept if c_1 and c_3 are correct.
If $b = 2$ set $z = (\pi(\boldsymbol{y}), \pi(\boldsymbol{s}))$.		Accept if c_2 and c_3 are correct and $\mathsf{wt}(\pi(\boldsymbol{s})) = w$.

It is easy to see that an honest prover is always accepted. In this case the protocol is said to be *complete*.
In terms of zero-knowledge, the scheme admits a simple proof. Very informally, the only data revealed during a run of the protocol is the two random objects $\boldsymbol{y}$ and π, the permuted strings $\pi(\boldsymbol{y})$ and $\pi(\boldsymbol{s})$ and the padding $\boldsymbol{y}+\boldsymbol{s}$. Clearly, $\boldsymbol{y}, \pi$ and $\pi(\boldsymbol{y})$ provide no information on s; the permuted string $\pi(\boldsymbol{s})$ doesn't leak anything apart from the weight of $\boldsymbol{s}$ (which is already known), while $\boldsymbol{y}+\boldsymbol{s}$ acts like a one-time pad. This is because $\boldsymbol{y}$ is randomly chosen, hence on average we will have $\mathsf{wt}(\boldsymbol{y}) = n/2$ and this is large enough to mask the support of $\boldsymbol{s}$. We will see in the next section how choosing these last two actions in a different way can become dangerous for the scheme.

The biggest flaw of the scheme is that it is very easy for an impersonator to provide a forgery. More specifically, an impersonator would be able to reply correctly to two of the three challenges, arbitrarily, in the following way:

- The impersonator chooses random $\boldsymbol{y}$ and π plus another string $\boldsymbol{x} \in \mathbb{F}_2^n$ (not necessarily of low weight) such that $H\boldsymbol{x}^\intercal = H\boldsymbol{y}^\intercal + \boldsymbol{S}$, then builds c_1 and c_2 normally and $c_3 = \mathcal{H}(\pi(\boldsymbol{x}))$. It replies to the challenge with $(\boldsymbol{y}, \pi)$ if $b=0$, or $(\boldsymbol{x}, \pi)$ if $b=1$. The strategy fails for $b=2$.

- The impersonator chooses random $\boldsymbol{y}$ and π plus another random string $\boldsymbol{x} \in \mathbb{W}_{2,n,w}$, then builds c_1 and c_2 normally and $c_3 = \mathcal{H}(\pi(\boldsymbol{y}+\boldsymbol{x}))$. It replies to the challenge with $(\boldsymbol{y}, \pi)$ if $b=0$ or $(\pi(\boldsymbol{y}), \pi(\boldsymbol{x}))$ if $b=2$. The strategy fails for $b=1$.

- The impersonator chooses random $\boldsymbol{y}$ and π plus another random string $\boldsymbol{x} \in \mathbb{W}_{2,n,w}$, then builds $c_1 = \mathcal{H}(\pi, H(\boldsymbol{y}+\boldsymbol{x})^\intercal + \boldsymbol{S})$, $c_2 = \mathcal{H}(\pi(\boldsymbol{y}))$ and $c_3 = \mathcal{H}(\pi(\boldsymbol{y}+\boldsymbol{x}))$. It replies to the challenge with $(\boldsymbol{y}+\boldsymbol{x}, \pi)$ if $b=1$ or $(\pi(\boldsymbol{y}), \pi(\boldsymbol{x}))$ if $b=2$. The strategy fails for $b=0$.

Overall the probability of cheating is exactly $2/3$. This means that an honest prover, in order to be accepted, needs to repeat the protocol many times. The author in [120] suggests 35 repetitions, leading to a cheating probability of $10^{-6} \approx 2^{-20}$, a weak authentication level. Still, even with this relatively small number of repetitions, communication costs per round amount to nearly 1146 bits for the original set of parameters $[512, 256, 56]$, for a total of more than 40110 bits. This results in a very long signature ($>$ 150Kb) when the scheme is instantiated in the Fiat-Shamir protocol. Moreover, the proposed parameters are susceptible to general decoding attacks and the public key is very large, as for all code-based schemes. It is easy to imagine that, with parameters secure against modern criteria (e.g. at least 2^{128} security level for general decoding attacks and a minimum authentication level of 2^{-32}), the scheme would be even less practical.

Many proposals have been designed to deal with the problem, both by reducing the communication costs and by lowering the probability of cheating. In the same paper [120] Stern describes a 5-pass protocol that replaces $\boldsymbol{s}$ with a collection of vectors $\boldsymbol{s}_1, \dots, \boldsymbol{s}_\ell$, with cheating probability equal to $\frac{1+2^{\ell-1}}{2^\ell} \approx 1/2$. Communication costs are, however, higher. Véron's scheme [125], on the other hand, reduces the communication costs slightly, but the cheating probability is still 2/3. The same holds for the scheme of Gaborit and Girault [50], even though their proposal, based on double circulant codes, is a first step to a concrete reduction in the public key size.

Recently, Cayrel, Véron and El Yousfi [24] proposed a variant that makes use of linear codes over $\mathbb{F}_q$, for $q \neq 2$. The scheme is a 5-pass protocol that relies on the hardness of the q-ary syndrome decoding problem. The cheating probability is shown to be exactly $\frac{q}{2(q-1)}$; this is reasonably close to 1/2 for big enough q. In addition, quasi-cyclic codes are used as in [50], to achieve smaller public key sizes. In the table below, we indicate with $*$ the coordinate-wise multiplication of vectors.

Table 7.11: Cayrel-Véron-El Yousfi Identification Scheme.

Public Data	The parameters $q, n, k, w \in \mathbb{N}$, an $(n-k) \times n$ parity-check matrix H over $\mathbb{F}_q$ and a hash function $\mathcal{H}$.
Private Key	$\boldsymbol{s} \in \mathbb{W}_{q,n,w}$.
Public Key	$\boldsymbol{S} = H\boldsymbol{s}^\intercal$.

PROVER		VERIFIER
Choose $\boldsymbol{y}, \boldsymbol{u} \xleftarrow{\$} \mathbb{F}_q^n$ with $\boldsymbol{u} \neq 0^n$ and a permutation $\pi \xleftarrow{\$} \mathrm{Sym}(n)$, then set $c_1 = \mathcal{H}(\pi, \boldsymbol{u}, H\boldsymbol{y}^\intercal)$ and $c_2 = \mathcal{H}(\pi(\boldsymbol{u}) * \pi(\boldsymbol{y}), \pi(\boldsymbol{u}) * \pi(\boldsymbol{s}))$.	$\xrightarrow{c_1, c_2}$	
	$\xleftarrow{\alpha}$	$\alpha \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}$.
Compute $\boldsymbol{v} = \pi(\boldsymbol{u}) * \pi(\boldsymbol{y} + \alpha\boldsymbol{s})$.	$\xrightarrow{\boldsymbol{v}}$	
	$\xleftarrow{b}$	$b \xleftarrow{\$} \{0,1\}$.
If $b = 0$ set $z = (\boldsymbol{u}, \pi)$.	$\xrightarrow{z}$	Accept if $c_1 = \mathcal{H}(\pi, \boldsymbol{u}, H(\pi^{-1}(\boldsymbol{u}) * \pi^{-1}(\boldsymbol{v}))^\intercal - \alpha\boldsymbol{S})$.
If $b = 1$ set $\boldsymbol{z} = (\pi(\boldsymbol{u}) * \pi(\boldsymbol{s}))$.	$\xrightarrow{\boldsymbol{z}}$	Accept if $c_2 = \mathcal{H}(\boldsymbol{v} - \alpha\boldsymbol{z}, \boldsymbol{z})$ and $\mathsf{wt}(\boldsymbol{z}) = w$.

Again, it is easy to check that the protocol is complete. The zero-knowledge is proved in the random oracle model, through the use of a simulator that is able to output a communication tape indistinguishable from a real prover-verifier interaction. Let's now consider an impersonator for the scheme. This would be able to cheat with either of the following procedures:

- The impersonator chooses random $\boldsymbol{y}, \boldsymbol{u}$ and π plus another string $\boldsymbol{r} \in \mathbb{W}_{q,n,w}$. It computes $\boldsymbol{x} \in \mathbb{F}_q^n$ such that $H\boldsymbol{x}^\intercal = \boldsymbol{S}$ (but without satisfying $\mathsf{wt}(\boldsymbol{x}) = w$), and guesses a value for α, say $\beta \in \mathbb{F}_q \setminus \{0\}$. It then builds c_1 normally and $c_2 = \mathcal{H}(\pi(\boldsymbol{u}) * \pi(\boldsymbol{y} + \beta\boldsymbol{x}) - \beta\boldsymbol{r}, \boldsymbol{r})$. Finally, it replies to the challenge with $(\boldsymbol{u}, \pi)$ if $b = 0$ or $\boldsymbol{r}$ if $b = 1$. The strategy succeeds if $b = 0$ or if $b = 1$ and $\alpha = \beta$.

- If $b = 1$ the impersonator chooses random $\boldsymbol{y}, \boldsymbol{u}$ and π plus another random string $\boldsymbol{x} \in \mathbb{W}_{q,n,w}$, and guesses a value for α, say $\beta \in \mathbb{F}_q \setminus \{0\}$. It then builds $c_1 = \mathcal{H}(\pi, \boldsymbol{u}, H\boldsymbol{y}^\intercal + \beta(h\boldsymbol{x}^\intercal - \boldsymbol{S}))$ and $c_2 = \mathcal{H}(\pi(\boldsymbol{u}) * \pi(\boldsymbol{y}), \pi(\boldsymbol{u}) * \pi(\boldsymbol{x}))$. Finally, it replies to the challenge with either $(\boldsymbol{u}, \pi)$ if $b = 0$ or $(\pi(\boldsymbol{u}) * \pi(\boldsymbol{x}))$ if $b = 1$. The strategy succeeds if $b = 0$ and $\alpha = \beta$ or if $b = 1$.

Overall, since the probability of guessing α is $\frac{1}{q-1}$, we have that a cheater succeeds with probability $\frac{q}{2(q-1)}$ as mentioned above.
A drawback of the scheme is that the data exchange consists now of $\mathbb{F}_q$ operations and the public matrix is defined over $\mathbb{F}_q$, so, since q needs to be chosen reasonably large, both communication costs and public key size suffer a performance loss.

In a preprint by Aguilar Melchor, Gaborit and Schrek [81], a variant of Véron's scheme is proposed, consisting of a 5-pass protocol that makes use of the double circulant construction. The paper also features a comparison of parameters for the previous schemes in the literature. We present it below.

Table 7.12: Comparison of the most popular zero-knowledge identification schemes, for the same cheating probability of 2^{-16}. All the sizes are expressed in bits and the prover's computation counts bit operations, except for CVE ($\mathbb{F}_{2^8}$-multiplications).

	Stern 3	Stern 5	Véron	CVE	AGS
Rounds	28	16	28	16	18
Public Data[2]	122500	122500	122500	32768	350
Private Key	700	4900	1050	1024	700
Public Key	350	2450	700	512	700
Total Communication Cost	42019	62272	35486	31888	20080
Prover's Computation	$2^{22.7}$	$2^{21.92}$	$2^{22.7}$	2^{16}	2^{21}

While the size of the public matrix is considerably smaller, the signature size, even if reduced to 93Kb in the AGS scheme, is still very large, and the communication costs high. Moreover, for signatures, one would expect computational costs to produce a forgery to be no less than 2^{80}; this would require, as claimed by the authors in [81], to multiply all the above data by 5. Clearly, such a scheme is completely impractical in many applications.

[2]The public matrix H.

Rank-based schemes

Classical coding theory problems rely on the well-known Hamming metric (Definitions 2.16, 2.17). An alternative is given by codes for the *rank metric*.

Definition 7.1 Let $\mathcal{C}$ be an $[n,k]$ linear code over $\mathbb{F}_{q^m}$ and let $\mathcal{B} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\}$ be a basis of $\mathbb{F}_{q^m}$ as a vector space over $\mathbb{F}_q$. Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathcal{C}$ be a codeword; we associate to $\boldsymbol{x}$ the matrix $M_{\boldsymbol{x}} = \{x_{i,j}\} \in \mathbb{F}_q^{m \times n}$ such that $x_{i,j}$ is the i-th coordinate of x_j with respect to $\mathcal{B}$. The *Rank Weight* $\mathsf{rk}(\boldsymbol{x})$ of the codeword is the rank of the associated matrix $M_{\boldsymbol{x}}$. The *Rank Distance* $\mathsf{d}_{\mathsf{rk}}(\boldsymbol{x}, \boldsymbol{y})$ between two codewords $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as $\mathsf{rk}(\boldsymbol{x} - \boldsymbol{y})$.

As we have briefly mentioned in Section 3.1.3, the rank metric has been used without much success for designing encryption schemes [47, 48]. The main problem is that there is only one family of efficiently decodable codes in the rank metric, the Gabidulin codes, for which the underlying algebraic structure reveals too much information about the private key. In principle, though, no problems of this sort should arise if the codes in use are random codes, that don't need to be decodable. This is the case for Chen's scheme [26], introduced in 1996. The protocol is the first proposal for an identification scheme that relies on the rank metric, and, unlike all other schemes, has the interesting feature of not requiring the use of a hash functions.

Table 7.13: Chen Identification Scheme.

Public Data	The parameters $q, m, n, k, w \in \mathbb{N}$ and an $(n-k) \times n$ parity-check matrix H over $\mathbb{F}_{q^m}$.
Private Key	$\boldsymbol{s} \in \mathbb{F}_{q^m}^n$ with $\mathsf{rk}(\boldsymbol{s}) \leq w$.
Public Key	$\boldsymbol{S} = H\boldsymbol{s}^\mathsf{T}$.

PROVER		VERIFIER
Choose $\boldsymbol{y} \xleftarrow{\$} \mathbb{F}_{q^m}^n$ and $P \xleftarrow{\$} \mathrm{GL}_n(\mathbb{F}_q)$, then set $c_1 = HP^\mathsf{T}\boldsymbol{y}^\mathsf{T}$ and $c_2 = H\boldsymbol{y}^\mathsf{T}$.	$\xrightarrow{c_1, c_2}$	
	$\xleftarrow{\alpha}$	$\alpha \xleftarrow{\$} \mathbb{F}_{q^m} \setminus \{0\}$.
Compute $\boldsymbol{v} = \boldsymbol{y} + \alpha \boldsymbol{s} P^{-1}$.	$\xrightarrow{\boldsymbol{v}}$	
	$\xleftarrow{b}$	$b \xleftarrow{\$} \{0,1\}$.
If $b = 0$ set $z = P$.	$\xrightarrow{z}$	Accept if $Hz^\mathsf{T}\boldsymbol{v}^\mathsf{T} = c_1 + \alpha\boldsymbol{S}$.
If $b = 1$ set $\boldsymbol{z} = \boldsymbol{y}$.	$\xrightarrow{\boldsymbol{z}}$	Accept if $H\boldsymbol{z}^\mathsf{T} = c_2$ and[3] $\mathsf{rk}(\boldsymbol{v} - \boldsymbol{z}) = w$.

The security comes from a version of the Syndrome Decoding problem based for the rank metric. Unlike its Hamming metric correspondent, this is not proven

[3]The original scheme includes the possibility of α being 0, in which case the check would become $\mathsf{rk}(\boldsymbol{v} - \boldsymbol{z}) = 0$.

to be NP-hard, but only believed to be hard in general. It is also possible to formulate a rank metric version of the GV bound (Definition 2.35); just like for codes in Hamming metric, Loidreau [71] shows that random rank metric codes lie on the bound with high probability. The parameter w is then chosen to be less or equal to $d/3$, where d is the minimum (rank) distance of the chosen code. With an argument similar to the CVE scheme, it is possible to prove that cheating probability is exactly $\frac{q^m+1}{2q^m} \approx 1/2$.
An interesting feature is that rank metric codes are in general much harder to decode, so general decoding attacks (Chabaud and Stern [25], Ourivski and Johansson [92]) have higher complexity. This means that even very small codes may provide sufficient security: the original parameters proposed in [26] are $q = 2$, $n = 32, k = m = 16, w = 4$.

The scheme is complete since the invertible matrix P is an invariant for the rank metric (Berger, [10]). Unfortunately, the action of P alone is not enough to map a word $\boldsymbol{x}$ of a given rank to any other word of the same rank (unlike the case of permutation matrices in the Hamming metric). This is because P has elements in $\mathbb{F}_q$, hence does not change the basis generated by the coordinates of $\boldsymbol{x}$. As we already mentioned, this is a dangerous choice: Gaborit, Schrek and Zémor in [51] describe an attack that exploits this flaw and allows to recover $\boldsymbol{s}$ fully. In fact, since $\boldsymbol{s}$ and $\boldsymbol{s}P^{-1}$ generate the same vector space, it is enough to compute $\boldsymbol{s}P^{-1}$ as $\alpha^{-1}(\boldsymbol{v} - \boldsymbol{y})$ and choose an arbitrary basis for the vector space generated by it, then solve the system of equations given by $\boldsymbol{S} = H\boldsymbol{s}^{\intercal}$ with coordinates in this basis. The system has nw unknowns and $(n-k)m$ equations, which is directly solvable for any practical choice of parameters.
Another attack is also presented in [51], and takes advantage of the fact that hash functions are not used in the protocol, so it is easier to extract information from the exchange of data. In particular, when $b = 0$, the only unknown in the expression $H\boldsymbol{v}^{\intercal} = c_2 + \alpha H(\boldsymbol{s}P^{-1})^{\intercal}$ is $\boldsymbol{s}$ (since P is revealed), hence the expression provides additional equations that, together with $\boldsymbol{S} = H\boldsymbol{s}^{\intercal}$, allow to recover $\boldsymbol{s}$ with only a few repetitions.

The authors also propose a "fix" of the scheme. The first issue is addressed by including a left multiplication by a matrix $Q \in \mathrm{GL}_m(\mathbb{F}_q)$: this is also an invariant for the metric, but, together with P, succeeds in changing the basis generated by the coordinates of $\boldsymbol{s}$ while preserving the same rank. The second issue is instead addressed by replacing the commitments with hash values, just like in Stern's scheme. In fact, the new protocol is essentially a translation of Stern's scheme to the rank metric case, and the cheating probability is again 2/3. The authors suggest the following choice of parameters for their scheme: $q = 2$, $n = m = 20, k = 11, w = 6$. With these values, the public matrix H has size 1980 bits and the public key is only 180 bits long. The number of bits exchanged in a single round is approximately 800, resulting in a total communication cost (for 28 rounds as in Stern's scheme) of 22400 bits. Overall, the performance of the scheme is similar to the AGS proposal [81], suggesting that rank metric could be an interesting research direction.

7.3 An Alternative Approach for Signatures

7.3.1 Number Theory and Lattices

There is an easy way to construct efficient signature schemes based on classical number theory problems, such as of factoring or computing discrete logarithm. The approach consists of successive reductions building on the original hard problem, in the following way. A collision-resistant hash function (see Definition 2.7) can be derived directly from the hard problem: for example, finding collisions for the function $\mathcal{H}(x) = g^x \pmod N$, where N is an RSA modulus, implies being able to factor N. The hash function can be converted into a one-time signature where the private key is a pair of integers (x, y), the public key is the pair $(\mathcal{H}(x), \mathcal{H}(y))$, and the signature of a message c is simply $cx + y$. The one-time signature can then be converted into a zero-knowledge identification scheme as described below: c is a challenge chosen by the verifer and y is the commitment (a distinct y is used in every run of the protocol). Finally, the identification scheme can be converted to a signature scheme by using the Fiat-Shamir transform as described in Table 7.9. A similar scheme can be instantiated with a hash function based on discrete logarithm, such as $\mathcal{H}(x_1, x_2) = g_1^{x_1} g_2^{x_2} \pmod p$. See Okamoto [90] for more details.

Table 7.14: Factoring-based Identification Scheme.

Public Data	An RSA modulus $N = pq$ and a group element g.
Private Key	$s \in D_s$.
Public Key	$S = g^s \pmod N$.

PROVER		VERIFIER
Choose $y \xleftarrow{\$} D_y$ and set $Y = g^y \pmod N$.	$\xrightarrow{Y}$	
	$\xleftarrow{c}$	$c \xleftarrow{\$} D_c$.
Compute $z = y + cs$.	$\xrightarrow{z}$	Accept if $g^z = YS^c \pmod N$.

Lyubashevsky in [74] showed how to translate the framework to the lattice case. The translation is direct, except for an issue which is inherent to the nature of the lattice schemes: unlike factoring or discrete logarithm, in fact, the hardness of lattice problems comes from finding elements that live in a specific *subset* of a ring, namely elements with small Euclidean norm. This results in a leakage of some parts of the private key. To overcome this limitation, Lyubashevsky makes use of a technique, already introduced in [73], called *aborting*. Briefly, this consists of refusing to answer to the challenge if in doing so the security of the scheme would be compromised. In practice, this is realized by limiting the set of possible answers to a smaller "safe" subset, consisting of elements whose norm satisfies a certain bound. Details are given below. The hash functions in this case are sampled from the family $\mathbb{H}(R, D, m) = \{\mathcal{H}_{\boldsymbol{a}} : \boldsymbol{a} \in R^m\}$ where R is the ring $\mathbb{Z}_p[x]/(x^n + 1)$, $D \subseteq R$ and, for every $\boldsymbol{z} \in D^m$, we define $\mathcal{H}_{\boldsymbol{a}}(\boldsymbol{z}) = \boldsymbol{a} \cdot \boldsymbol{z}$.

Table 7.15: Lattice-based Identification Scheme.

Public Data	A hash function $\mathcal{H} \xleftarrow{\$} \mathbb{H}(R, D, m)$.
Private Key	$\boldsymbol{s} \in D_s^m$.
Public Key	$\boldsymbol{S} = \mathcal{H}(\boldsymbol{s})$.

PROVER		VERIFIER
Choose $\boldsymbol{y} \xleftarrow{\$} D_y^m$ and set $\boldsymbol{Y} = \mathcal{H}(\boldsymbol{y})$.	$\xrightarrow{\boldsymbol{Y}}$	
	$\xleftarrow{\boldsymbol{c}}$	$\boldsymbol{c} \xleftarrow{\$} D_c$.
Compute $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{cs}$. If $\boldsymbol{z} \notin G^m$ set $\boldsymbol{z} = \perp$.	$\xrightarrow{\boldsymbol{z}}$	Accept if $\boldsymbol{z} \in G^m$ and $\mathcal{H}(\boldsymbol{z}) = \boldsymbol{Y} + \boldsymbol{cS}$.

The subset G is exactly the "safe" subset described above. To further clarify the scheme, we present below the proposed parameters for the scheme ([74, Fig. 2]). In the following table, we denote with $||\cdot||_\infty$ the usual ℓ_∞ norm for vectors, that is $||\boldsymbol{x}||_\infty = \max_i(|x_i|)$.

Table 7.16: Parameter Definitions and Sample Instantiations.

Parameter	Definition	Sample Instantiations			
n	a power of 2	512	512	512	1024
m	any integer	4	5	8	8
σ	any integer	127	2047	2047	2047
κ	integer such that $2^\kappa\binom{n}{\kappa} \geq 2^{160}$	24	24	24	21
p	integer $\approx (2\sigma+1)^m \cdot 2^{-\frac{128}{n}}$	$2^{31.7}$	$2^{59.8}$	$2^{95.8}$	$2^{95.9}$
R	the ring $\mathbb{Z}_p[x]/(x^n+1)$				
D	$\{\boldsymbol{g} \in R : \|\|\boldsymbol{g}\|\|_\infty \leq mn\sigma\kappa\}$				
D_s	$\{\boldsymbol{g} \in R : \|\|\boldsymbol{g}\|\|_\infty \leq \sigma\}$				
D_c	$\{\boldsymbol{g} \in R : \|\|\boldsymbol{g}\|\|_\infty \leq \kappa\}$				
D_y	$\{\boldsymbol{g} \in R : \|\|\boldsymbol{g}\|\|_\infty \leq mn\sigma\kappa\}$				
G	$\{\boldsymbol{g} \in R : \|\|\boldsymbol{g}\|\|_\infty \leq mn\sigma\kappa - \sigma\kappa\}$				
Signature	$\approx mn\log 2mn\sigma\kappa$ bits	49000	72000	119000	246000
Public Key	$\approx n\log p$ bits	16000	31000	49000	98000
Private Key	$\approx mn\log 2\sigma + 1$ bits	16000	31000	49000	98000
Hash	$\approx mn\log p$ bits	65000	153000	392000	786000
Length of vector needed to break signature		$2^{23.5}$	$2^{27.9}$	$2^{28.6}$	$2^{29.4}$
Length of shortest vector that can be found		$2^{25.5}$	$2^{36.7}$	$2^{47.6}$	$2^{69.4}$

The last two lines refer to cryptanalytic parameters which are specific for lattice cryptography (such as the LLL algorithm [67]).

Recently, Galbraith and Dwarakanath [52] showed that, despite its simplicity and theoretical elegance, the scheme presents some implementation difficulties, at least for constrained devices. Namely, sampling objects from Gaussian distributions with very large standard deviation is not trivial, and rejection sampling doesn't seem applicable in practice; moreover, the signature size is very large.

7.3.2 A Coding Theory Scenario

We now discuss the possibility of creating an identification scheme following the simple framework described in the previous section; we will show that such a construction is not feasible.
One could think to translate the framework directly, that is, following the paradigm described in Table 7.7. This would be based on Syndrome Decoding, hence featuring a public matrix H, a secret $\boldsymbol{s}$ and the public key $\boldsymbol{S} = H\boldsymbol{s}^{\mathsf{T}}$. The scheme would need to satisfy precise requirements:

- The secret $\boldsymbol{s}$ should have weight below the GV bound. This is to ensure that the secret is unique and that SD is hard.
- The final verification should include an algebraic formula consisting of H, the commitment $\boldsymbol{Y}$ and $\boldsymbol{S}$, plus a check on the weight of the response $\boldsymbol{z}$.
- The challenge c should be such that $c\boldsymbol{s}$ is defined, $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s}$ is defined and c is compatible with the final verification[4].

By analogy with the previous part of this section, we could think about the syndrome computation as a hash function $\mathcal{H}(\boldsymbol{x}) = H\boldsymbol{x}^{\mathsf{T}}$. Of course, this is preimage-resistant only if the weight of $\boldsymbol{x}$ is small. It follows that the scheme is subject to additional constraints. For example, the random element $\boldsymbol{y}$ and the challenge c should be chosen such that $\mathsf{wt}(\boldsymbol{z}) \leq w$, where w is the value of the GV distance. A natural choice for c is to be an element of $\mathbb{F}_q$. Since multiplication by a field element is an invariant for the Hamming weight, this means that $\boldsymbol{s}$ and $\boldsymbol{y}$ must satisfy $\mathsf{wt}(\boldsymbol{s}) = \gamma_1 w, \mathsf{wt}(\boldsymbol{y}) = \gamma_2 w$, for certain constants $\gamma_1, \gamma_2 \leq 1$ such that $\gamma_1 + \gamma_2 = 1$.
A sample instantiation is described below (for the case $\gamma_1 = \gamma_2 = 1/2$).

Table 7.17: Syndrome-based Identification Scheme.

Public Data	The parameters $q, n, k, w \in \mathbb{N}$ and an $(n-k) \times n$ parity-check matrix H over $\mathbb{F}_q$.
Private Key	$\boldsymbol{s} \in \mathbb{W}_{q,n,w/2}$.
Public Key	$\boldsymbol{S} = H\boldsymbol{s}^{\mathsf{T}}$.

PROVER		VERIFIER
Choose $\boldsymbol{y} \xleftarrow{\$} \mathbb{W}_{q,n,w/2}$ and set $\boldsymbol{Y} = H\boldsymbol{y}^{\mathsf{T}}$.	$\xrightarrow{\boldsymbol{Y}}$	
	$\xleftarrow{c}$	$c \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}$.
Compute $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s}$.	$\xrightarrow{\boldsymbol{z}}$	Accept if $H\boldsymbol{z}^{\mathsf{T}} = \boldsymbol{Y} + c\boldsymbol{S}$ and $\mathsf{wt}(\boldsymbol{z}) \leq w$.

The protocol is complete and well-defined. However, the conditions on the weight of $\boldsymbol{s}, \boldsymbol{y}$ and $\boldsymbol{z}$ make the scheme vulnerable to an attacker who tries to learn the secret. We will see how in the next theorem.

[4]This means $Hc = c'H$ for some c' not necessarily equal to c.

Theorem 7.1 *Any identification scheme that satisfies the above requirements cannot be a zero-knowledge protocol.*

Proof We build an attacker $\mathcal{A}$ that will compute the private key. $\mathcal{A}$ is a passive attacker, that is, $\mathcal{A}$ has only evidence of the exchanges between $\mathcal{P}$ and $\mathcal{V}$: this is the weakest possible adversarial model. To recover the secret information $\boldsymbol{s}$, $\mathcal{A}$ runs the protocol several times. At every run of the protocol, it stores the challenge c and the corresponding response $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s}$, then computes $\boldsymbol{z}' = c^{-1}\boldsymbol{y} + \boldsymbol{s}$. Note that this is always possible, since c is a field element and is non-zero. Without loss of generality, we can consider $\boldsymbol{z}'$ to be of the form $\boldsymbol{y}' + \boldsymbol{s}$. If $\boldsymbol{y}$ is randomly generated, so is $\boldsymbol{y}'$; moreover, $\mathsf{wt}(\boldsymbol{y}') = \mathsf{wt}(\boldsymbol{y}) = w/2 << n/2$ by construction, so each of the coordinates $\boldsymbol{y}'_i$ is biased to be more likely 0 than non-zero. Therefore, $\mathcal{A}$ can perform successfully a very simple statistical analysis: it fixes a particular i and observes the behavior of $\boldsymbol{y}'_i$ for multiple runs. If $\boldsymbol{y}'_i$ is non-zero most of the times, then $i \in \mathrm{supp}(\boldsymbol{s})$. Eventually, $\mathcal{A}$ is able to fully recover the support of $\boldsymbol{s}$. This completes the attack in the binary case, and gives enough information to recover $\boldsymbol{s}$ even in the non-binary case (for example with a general decoding algorithm). $\triangle$

The crucial point is that $\boldsymbol{y}$ is constrained to be of small weight, hence the expression $\boldsymbol{y} + c\boldsymbol{s}$ is not enough to properly hide the support of $\boldsymbol{s}$. This clashes with the other security requirement, that is, to avoid forgeries. If one drops the condition on $\mathsf{wt}(\boldsymbol{z})$, in fact, it is easy to find a vector that satisfies the verification equation. We conclude that it is infeasible to create a scheme in such a direct way.

Other more elaborate proposals involving permutation matrices and hash functions have been analyzed and are still vulnerable. The issue seems to come from the Hamming metric, which is too constraining to be able to hide the secret. Unlike the lattice case, in fact, vectors in the Hamming metric are measured on a position-dependent basis rather than on Euclidean norm. This seems to be leaking too much information to provide zero-knowledge (unless using a Stern-like protocol).
A natural approach would then be to investigate other metrics. We have seen in the previous section that rank metric schemes have been proposed in the literature. It is possible to imagine a rank-based scheme following the above guidelines, by simply replacing the Hamming weight with rank weight and q with q^m for a certain $m > 1$. Note that multiplication by $c \in \mathbb{F}_{q^m}$ is an invariant for the rank weight. The change of metric would have the advantage that now a vector is no longer measured by the number of its non-zero positions but rather by the dimension of the vector space generated by its coordinates when seen as $\mathbb{F}_q$-vectors. It follows that a statistical analysis as above would not be applicable in this setting. A further advantage would be the small size of the codes in use, resulting in a very practical scheme. Unfortunately, rank metric schemes are vulnerable to another kind of threat, related to basis of vector spaces, as we have already seen for Chen's identification scheme (Table 7.13). It is then possible to prove a theorem analogous to Theorem 7.1 for the rank metric case.

Theorem 7.2 *Any rank metric identification scheme that satisfies the above requirements cannot be a zero-knowledge protocol.*

Proof We build a passive zero-knowledge attacker $\mathcal{A}$. Suppose that the response $\boldsymbol{z}$ is such that $\mathsf{rk}(\boldsymbol{z}) = \mathsf{rk}(\boldsymbol{y}) + \mathsf{rk}(\boldsymbol{s})$. Then the vector space $V_{\boldsymbol{z}}$ generated by the columns of $\boldsymbol{z}$ (seen as a matrix over $\mathbb{F}_q$) will coincide with the space generated by the union of the columns of $\boldsymbol{y}$ and $c\boldsymbol{s}$, that is

$$V_{\boldsymbol{z}} = \operatorname{span}(\{\boldsymbol{z}_1, \dots, \boldsymbol{z}_n\}) = \operatorname{span}(\{\boldsymbol{y}_1, \dots, \boldsymbol{y}_n\} \cup \{c\boldsymbol{s}_1, \dots, c\boldsymbol{s}_n\}). \tag{7.4}$$

To recover $\boldsymbol{s}$, $\mathcal{A}$ runs the protocol several times and proceeds as before, collecting t samples $\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(t)}$ of the form $\boldsymbol{y}' + \boldsymbol{s}$ that satisfy the above property. Next, $\mathcal{A}$ intersects the corresponding vector spaces $V_{\boldsymbol{z}^{(i)}}$. Even for a very small t, we will have with overwhelming probability

$$\bigcap_{i=1}^{t} V_{\boldsymbol{z}^{(i)}} = \operatorname{span}(\{\boldsymbol{s}_1, \dots, \boldsymbol{s}_n\}). \tag{7.5}$$

We therefore assume that $\operatorname{span}(\{\boldsymbol{s}_1, \dots, \boldsymbol{s}_n\})$ is efficiently computable. Once $\mathcal{A}$ has evidence of this space, it is enough to fix an arbitrary basis for it and solve the system of equations given by $\boldsymbol{S} = H\boldsymbol{s}^{\mathsf{T}}$ with respect to this basis. The system has $nw/2$ unknowns and $(n-k)m$ equations, hence $\mathcal{A}$ is able to recover $\boldsymbol{s}$ for all practical choice of parameters. This concludes the attack. $\triangle$

Just like for the Hamming case, the main issue is the clash between two opposite requirements. If $\boldsymbol{y}$ is of small rank, the vector $\boldsymbol{z}$ leaks information about the vector space generated by the columns of $\boldsymbol{s}$; if not, we lose the condition on $\mathsf{rk}(\boldsymbol{z})$ and the scheme becomes vulnerable to impersonation. We conclude that it is infeasible to create a scheme in such a direct way.

7.4 Conclusions

In the first part of this chapter, we have thoroughly reviewed the literature about code-based signature schemes. None of the current protocols is completely satisfying, because of very large keys and signatures, slow signing algorithms or security issues. This makes code-based schemes a very impractical choice for many applications. The work of Lyubashevsky [74] suggests a potential new direction for designing efficient signature schemes. We show in Section 7.3.2 that this approach fails if translated directly to the case of coding theory; the syndrome decoding problem is in fact too constraining both for the Hamming metric and the rank metric. We conclude that the design of an efficient signature schemes based on coding theory remains an open problem.

Conclusions and Future Work

Code-based cryptography is one of the more accredited candidates for public-key cryptography in a “post-quantum” scenario. So far, though, some flaws have prevented its use in many applications, in particular the huge size of the public key. In this thesis, we have thoroughly investigated the area of code-based cryptography, with the aim of addressing its main issues. The contributions of this thesis can be summarized as follows:

Public Key Size

In Chapter 4, we present a proposal for a variant of the McEliece cryptosystem. This is an individual work by the author, and follows on from the proposal by Misoczki and Barreto based on quasi-dyadic codes [85]. The main novelty of the scheme is replacing Goppa codes with Generalized Srivastava codes; to the best of our knowledge, this is the first time that this family of codes is proposed in cryptography. The particular structure of the family brings numerous benefits: in particular, the extra parameter t allows us to modulate the construction in a much more flexible way. It is in fact possible to use codes over relatively small extension fields without losing in security. Moreover, t quantifies trade-offs both for security (the ratio (extension degree)/(number of free variables) is crucial for the FOPT attack of [38]), and for reduction in the public key size. This results in a flexible and practical scheme which produces very small keys and resists all the attacks presented so far.
The practicality of the scheme is highlighted in a subsequent work by Cayrel, Hoffmann and the author, which is presented in the second part of Chapter 4. An implementation of the scheme is provided, both for C++ language and for an embedded device. The timings reported show how the scheme is more suitable for a practical implementation than its Goppa codes-based counterpart (implemented by Heyse in [58]): for example, the log/antilog tables for the finite field used by our construction fit completely in the flash memory of the device, hence there is no need for external memory and tower field arithmetic. It is also evident that adding a CCA2 conversion such as the Fujisaki-Okamoto transform [46] results in a rather small computational overhead. The transform is implemented thanks to a simple tweak, consisting of exchanging the role of the message and the randomness in the McEliece encryption process, which is an original contribution. Finally, the choice of the Keccak for both our hash functions and as a random number generator is certainly a major advantage since this has recently been chosen as the new SHA-3.

IND-CCA2 Security

IND-CCA2 is rightly considered the most desirable security property for public-key encryption schemes. While the original McEliece and Niederreiter schemes are clearly not secure in this sense, there are a number of possible ways to obtain IND-CCA2 security for code-based schemes. In the random oracle model, a very general one is represented by the Fujisaki-Okamoto transform that we just mentioned; there exist also other, more specific transforms such as

the one by Kobara and Imai [64]. In Chapter 5, we propose a new construction, based on the KEM-DEM paradigm for hybrid encryption. A KEM based on the Niederreiter scheme is described, and its security is rigorously proved. The work stems from the RSA-KEM of Cramer [115] and a suggestion by Bernstein, and the main contribution is a slight modification of the original KEM structure, in order to take care of non-decryptable ciphertexts without leaking information.
In the standard model, we carefully analyze (Chapter 6) a work by Dowsley, Müller-Quade and Nascimento [36], in which the authors try to apply the general framework of correlated products presented by Rosen and Segev in [106]. In the paper there are several inaccuracies that lead to an ambiguous description of the proposed scheme. All of these are addressed and corrected, and a more rigorous description is presented together with a proposal for applying the original Rosen-Segev framework in the coding theory environment.

Signatures

Digital signatures are arguably one of the most important cryptographic primitives in modern society. Unfortunately, unlike the case of encryption schemes, coding theory schemes have so far failed to provide an efficient solution for digital signatures. In Chapter 7 we present a very accurate literature review, featuring the main proposals for code-based signature schemes. All of the schemes presented over the years end up being either insecure or highly inefficient, due to large public keys, long signatures and slow signing algorithms. We discuss some recent proposals for lattice signatures, based on zero-knowledge identification schemes. We show that it is impossible to translate this approach directly to the coding theory scenario. This is mainly due to the strongly constraining nature of the syndrome decoding problem, both for the Hamming metric and the rank metric. Thus, the design of an efficient code-based signature scheme is still an open problem.

Future Work

It is clear that much work is still needed to truly allow code-based cryptography to be considered for practical applications. For example, it would be very helpful to have a precise assessment of the security of the quasi-dyadic schemes. In particular, the FOPT attack doesn't have an accurate complexity analysis (only partially addressed in [39]).
As for IND-CCA2 security, a few promising recent results have been published, for instance by Preetha Mathew, Vasant, Venkatesan and Pandu Rangan [78] at ACISP 2012. We also plan to publish our work on the Niederreiter KEM, together with an assessment of other interesting properties such as anonymity.
Most of all, future work needs to carefully consider the current state of code-base signature schemes. In this case, it seems harder to reach a satisfactory conclusion. Recent results like the work of Preetha Mathew, Vasant and Pandu Rangan on CFS [77], while undoubtedly constituting an improvement, still fail to address the main issues of the scheme, in this case the very slow signing algorithm.

Bibliography

[1] M. Baldi, F. Chiaraluce, R. Garello, and F. Mininni. Quasi-Cyclic Low-Density Parity-Check Codes in the McEliece Cryptosystem. In *ICC*, pages 951–956. IEEE, 2007.

[2] P. S. L. M. Barreto, P.-L. Cayrel, R. Misoczki, and R. Niebuhr. Quasi-Dyadic CFS Signatures. In X. Lai, M.i Yung, and D. Lin, editors, *Inscrypt*, volume 6584 of *Lecture Notes in Computer Science*, pages 336–349. Springer, 2010.

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

[4] A. Becker, A. Joux, A. May, and A. Meurer. Decoding Random Binary Linear Codes in $2^{n/20}$: How $1 + 1 = 0$ Improves Information Set Decoding. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 520–536. Springer, 2012.

[5] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *FOCS*, pages 394–403. IEEE Computer Society, 1997.

[6] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In H. Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998.

[7] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, editors, *ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[8] M. Bellare and P. Rogaway. The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. In U. M. Maurer, editor, *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer, 1996.

[9] M. Bellare and P. Rogaway. Introduction to Modern Cryptography. In *UCSD CSE 207 Course Notes*, 2005.

[10] T. P. Berger. Isometries for rank distance and permutation group of Gabidulin codes. *IEEE Transactions on Information Theory*, 49(11):3016–3019, 2003.

[11] T. P. Berger, P.-L. Cayrel, P. Gaborit, and A. Otmani. Reducing Key Length of the McEliece Cryptosystem. In B. Preneel, editor, *AFRICACRYPT*, volume 5580 of *Lecture Notes in Computer Science*, pages 77–97. Springer, 2009.

[12] E. Berlekamp. Nonbinary BCH decoding. *IEEE Transactions on Information Theory*, 14(2):242, march 1968.

[13] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384 – 386, may 1978.

[14] D. J. Bernstein. Personal communication, May 2012.

[15] D. J. Bernstein. Grover vs. McEliece. In N. Sendrier, editor, *PQCrypto*, volume 6061 of *Lecture Notes in Computer Science*, pages 73–80. Springer, 2010.

[16] D. J. Bernstein, T. Lange, and C. Peters. Attacking and Defending the McEliece Cryptosystem. In J. Buchmann and J. Ding, editors, *PQCrypto*, volume 5299 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2008.

[17] D. J. Bernstein, T. Lange, and C. Peters. Smaller Decoding Exponents: Ball-Collision Decoding. In P. Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 743–760. Springer, 2011.

[18] D. J. Bernstein, T. Lange, C. Peters, and H. C. A. van Tilborg. Explicit bounds for generic decoding algorithms for code-based cryptography. In A. Kholosha, E. Rosnes, and M. Parker, editors, *Pre-proceedings of WCC 2009*, pages 168–180, Bergen, 2009.

[19] B. Biswas and N. Sendrier. McEliece Cryptosystem Implementation: Theory and Practice. In J. Buchmann and J. Ding, editors, *PQCrypto*, volume 5299 of *Lecture Notes in Computer Science*, pages 47–62. Springer, 2008.

[20] R. C. Bose and D. K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68–79, 1960.

[21] A. Canteaut and F. Chabaud. A New Algorithm for Finding Minimum-Weight Words in a Linear Code: Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511. *IEEE Transactions on Information Theory*, 44(1):367–378, 1998.

[22] P.-L. Cayrel, G. Hoffmann, and E. Persichetti. Efficient Implementation of a CCA2-Secure Variant of McEliece Using Generalized Srivastava Codes. In M. Fischlin, J. Buchmann, and M. Manulis, editors, *Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 138–155. Springer, 2012.

[23] P.-L. Cayrel, A. Otmani, and D. Vergnaud. On Kabatianskii-Krouk-Smeets Signatures. In C. Carlet and B. Sunar, editors, *WAIFI*, volume 4547 of *Lecture Notes in Computer Science*, pages 237–251. Springer, 2007.

[24] P.-L. Cayrel, P. Véron, and S. M. El Yousfi Alaoui. A Zero-Knowledge Identification Scheme Based on the q-ary Syndrome Decoding Problem. In A. Biryukov, G. Gong, and D. R. Stinson, editors, *Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Computer Science*, pages 171–186. Springer, 2010.

[25] F. Chabaud and J. Stern. The Cryptographic Security of the Syndrome Decoding Problem for Rank Distance Codes. In K. Kim and T. Matsumoto, editors, *ASIACRYPT*, volume 1163 of *Lecture Notes in Computer Science*, pages 368–381. Springer, 1996.

[26] K. Chen. A New Identification Algorithm. In E. Dawson and J. D. Golic, editors, *Cryptography: Policy and Algorithms*, volume 1029 of *Lecture Notes in Computer Science*, pages 244–249. Springer, 1995.

[27] R. Chien. Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghem codes. *IEEE Transactions on Information Theory*, 10(4):357 – 363, October 1964.

[28] J.-S. Coron. On the Exact Security of Full Domain Hash. In M. Bellare, editor, *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer, 2000.

[29] N. Courtois, M. Finiasz, and N. Sendrier. How to Achieve a McEliece-Based Digital Signature Scheme. In C. Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 157–174. Springer, 2001.

[30] R. Cramer and V. Shoup. Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. *SIAM J. Comput.*, 33(1):167–226, January 2004.

[31] J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard.* Springer, 2002.

[32] L. Dallot. Towards a Concrete Security Proof of Courtois, Finiasz and Sendrier Signature Scheme. In S. Lucks, A.-R. Sadeghi, and C. Wolf, editors, *WEWoRC*, volume 4945 of *Lecture Notes in Computer Science*, pages 65–77. Springer, 2007.

[33] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[34] D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography (Extended Abstract). In C. Koutsougeras and J. Scott Vitter, editors, *STOC*, pages 542–552. ACM, 1991.

[35] D. Dolev, C. Dwork, and M. Naor. Nonmalleable Cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

[36] R. Dowsley, J. Müller-Quade, and A. C. A. Nascimento. A CCA2 Secure Public Key Encryption Scheme Based on the McEliece Assumptions in the Standard Model. In M. Fischlin, editor, *CT-RSA*, volume 5473 of *Lecture Notes in Computer Science*, pages 240–251. Springer, 2009.

[37] J.-C. Faugère, V. Gauthier-Umaña, A. Otmani, L. Perret, and J.-P. Tillich. A distinguisher for high rate McEliece cryptosystems. In *Information Theory Workshop (ITW), 2011 IEEE*, pages 282 –286, oct. 2011.

[38] J.-C. Faugère, A. Otmani, L. Perret, and J.-P. Tillich. Algebraic Cryptanalysis of McEliece Variants with Compact Keys. In H. Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 279–298. Springer, 2010.

[39] J.-C. Faugère, A. Otmani, L. Perret, and J.-P. Tillich. Algebraic Cryptanalysis of McEliece Variants with Compact Keys – Towards a Complexity Analysis. In *SCC '10: Proceedings of the 2nd International Conference on Symbolic Computation and Cryptography*, pages 45–55, RHUL, June 2010.

[40] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1:77–94, 1988.

[41] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In A. M. Odlyzko, editor, *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[42] M. Finiasz and N. Sendrier. Security Bounds for the Design of Code-Based Cryptosystems. In M. Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 88–105. Springer, 2009.

[43] J.-B. Fischer and J. Stern. An Efficient Pseudo-Random Generator Provably as Secure as Syndrome Decoding. In U. M. Maurer, editor, *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 245–255. Springer, 1996.

[44] E. Fleischmann, C. Forler, and M. Gorski. Classification of the SHA-3 Candidates. *http://drops.dagstuhl.de/volltexte/2009/1948/pdf /09031.Forler-Christian.Paper.1948.pdf.*

[45] D. Mandell Freeman, O. Goldreich, E. Kiltz, A. Rosen, and G. Segev. More Constructions of Lossy and Correlation-Secure Trapdoor Functions. In P. Q. Nguyen and D. Pointcheval, editors, *Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 279–295. Springer, 2010.

[46] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In M. J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.

[47] E. M. Gabidulin, A. V. Ourivski, B. Honary, and B. Ammar. Reducible rank codes and their applications to cryptography. *IEEE Transactions on Information Theory*, 49(12):3289–3293, 2003.

[48] E. M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov. Ideals over a Non-Commutative Ring and thier Applications in Cryptology. In D. W. Davies, editor, *EUROCRYPT*, volume 547 of *Lecture Notes in Computer Science*, pages 482–489. Springer, 1991.

[49] P. Gaborit. Shorter keys for code-based cryptography. In *Proceedings of Workshop on Codes and Cryptography, WCC 2005*, pages 81–90, France, 2005.

[50] P. Gaborit and M. Girault. Lightweight code-based identification and signature. In *IEEE International Symposium on Information Theory, 2007. ISIT 2007*, pages 191 –195, June 2007.

[51] P. Gaborit, J. Schrek, and G. Zémor. Full Cryptanalysis of the Chen Identification Protocol. In B.-Y. Yang, editor, *PQCrypto*, volume 7071 of *Lecture Notes in Computer Science*, pages 35–50. Springer, 2011.

[52] S. D. Galbraith and N. C. Dwarakanath. Efficient sampling from discrete Gaussians for lattice-based cryptography on a constrained device. *Preprint*, 2012.

[53] S. Goldwasser and S. Micali. Probabilistic Encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[54] S. Goldwasser, S. Micali, and R. L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[55] V. D. Goppa. A new class of linear correcting codes. *Problemy Peredači Informacii*, 6(3):24–30, 1970.

[56] V. D. Goppa. Algebraico-Geometric Codes. *Izvestiya: Mathematics*, 21:75–91, February 1983.

[57] H. J. Helgert. Srivastava codes. *IEEE Transactions on Information Theory*, 18(2):292 – 297, March 1972.

[58] S. Heyse. Implementation of McEliece Based on Quasi-dyadic Goppa Codes for Embedded Devices. In B.-Y. Yang, editor, *PQCrypto*, volume 7071 of *Lecture Notes in Computer Science*, pages 143–162. Springer, 2011.

[59] A. Hocquenghem. Codes correcteurs d'erreurs. *Chiffres*, 2:147–156, 1959.

[60] J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: A Ring-Based Public Key Cryptosystem. In J. Buhler, editor, *ANTS*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.

[61] H. Janwa and O. Moreno. McEliece Public Key Cryptosystems Using Algebraic-Geometric Codes. *Des. Codes Cryptography*, 8(3):293–307, 1996.

[62] G. Kabatianskii, E. Krouk, and B. J. M. Smeets. A Digital Signature Scheme Based on Random Error-Correcting Codes. In M. Darnell, editor, *IMA Int. Conf.*, volume 1355 of *Lecture Notes in Computer Science*, pages 161–167. Springer, 1997.

[63] T. Kasami, S. Lin, and W. Peterson. New generalizations of the Reed-Muller codes–I: Primitive codes. *IEEE Transactions on Information Theory*, 14(2):189 – 199, mar 1968.

[64] 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, 2001.

[65] L. Lamport. Constructing digital signatures from a one-way function. Technical report, October 1979.

[66] P. J. Lee and E. F. Brickell. An Observation on the Security of McEliece's Public-Key Cryptosystem. In C. G. Günther, editor, *EUROCRYPT*, volume 330 of *Lecture Notes in Computer Science*, pages 275–280. Springer, 1988.

[67] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[68] J. S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, 34(5):1354–1359, 1988.

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

[70] G. Locke and P. Gallagher. FIPS PUB 186-3: Digital Signature Standard (DSS). *National Institute of Standards and Technology*, 2009.

[71] P. Loidreau. Properties of codes in rank metric. In *Eleventh International Workshop on Algebraic and Combinatorial Coding Theory ACCT2008*, Pamporovo, Bulgarie, June 2008.

[72] P. Loidreau and N. Sendrier. Weak keys in the McEliece public-key cryptosystem. *IEEE Transactions on Information Theory*, 47(3):1207–1211, 2001.

[73] V. Lyubashevsky. Lattice-Based Identification Schemes Secure Under Active Attacks. In R. Cramer, editor, *Public Key Cryptography*, volume 4939 of *Lecture Notes in Computer Science*, pages 162–179. Springer, 2008.

[74] V. Lyubashevsky. Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures. In M. Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616. Springer, 2009.

[75] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*, volume 16. North-Holland Mathematical Library, 1977.

[76] J. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15(1):122 – 127, January 1969.

[77] K. Preetha Mathew, S. Vasant, and C. Pandu Rangan. ON PROVABLY SECURE CODE-BASED SIGNATURE AND SIGNCRYPTION SCHEME. *IACR Cryptology ePrint Archive*, 2012:585, 2012.

[78] K. Preetha Mathew, S. Vasant, S. Venkatesan, and C. Pandu Rangan. An Efficient IND-CCA2 Secure Variant of the Niederreiter Encryption Scheme in the Standard Model. In Willy Susilo, Yi Mu, and Jennifer Seberry, editors, *ACISP*, volume 7372 of *Lecture Notes in Computer Science*, pages 166–179. Springer, 2012.

[79] A. May, A. Meurer, and E. Thomae. Decoding Random Linear Codes in $\mathcal{O}(2^{0.054n})$. In D. H. Lee and X. Wang, editors, *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 107–124. Springer, 2011.

[80] R. McEliece. A Public-Key Cryptosystem Based on Algebraic Coding Theory. Technical report, NASA, 1978.

[81] C. Aguilar Melchor, P. Gaborit, and J. Schrek. A new zero-knowledge code based identification scheme with reduced communication. *CoRR*, abs/1111.1644, 2011.

[82] D. Micciancio. Improving Lattice Based Cryptosystems Using the Hermite Normal Form. In J. H. Silverman, editor, *CaLC*, volume 2146 of *Lecture Notes in Computer Science*, pages 126–145. Springer, 2001.

[83] L. Minder. *Cryptography based on error correcting codes*. PhD thesis, École Polytechnique Fédérale de Lausanne (Switzerland), 2007.

[84] L. Minder and A. Shokrollahi. Cryptanalysis of the Sidelnikov Cryptosystem. In M. Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 347–360. Springer, 2007.

[85] R. Misoczki and P. S. L. M. Barreto. Compact McEliece Keys from Goppa Codes. In M. J. Jacobson Jr., V. Rijmen, and R. Safavi-Naini, editors, *Selected Areas in Cryptography*, volume 5867 of *Lecture Notes in Computer Science*, pages 376–392. Springer, 2009.

[86] C. Monico, J. Rosenthal, and A. Shokrollahi. Using low density parity check codes in the McEliece cryptosystem. In *IEEE International Symposium on Information Theory, ISIT 2000*, page 215. IEEE, 2000.

[87] M. Naor and M. Yung. Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In H. Ortiz, editor, *STOC*, pages 427–437. ACM, 1990.

[88] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):159–166, 1986.

[89] R. Nojima, H. Imai, K. Kobara, and K. Morozov. Semantic security for the McEliece cryptosystem without random oracles. *Des. Codes Cryptography*, 49(1-3):289–305, 2008.

[90] T. Okamoto. Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In E. F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer, 1992.

[91] A. Otmani and J.-P. Tillich. An Efficient Attack on All Concrete KKS Proposals. In B.-Y. Yang, editor, *PQCrypto*, volume 7071 of *Lecture Notes in Computer Science*, pages 98–116. Springer, 2011.

[92] A. V. Ourivski and T. Johansson. New Technique for Decoding Codes in the Rank Metric and Its Cryptography Applications. *Problems of Information Transmission*, 38:237–246, 2002.

[93] R. Overbeck. A New Structural Attack for GPT and Variants. In E. Dawson and S. Vaudenay, editors, *Mycrypt*, volume 3715 of *Lecture Notes in Computer Science*, pages 50–63. Springer, 2005.

[94] 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.

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

[96] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 333–342, New York, NY, USA, 2009. ACM.

[97] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In C. Dwork, editor, *STOC*, pages 187–196. ACM, 2008.

[98] E. Persichetti. Compact McEliece keys based on Quasi-Dyadic Srivastava codes. *IACR Cryptology ePrint Archive*, 2011:179, 2011.

[99] C. Peters. Information-Set Decoding for Linear Codes over F_q. In N. Sendrier, editor, *PQCrypto*, volume 6061 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2010.

[100] C. Rackoff and D. R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In J. Feigenbaum, editor, *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer, 1991.

[101] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[102] B. Riemann. Theorie der Abel'schen Functionen. *Journal für die reine und angewandte Mathematik*, 54:115–155, 1857.

[103] R. L. Rivest. Cryptography. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 717–755. MIT Press, Cambridge, MA, USA, 1990.

[104] R. L. Rivest, A. Shamir, and L. M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[105] G. Roch. Über die Anzahl der willkurlichen Constanten in algebraischen Functionen. *Journal für die reine und angewandte Mathematik*, 64:372–376, 1865.

[106] A. Rosen and G. Segev. Chosen-Ciphertext Security via Correlated Products. In O. Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 419–436. Springer, 2009.

[107] D. Sarwate. On the complexity of decoding Goppa codes. *IEEE Transactions on Information Theory*, 23(4):515 – 516, jul 1977.

[108] S. Schechter. On the inversion of certain matrices. *Mathematical Tables and Other Aids to Computation*, 13(66):73–77, 1959.

[109] N. Sendrier. On the Concatenated Structure of a Linear Code. *Appl. Algebra Eng. Commun. Comput.*, 9(3):221–242, 1998.

[110] N. Sendrier. Finding the permutation between equivalent linear codes: The support splitting algorithm. *IEEE Transactions on Information Theory*, 46(4):1193–1203, 2000.

[111] N. Sendrier. The tightness of security reductions in code-based cryptography. *IEEE Information Theory Workshop (ITW)*, pages 415–419, October 2011.

[112] Nicolas Sendrier. Decoding One Out of Many. In Bo-Yin Yang, editor, *PQCrypto*, volume 7071 of *Lecture Notes in Computer Science*, pages 51–67. Springer, 2011.

[113] C. E. Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

[114] P. W. Shor. Polynomial time algorithms for discrete logarithms and factoring on a quantum computer. In L. M. Adleman and M.-D. A. Huang, editors, *ANTS*, volume 877 of *Lecture Notes in Computer Science*, page 289. Springer, 1994.

[115] V. Shoup. A proposal for an ISO standard for public key encryption (version 2.1). *IACR Cryptology ePrint Archive*, 112, 2001.

[116] V. M. Sidelnikov. A public-key cryptosystem based on binary Reed-Muller codes. *Discrete Mathematics and Applications*, 4(3):191 – 208, 1994.

[117] V. M. Sidelnikov and S. O. Shestakov. On insecurity of cryptosystems based on generalized Reed-Solomon codes. *Discrete Mathematics and Applications*, 2(4):439 – 444, 1992.

[118] J. R. Silvester. Determinants of Block Matrices. *The Mathematical Gazette*, 84(501):460–467, 2000.

[119] J. Stern. A method for finding codewords of small weight. In G. D. Cohen and J. Wolfmann, editors, *Coding Theory and Applications*, volume 388 of *Lecture Notes in Computer Science*, pages 106–113. Springer, 1988.

[120] J. Stern. A New Identification Scheme Based on Syndrome Decoding. In D. R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 13–21. Springer, 1993.

[121] J. Stern. Designing Identification Schemes with Keys of Short Size. In Y. Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 164–173. Springer, 1994.

[122] F. Strenzke. A Timing Attack against the Secret Permutation in the McEliece PKC. In N. Sendrier, editor, *PQCrypto*, volume 6061 of *Lecture Notes in Computer Science*, pages 95–107. Springer, 2010.

[123] F. Strenzke, E. Tews, H. G. Molter, R. Overbeck, and A. Shoufan. Side Channels in the McEliece PKC. In J. Buchmann and J. Ding, editors, *PQCrypto*, volume 5299 of *Lecture Notes in Computer Science*, pages 216–229. Springer, 2008.

[124] K. Tzeng and K. Zimmermann. On extending Goppa codes to cyclic codes. *IEEE Trans. Inf. Theor.*, 21(6):712–716, November 1975.

[125] P. Véron. Improved identification schemes based on error-correcting codes. *Appl. Algebra Eng. Commun. Comput.*, 8(1):57–69, 1996.

[126] A. C. Yao. Theory and Applications of Trapdoor Functions. In *FOCS*, pages 80–91. IEEE Computer Society, 1982.