UNIVERSITÀ DEGLI STUDI ROMA TRE
FACOLTÀ DI S.M.F.N.

Graduation Thesis in Mathematics
by Silvia Pastore

Normal bases in finite fields and Efficient Compact Subgroup Trace Representation

Supervisor
Prof. Francesco Pappalardi

The Candidate The Supervisor

ACADEMIC YEAR 2006 - 2007

Classification: 94A60 11T71 11Y16
Key Words: XTR, Normal Basis, Optimal Normal Basis.

A te che meritavi tanto,
A te, Nico.

Contents

Introduction

In the last twenty years, with the development of coding theory and appearance of several cryptosystems, the study of finite fields has generated much interest.

This impetus behind this interest has been the need to produce efficient algorithms that can be implemented in the hardware or software used for applications such as data encryption and public key distribution.

The necessity to produce efficient algorithms is translated into solving mathematical problems involving computations in finite fields.

From many years, normal bases have been used to represent finite fields and the advantages of this choice are been shown.

Hensel (1888) pioneered the study of normal bases for finite fields and proved that they always exist. Eisenstein (1850) had already noted that normal bases always exist. Hensel, and also Ore (1934), determine exactly the number of these bases.

In particular, in this thesis, we will show that the complexity of several hardware or software implementations, based on multiplication schemes, depends on the choice of the normal bases used. Hence is essential to find normal basis with "low complexity".

So we are interested in finding a lower bound for this complexity and we will present a costruction of a normal basis with this complexity.

Such a normal basis is called *optimal normal basis*.

We will analyse the effects of optimal normal bases on computations in finite fields, in particular on the finite field $\mathbb{F}_{p^2}$.

We will also describe how, using optimal normal basis, the computations in $\mathbb{F}_{p^2}$ can be done efficently. In particular we will obtain that p^{th} powers in $\mathbb{F}_{p^2}$ does not require any arithmetic operations.

This last result, with others obtained in $\mathbb{F}_{p^2}$, are at the base of a new method that use arithmetic in $\mathbb{F}_{p^2}$, the so-called XTR.

XTR was introduced at Crypto 2000 [9] by Arjen K. Lenstra and Eric R. Verheul, it stands for "ECSTR", which is an abbreviation for *Efficient and Compact Subgroup Trace Representation.*

XTR is a new method to represent elements of a subgroup of a multiplicative group of a finite field.

XTR is based on the same idea introduced by Shannon that is to replace a generator of the full multiplicative group of a finite field by the generator of a relatively small subgroup of sufficiently large prime order q.

XTR, infact, uses a subgroup of prime order q of the order $p^2 - p + 1$ subgroup of $\mathbb{F}_{p^6}^*$.

The cyclic group $\mathbb{F}_{p^6}^*$ has order $p^6 - 1$, therefore only a cyclic subgroup with order $p^2 - p + 1$ exists. We will refer to this last group as the *XTR supergroup* $\boldsymbol{X}$, while to the order q subgroup, with q dividing $p^2 - p + 1$, as the *XTR subgroup* $\boldsymbol{Y} = \langle g \rangle$.

We will show how XTR uses the trace over $\mathbb{F}_{p^2}$ to represent and calculate powers of elements of the *XTR subgroup*.

In fact, the arbitrary powers of an element of g can be represented using a single element of the subfield $\mathbb{F}_{p^2}$ and such powers can be computed efficiently using arithmetic operations in $\mathbb{F}_{p^2}$ while avoiding arithmetic in $\mathbb{F}_{p^6}$.

This last result is useful for cryptographic purposes if there is a way to efficently compute $Tr(g^n)$ given $Tr(g)$.

So we will find an algorithm that not only solves our last problem, but is also three times faster than computing g^n given g.

Naturally, we will have to choose appropriate parameters p and q to obtain an optimal normal basis and such that the *XTR subgroup* Y, with order q, cannot be embedded in the multiplicative group of any true subfield of $\mathbb{F}_{p^6}$.

Given $p, q > 3$, we will find $Tr(g) \in \mathbb{F}_{p^2}$ for an element $g \in \mathbb{F}_{p^6}$ of order q, through different methods.

From a security point of view, XTR is a traditional system based on the difficulty of solving discrete logarithm problems in the multiplicative group of a finite field.

Therefore XTR can be used in any cryptosystem that relies on the subgroup discrete logarithm problem. To illustrate this, we will present the XTR version of the Diffie Hellman key agreement protocol and ElGamal encryption and decryption protocols.

These versions follow by replacing the ordinary representation of subgroup elements with the XTR representation of subgroup elements of a multiplicative group that, as we have seen before, consists in replacing XTR group elements by their trace.

We will also see that, through this representation, the computation of Diffie Hellman Key agreement protocol and ElGamal encryption and decryption protocols becomes three times faster than computing g^n given g, as in their traditional versions.

We know that the traditional versions base their security on the difficulty to solve the discrete logarithm problem.

We will show that there is, also, the XTR discrete logarithm problem version on wich the security of XTR - Diffie Hellman and XTR - ElGamal protocols is based.

Moreover as *Diffie Hellman* problem and *Diffie Hellman Decision* problem are related to the *Discrete Logarithm* problem as, their XTR version, the *XTR - Diffie Hellman* problem and *XTR - Diffie Hellman Decision* problem, are related to XTR - Discrete Logarithm problem.

There are, also, some equivalences between old a new problems. We will use these equivalences to prove that DL problem in XTR subgroup Y is as hard as $\mathbb{F}_{p^6}$.

Because of the facts above, we can conclude that application of XTR in cryptographic protocols leads to the advantages in comunication and computational overhead without compromising security.

Our work is organized in the following way.

In the Chapter 1 we present some necessary definitions and properties. In particular it is necessary to define finite fields, Frobenius automorfism, cyclotomic fields, cyclotomic polynomials and roots of unity, all in the context

in which they will be used in this thesis.

It is presented a proof that shows that a normal basis exists for every finite extension of a finite field. The existence of a normal basis allows for particularly simple multiplications of the elements of the field.

In fact, we show as it is possible to compute the multiplication between two elements in $\mathbb{F}_{p^n}$, with respect to a normal bases $\mathcal{N} = \{\alpha, \alpha^p, \ \dots \ , \alpha^{p^{n-1}}\}$, using $n \times n$-matrix $T_{\mathcal{N}} = (t_{i,k})_{0 \leq i,k < n}$ with entries in $\mathbb{F}_p$, defined by:

$$\alpha^{p^i} \cdot \alpha = \sum_{0 \leq k < n} t_{i,k} \alpha^{p^k} \ for \ 0 \leq i < n$$

and as this multiplication is completely determined by $T_{\mathcal{N}}$.

This result is a consequence of the property of symmetry of the normal basis $\mathcal{N}$ that allows to reduce the n $n \times n$-matrices, $T_k = (t^k_{i,j})$ for $k = 0, \ \dots \ , n-1$, obtained by traditional multiplication between A and B, to the matrix T_0.

Then through the relation:

$$t^{(k)}_{i,j} = t_{i-j,k-j}$$

is set one correspondence between the matrix T_0 and the matrix T_N.

Therefore we use the matrix $T_{\mathcal{N}}$ to compute the multiplication between A and B.

This matrix is advantageous not only for the memory of the calculator saving, but also from computational costs point of view. We see that these costs depend by $d_{\mathcal{N}}$, the number of non-zero entries in $T_{\mathcal{N}}$.

Thus we define as $d_{\mathcal{N}}$ the complexity of normal basis $\mathcal{N}$, along with a proof that shows the lower bound for the complexity of any normal basis.

A particular type of normal basis, whose complexity value achieves the lower bound, is called *optimal normal basis*. Then, by this last basis, we obtain, also, a lower bound for the computational cost.

To construct the optimal normal basis, we prove the result:

Theorem 1.5 (Optimal normal bases, see [14]) *Let $n+1$ be a prime and p be a primitive root modulo $n+1$ (i.e. p modulo $n+1$ generates $\mathbb{F}^*_{n+1}$). Let ς*

be a primitive $(n+1)$*-th root of unity in an extension field of* $\mathbb{F}_p$ *and* $\alpha = \varsigma$*. Then* $N = \{\alpha, \alpha^p,\ \alpha^{p^2},\ \ldots, \alpha^{p^{n-1}}\}$ *is an optimal normal basis of* $\mathbb{F}_{p^n}$ *over* $\mathbb{F}_p$*.*

Then knowing the conditions for which an optimal normal basis exists, we obtain, also, the semplification of the process of selecting finite fields for specific applications.

In Chapter 2 we deal with the mathematic of XTR.

To this purpose, at first we present a representation of $\mathbb{F}_{p^2}$ and then we describe how the computations in $\mathbb{F}_{p^2}$ can be done efficiently.

So we set $p \equiv 2 \pmod 3$. Hence $X^2 + X + 1$ is an irriducible polynomial over $\mathbb{F}_p$. We represent $\mathbb{F}_{p^2}$ as:

$$\mathbb{F}_{p^2} \cong \left\{x_1\alpha \ + \ x_2\alpha^p : \ \alpha^2 + \alpha + 1 = 0 \ , \ x_1, \ x_2 \in \mathbb{F}_p\right\}$$

where $\{\alpha,\ \alpha^p\}$ are the zeros of the polynomial X^2+X+1 and form an optimal normal basis for $\mathbb{F}_{p^2}$ over $\mathbb{F}_p$, since $p \equiv 2 \pmod 3$ and p is a primitive root modulo 3, i.e. $p \pmod 3$ generates $\mathbb{F}_3^*$.

Considering the elements $x, y, z \in \mathbb{F}_{p^2}$, we see as it is possible to compute efficently $x^p, x^2, x \cdot y$ and $x \cdot z - y \cdot z^p$ in $\mathbb{F}_p$. For comparison purposes, we compute the same relation in the case $x, y, z \in \mathbb{F}_{p^6}$.

These results follow, also, by the use of Karatsuba's trick and multi-exponentation algorithm, described in the appendices.

In Chapter 3 we present the definition of the trace over $\mathbb{F}_{p^2}$ we show that it is $\mathbb{F}_{p^2}$-linear. We show that the trace of the element $g \in \mathbb{F}_{p^6}^*$ of order q that divides $p^2 - p + 1$ is equal to the trace of its conjugates.

Set p a prime and set $F(c, X)$ for $c \in \mathbb{F}_{p^2}$ be the polynomial $X^3 - cX^2 + c^pX - 1 \in \mathbb{F}_{p^2}[X]$; for $n \in \mathbb{Z}$ we denote by c_n the sum of the n^{th} powers of the roots of $F(c, X)$ and so we prove the following result:

Theorem 3.1. *The roots of* $X^3 - Tr(g)X^2 + Tr(g)^pX - 1$ *are the conjugates of* g*.*

Then the minimal polynomial of any $g \in \mathbb{F}_{p^6}$ of order dividing $p^2 - p + 1$

and > 3 is equal to $F(Tr(g), X)$. Hence the fundamental idea of XTR is, as illustrated, that the trace value fully specifies g's minimal polynomial and thus the conjugates of g.

In a similar way we illustrate that this last result is true for any power of n. In fact, for any integer n, the conjugates of g^n are the roots of $X^3 - Tr(g^n)X^2 + Tr(g^n)^p X - 1 \in \mathbb{F}_{p^2}$ and the latter polynomial and the conjugates of g^n are determined by $Tr(g^n) \in \mathbb{F}_{p^2}$.

Vice versa we show that if $F(c, X) \in \mathbb{F}_{p^2}[X]$ is irreducible, then the roots of $F(c, X)$ take the form h, h^{p^2}, h^{p^4} for some $h \in \mathbb{F}_{p^6}$ of order dividing $p^2 - p + 1$ and > 3.

This implies one of the property of c_n, i.e. it is of the form $Tr(h^n) \in \mathbb{F}_{p^2}$.

Another result that we obtain on $F(c, X)$ over $\mathbb{F}_{p^2}$ is the theorem:

Theorem 3.3 *$F(c, X)$ is reducible over $\mathbb{F}_{p^2}$ if and only if $c_{p+1} \in \mathbb{F}_p$.*

Then, let $g \in \mathbb{F}_{p^6}$ with order q for a prime $q > 3$ dividing $p^2 - p + 1$, it follows from the results cited above that $F(Tr(g^n), g^n) = 0$ and $c_n = Tr(g^n) \in \mathbb{F}_{p^2}$.

These results are useful for cryptographic purpose if, as we have said above, there is a way to efficiently compute $Tr(g^n)$ given $Tr(g)$. To this goal we describe an Algorithm that computes c_n for any $n \in \mathbb{Z}$ and such that is three times faster than the algorithm that computes g^n given g.

The efficiency of this algorithm it is based on properties of c_n that have been illustrated in this chapter.

In Chapter 4 we describe fast and pratical methods that satisfy these two conditions:

1. p and $n + 1$ are prime numbers and p is a primitive root $(\mathrm{mod}\ n + 1)$.
2. $\Phi_6(p) = p^2 - p + 1$ has a prime factor q.

By these conditions we obtain that $\mathbb{F}_{p^2}$ has an optimal normal basis and that the *XTR subgroup* $\boldsymbol{Y}$, with the order q, cannot be embedded in the multiplicative group of any proper subfield of $\mathbb{F}_{p^6}$.

So the primes p and q of appropriate sizes can be found using both the methods that we present in this thesis. In particular through the first one, we find a prime p that satisfies a second degree polynomial with small coefficients. Moreover, for $k = 1$, the prime p assumes a very nice form.

Given p and $q > 3$, we must find $Tr(g) \in \mathbb{F}_{p^2}$ for an element $g \in \mathbb{F}_{p^6}$ of order q.

We show that to find a proper $Tr(g)$ it suffices to find $c \in \mathbb{F}_{p^2} \setminus \mathbb{F}_p$ such that $F(c, X) \in \mathbb{F}_{p^2}[X]$ is irreducible, such that $c_{(p^2-p+1)/q} \neq 3$ and to put $Tr(g) = c_{(p^2-p+1)/q}$.

To find $c \in \mathbb{F}_{p^2}$ such that $F(c, X)$ is irreducible we consider the following result:

Lemma 4.1 *For a randomly selected $c \in \mathbb{F}_{p^2}$ the probability that $F(c, X) \in \mathbb{F}_{p^2}[X]$ is irreducible is about one third.*

We close the chapter presenting an algorithm that it is based on the above Lemma and on the fact that $F(c, x)$ is reducible if and only if $c_{p+1} \in \mathbb{F}_p$, that is a result of Chapter 3.

In the Chapter 5, we deal with the XTR version of the Diffie Hellman key agreement protocol and ElGamal encryption and decryption protocols, since XTR can be used in any cryptosystem that relies on the subgroup discrete logarithm problems.

This version is different to the traditional, since it is based on the XTR representation of the XTR group elements that consists in replacing the XTR elements with their trace.

In these protocols there are two users **A** and **B** that already agreed upon *XTR subgroup* $\boldsymbol{Y}$ and on the XTR public key data $(p, q, Tr(g))$, where the two primes p and q are as in chapter 4, and $Tr(g)$ is the trace of an generator of *XTR subgroup* $\boldsymbol{Y}$.

As is well-know, while in the XTR Diffie Hellman key agreement protocol the two users want to agree on a secret key K, in ElGamal encryption and decryption protocols, we suppose that **B** wants to encrypt a message M

intended for **A** and that **A** decrypts the message $(Tr(g^b), E)$ that he has received by **B**. For this reason, we suppose that in the key XTR public data of **A** there is also the value $Tr(g^k)$ that is computed and made public by **A**, for an integer k selected by **A**, that it is kept secret, the so-called private key.

Analysing these algorithms, we note that they compute $Tr(g^h)$, where g^h is an element in XTR subgroup with $0 < h < q - 2$, using the algorithm described in Chapter 3, to compute $Tr(g^h)$ given $Tr(g)$.

For this reason the XTR Diffie Hellman key agreement protocol and XTR ElGamal encryption and decryption protocols are three times faster than their traditional versions.

Moreover these last versions base their security on the difficulty to solve the *Discrete Logarithm* (DL) problem and we show that *Diffie Hellman* (DH) problem and *Diffie Hellman Decision* (DHD) problem are related to DL problem.

But since the XTR version of the protocols follows by replacing elements of the XTR group by their traces, then the security of these protocols is no longer based on the original DH, DHD or DL problems, but on their XTR versions.

If we define the equivalence between two problem through this result:

Definition 5.1 *We say that problem $\mathcal{A}$ is (a, b)-equivalent to problem $\mathcal{B}$, if any istance of problem $\mathcal{A}$ (or $\mathcal{B}$) can be solved by at most a (or b) calls to an algorithm solving problem $\mathcal{B}$ (or $\mathcal{A}$).*

We obtain this fundamental equivalences between old and new problems:

Theorem 5.1 *The following equivalences hold:*

i. The XTR-DL problem is $(1, 1)$-equivalent to the DL problem in G.

ii. The XTR-DH problem is $(1, 2)$-equivalent to the DH problem in G.

iii. The XTR-DHD problem is $(3, 2)$-equivalent to the DHD problem in G.

Then thanks to this last result, we prove that DL problem in XTR sub-

group $\boldsymbol{Y}$ is as hard as in $\mathbb{F}_{p^6}^*$.

In fact we show that the *Discrete Logarithm problem* in $\mathbb{F}_{p^6}$, can be reduced to the discrete logarithm problems in multiplicative groups of proper subfields $\mathbb{F}_p$, $\mathbb{F}_{p^2}$ and $\mathbb{F}_{p^3}$ of $\mathbb{F}_{p^6}$ and in the subgroup of order p^2-p+1 that cannot be embedded in any proper subfield of $\mathbb{F}_{p^6}$.

Since the first three problems are believed to be considerably easier than the problem in $\mathbb{F}_{p^6}^*$, we conclude that the hardness of computing discrete logarithm in the group $\mathbb{F}_{p^6}^*$ must reside in its subgroup of order p^2-p+1.

From this result, considering the XTR subgroup $\boldsymbol{Y}$ that has order a large prime q that divide p^2-p+1 and that cannot be embedded into any proper subfield of $\mathbb{F}_{p^6}$, we have concluded that the discrete logarithm problem in $\boldsymbol{Y}$ is as hard as it is $\mathbb{F}_{p^6}$.

Then the XTR version of the Diffie Hellman key agreement protocol and XTR ElGamal encryption and decryption protocols makes this protocol more fast and secure.

At the end of this thesis there are two appendices, in which some of algorithms, used in Chapter 2, are discussed.

In Appendix A, we describe the Karatsuba's trick for the multiplication of two polynomials of degree less than n and we show that this trick makes the computational cost of this multiplication lower.

In fact, while with the classic multiplication algorithm the number of multiplication is $O(n^2)$, with the Karatsuba's trick the number of multiplication it is reduced to $O(n^{1.59})$.

In Appendix B, we deal with the Pekmestzi's algorithm, that is a method to compute the multi-exponentation, i.e. $Z=\prod_{0<i\leq k} X_i^{a_i} \bmod N$.

In particular we consider the computation of $X^aY^b \bmod N$ and we show how this method is faster than the traditional computation that computes the two modular exponentations $X^a \bmod N$ and $Y^b \bmod N$ separetely.

Naturally with the Pekmestzi's algorithm, the computational cost im-

proves and so it can be used to implement fast modern cryptosystems that have as a major task to obtain an efficient performance.

Chapter 1

Background

In this chapter we present some basic facts about the algebraic structures that we shall use. In particular we focus our attention on the existence of normal bases over finite fields and in the construction of optimal normal bases in an extension finite field. We shall debate, also, on the advantages to use normal and optimal normal basis from computational point of view.

1.1 Algebraic structures

We report in this section some basic definitions and basic facts of the algebraic structures that are used in this thesis. For details and proofs we refer to book of *Lidl & Niederreiter (1983)*.

A finite field $\mathbb{F}$ is a finite ring satisfying the property that $(\mathbb{F}\setminus\{0\},\ \cdot)$ is a commutative group.

It is a standard fact that if $\mathbb{F}$ is a finite field containing a subfield K with p elements, then $\mathbb{F}$ has p^n elements, where $n = [\mathbb{F} : K]$ is the degree of extension of $\mathbb{F}$ over K.

Two finite fields with the same number of elements are isomorphic.

Thus we denote finite field with $q = p^n$ elements, or the finite field of order $q = p^n$ by $\mathbb{F}_q$, where the prime p is the characteristic of $\mathbb{F}$ and n is the degree of $\mathbb{F}$ over its prime subfield.

There are different ways to represent a finite field with p^n elements, one of these methods is to consider an irreducible polynomial over $\mathbb{F}_p$ of degree n, $f \in \mathbb{F}_p[x]$, then adjoining a root α of f to $\mathbb{F}_p$ we get finite field with p^n elements and $\mathbb{F}_{p^n} = \mathbb{F}_p(\alpha)$ (see [10], Theorem 2.14).

Every element of $\mathbb{F}_{p^n}$ can be uniquely expressed as a polynomial in α over $\mathbb{F}_p$, of degree less than n (see [10], Theorem 1.86):

$$\begin{aligned}\mathbb{F}_q &= \frac{\mathbb{F}_p}{(f)} \\ &= \left\{A_0 + ... + A_{n-1}\alpha^{n-1} : A_i \in \mathbb{F}_p \text{ for } i = 0, ..., n-1 \text{ and } f(\alpha) \equiv 0\right\}\end{aligned}$$

1.2 Normal bases

The importance of the normal bases is due to the development of coding theory and the implementation of several cryptosystems using finite fields. We see, before, that the advantages of this choice reside in the improvement of the computational cost (see [11]).

Definition 1.1. *Let $\mathbb{F}_{p^n}$ be an extension of $\mathbb{F}_p$ and let $\alpha \in \mathbb{F}_{p^n}$. Then the elements $\alpha,\ \alpha^p,\ \alpha^{p^2},\ ...\ , \alpha^{p^{n-1}}$ are called the conjugates of α with respect to $\mathbb{F}_p$*

Definition 1.2 (Normal basis, see [10])**.** *Let $\mathbb{F}_{p^n}$ be an extension of $\mathbb{F}_p$. Then a basis of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$ of the form $\mathcal{N} = \{\alpha,\ ...\ , \alpha^{p^{n-1}}\}$ consisting of the conjugates of a suitable element $\alpha \in \mathbb{F}_{p^n}$ with respect to $\mathbb{F}_p$, is called a normal basis of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$. The element α is then called normal in $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$.*

Then every $A \in \mathbb{F}_{p^n}$ may be uniquely expressed in term of $\mathcal{N}$ as

$$A = A_0\alpha + A_1\alpha^p + ... + A_{n-1}\alpha^{p^{n-1}} = \sum_{i=0}^{n-1} A_i \cdot \alpha^{p^i}$$

with coefficients $A_0,\ A_1,\ \ldots\ ,\ A_{n-1} \in \mathbb{F}_p$

EXAMPLE (Normal and not normal elements)

Let $f(x) = x^3 + x + 1$ an irreducible polynomial on $\mathbb{F}_2$, then:

$$\begin{aligned}\mathbb{F}_8 = \frac{\mathbb{F}_2}{(x^3+x+1)} &= \{a_0 + a_1\alpha + a_2\alpha^2 : a_0, a_1, a_2 \in \mathbb{F}_2 \text{ and } \alpha^3 + \alpha + 1 \equiv 0\} \\ &= \{0,\ 1,\ \alpha,\ \alpha^2,\ \alpha+1,\ \alpha^2+1,\ \alpha^2+\alpha,\ \alpha^2+\alpha+1\}\end{aligned}$$

Note that α is not normal since

$$\{\alpha,\ \alpha^2,\ \alpha^4 \equiv \alpha^2 + \alpha\}$$

is not a base.

While $\alpha + 1$ is normal, since:

$$\begin{aligned}&(\alpha+1) \qquad (\text{mod } f)\\ &(\alpha+1)^2 = \alpha^2 + 1 \quad (\text{mod } f)\\ &(\alpha+1)^4 = (\alpha^2+1)^2 = \alpha^4 + 1 = \alpha^2 + \alpha + 1 \quad (\text{mod } f)\end{aligned}$$

is a basis for $\mathbb{F}_2[\alpha]$ over $\mathbb{F}_2$.

If we compute the conjugates of all the elements of $\mathbb{F}_8$, we obtain that:

$$\{0,\ 1,\ \alpha,\ \alpha^2,\ \alpha^2+\alpha\}$$

are not normal elements, while

$$\{\alpha+1,\ \alpha^2+1,\ \alpha^2+\alpha\}$$

are normal elements.

Now we present a proof that shows the existence of a normal basis in a finite field extension and in the next section we show how the existence of a

normal basis allows for particularly simple multiplication of the elements of the field.

For this purpose we recall few useful definitions.

Definition 1.3 (Frobenius automorfism, see [11]). *Let $\mathbb{F}_{p^n}$ be an extension of $\mathbb{F}_p$, then the Frobenius automorphism is a map of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$ of the form:*

$$\sigma : \eta \mapsto \eta^p, \ \eta \in \mathbb{F}_{p^n}$$

If we view $\mathbb{F}_{p^n}$ as a vector space of dimension n over $\mathbb{F}_p$ then the Frobenius map σ is a linear transformation of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$, since $\sigma(\alpha + \beta) = \sigma(\alpha) + \sigma(\beta)$ and $\sigma(c \cdot \alpha) = c \cdot \sigma(\alpha)$ for all $\alpha, \ \beta \in \mathbb{F}_{p^n}$ and $c \in \mathbb{F}_p$.

Definition 1.4 (see [11]). *Let Γ a linear transformation on a finite-dimensional vector space V over a (arbitrary) field F, a polynomial $f(x) = \sum_{i=0}^{n} A_i x^i$ in $F[x]$ is said to annihilate Γ if*

$$f(\Gamma) = A_n \Gamma^n \ + \ A_{n-1}\Gamma^{n-1} \ + \ ... \ + \ \Gamma_0 I = 0$$

where I is the identity map and 0 is the zero map on V.

The uniquely determined monic polynomial of least degree that satisfies the last property is called the *minimal polynomial* for Γ.

In particular this polynomial divides any other polynomial in $F[x]$ annihilating Γ and the cheracteristic polynomial for Γ (Cayley-Hamilton theorem).

We say that a vector $\alpha \in V$ is a *cyclic vector* for T if the vectors $T^k\alpha, \ k = 0, \ 1, \ ...$ span V

Lemma 1.1. *(see [10])*

Let T be a linear operator on the finite-dimensional vector space V. Then T has a cyclic vector if and only if the characteristic and minimal polynomials for T are identical.

Remark 1.1. We are not able to prove this last Lemma in the particular case of finite fields.

Theorem 1.1 (Normal Basis Theorem, see [10, 11])**.** *For any finite field* $\mathbb{F}_p$ *and any finite extension* $\mathbb{F}_{p^n}$ *of* $\mathbb{F}_p$ *there exists a normal basis of* $\mathbb{F}_{p^n}$ *over* $\mathbb{F}_p$*.*

Proof . We have note before that the Frobenius automorphism σ is a linear operator on $\mathbb{F}_{p^n}$.

Thus we know that $\sigma^n(\eta) = \eta^{p^n} = \eta$ for every $\eta \in \mathbb{F}_{p^n}$, then $\sigma^n = 1$ and so σ satsfies the polynomial $x^n - 1$.

We prove that $x^n - 1$ is the minimal polynomial of σ.

We suppose that there is a polynomial

$$f(x) = \sum_{i=0}^{n-1} A_i x^i \in \mathbb{F}_p[x]$$

of degree less than n that annihilates σ, therefore we have:

$$f(x) = \sum_{i=0}^{n-1} A_i \sigma^i = 0$$

Then

$$\left(\sum_{i=0}^{n-1} A_i \sigma^i\right)(\eta) = \sum_{i=0}^{n-1} A_i \sigma^i(\eta) = \sum_{i=0}^{n-1} \eta^{p^i} = 0$$

for any $\eta \in \mathbb{F}_{p^n}$.

Thus η is a root of the polynomial $f(x)$. This is absurd because the degree of the polynomial $f(x)$ is at most p^{n-1} and so $f(x)$ cannot have $p^n > p^{n-1}$ roots in $\mathbb{F}_{p^n}$. Hence $x^n - 1$ is the minimal polynomial of σ.

Moreover the minimal polynomial of σ divides the characteristic polynomial of σ (Cayley-Hamilton theorem). But the characteristic polynomial of σ is monic and of degree n then the characteristic polynomial of σ is equal to the minimal polynomial of σ, namely $x^n - 1$.

Thus there exists an element $\alpha \in \mathbb{F}_{p^n}$ such that $\alpha,\ \sigma(\alpha),\ \sigma^2(\alpha),\ \ldots$ span $\mathbb{F}_{p^n}$ and form a basis of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$ (cf. Lemma 1.1). But the elements of this basis are α and its conjugates with respect to $\mathbb{F}_p$ so these elements form a normal basis of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$ □

1.2.1 The multiplication matrix

The crucial point in normal basis arithmetic is multiplication. We first consider the general multiplication scheme in $\mathbb{F}_{p^n}$.

Let $\mathbb{F}_{p^n}$ be an extension of $\mathbb{F}_p$, let $\mathcal{B} = \{\alpha_0,\ \alpha_1,\ \ldots\ , \alpha_{n-1}\}$ a basis over $\mathbb{F}_p$ and let

$$A = \sum_{0 \le i < n} A_i \alpha_i \text{ and } B = \sum_{0 \le j < n} B_j \alpha_j \text{ with } A_i,\ B_j \in \mathbb{F}_p$$

be two elements in $\mathbb{F}_{p^n}$ with respect to $\mathcal{B}$.

Because $\mathbb{F}_{p^n}$ can be identified as $\mathbb{F}_p^n$, i.e the set of all $n-$tuples over $\mathbb{F}_p$, so we can write A and B in the form

$$A = (A_0,\ A_1,\ \ldots\ , A_{n-1}) \text{ and } B = (B_0,\ B_1,\ \ldots\ , B_{n-1})$$

Hence multiplication is:

$$A \cdot B = \sum_{0 \le i,j < n} A_i B_j \alpha_i \cdot \alpha_j$$

Set

$$\alpha_i \cdot \alpha_j = \sum_{0 \le k < n} t_{i,j}^{(k)} \alpha_k \text{ with } t_{i,j}^{(k)} \in \mathbb{F}_p \tag{1.1}$$

we obtain

$$\begin{aligned} C = A \cdot B &= \sum_{0 \le i,j < n} A_i B_j \cdot \alpha_i \cdot \alpha_j = \\ &= \sum_{0 \le i,j < n} A_i B_j \left(\sum_{0 \le k < n} t_{i,j}^{(k)} \alpha_k \right) = \\ &= \sum_{0 \le k < n} \left(\sum_{0 \le i,j < n} A_i B_j t_{i,j}^{(k)} \right) \alpha_k = \\ &= \sum_{0 \le k < n} c_k \alpha_k \end{aligned}$$

where

$$c_k = \sum_{0 \le i,j < n} A_i B_j t_{i,j}^{(k)}$$

and $T_k = (t^k_{i,j})$ is an $n \times n$ matrix over $\mathbb{F}_p$.

Thus to implement the multiplication in $\mathbb{F}_{p^n}$ is necessary to compute c_k for any $k = 0,\ 1,\ \ldots\ , n-1$ and each c_k require from computational point of view d_k multiplications in $\mathbb{F}_p$ where d_k is the number of nonzero entries in T_k for any $k = 0,\ 1,\ \ldots,\ n-1$, then to compute C are necessary $n \cdot d_k$ multiplications for any $k = 0,\ 1,\ \ldots,\ n-1$ and $(d_k - 1) \cdot n$ additions in $\mathbb{F}_{p^n}$.

Now we consider a normal basis $\mathcal{N} = \{\alpha_0,\ \alpha_1,\ \ldots\ ,\ \alpha_{n-1}\}$ of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$, where we have set $\alpha_i = \alpha^{p^i}$.

Let $A = \sum_{0 \le i < n} A_i \alpha_i$ then

$$A^p = \sigma(A) = \sigma\left(\sum_{0 \le i < n} A_i \alpha_i\right) = \sum_{0 \le i < n} A_i \sigma(\alpha_i) = \sum_{0 \le i < n} A_{i-1} \alpha_i$$

so the coordinate vectors of A are $(A_{n-1},\ A_0,\ \ldots\ ,\ A_{n-2})$, then A^p is obtained from A by one cyclic shift of the the coordinates of A and can be compute without any operations in $\mathbb{F}_p$.

If we now consider the identity (1.1) and we raise both sides of equation to the p^{-m}-th powers one finds that:

$$t^{(m)}_{i,j} = t^{(0)}_{i-m,j-m} \text{ for any } 0 \le i, j, m < n$$

Thus the matrix T_0 that we use to compute c_0 with the coordinates of A and B is the same matrix that we use to compute c_m with the coordinates of $A^{p^{-m}}$ and $B^{p^{-m}}$, but we know that $A^{p^{-m}}$ as $B^{p^{-m}}$ are obtain with m-cyclic shift of the vector representations of A and B, then we can compute any c_k shifting the A and B vectors. From this property of symmetry of normal basis $\mathcal{N}$ it follows that the computational cost of the multiplication between A and B depend to the number of nonzero terms of T_0 (see [11]).

Now we define $n \times n-$matrix $T_{\mathcal{N}} = (t_{i,k})_{0 \le i,k < n}$ with entries in $\mathbb{F}_p$ by:

$$\alpha^{p^i} \cdot \alpha = \sum_{0 \le k < n} t_{i,k} \alpha^{p^k} \ \ for\ 0 \le i < n$$

this matrix is called the *multiplication matrix* of $\mathcal{N}$ and is denoted with $T_{\mathcal{N}}$.

We shall show how is possible to compute the multiplication between A and B using $T_{\mathcal{N}}$ and how the computational cost of their multiplication with this matrix improves.

Let

$$A = \sum_{0 \leq i < n} A_i \cdot \alpha^{p^i} \text{ and } B = \sum_{0 \leq j < n} B_j \cdot \alpha^{p^j}$$

be two elements in $\mathbb{F}_{p^n}$ with respect to $\mathcal{N}$, it follows:

$$\begin{aligned} C = AB &= \left(\sum_{0 \leq i < n} A_i \cdot \alpha^{p^i} \right) \cdot \left(\sum_{0 \leq j < n} B_j \cdot \alpha^{p^j} \right) \\ &= \sum_{0 \leq i < n} \sum_{0 \leq j < n} A_i B_j \alpha^{p^i} \alpha^{p^j} \end{aligned} \tag{1.2}$$

We can write:

$$\begin{aligned} \alpha^{p^i} \cdot \alpha^{p^j} &= \left(\alpha^{p^{i-j}} \cdot \alpha \right)^{p^j} = \left(\sum_{0 \leq k < n} t_{i-j,k} \alpha^{p^k} \right)^{p^j} \\ &= \sum_{0 \leq k < n} t_{i-j,k} \alpha^{p^{k+j}} = \sum_{0 \leq k < n} t_{i-j,k-j} \alpha^{p^k} \end{aligned} \tag{1.3}$$

and substituting in (1.2):

$$\begin{aligned} C = AB &= \sum_{0 \leq i < n} \sum_{0 \leq j < n} A_i B_j \alpha^{p^i} \alpha^{p^j} \\ &= \sum_{0 \leq i,\ j < n} A_i B_j \cdot \left(\sum_{0 \leq k < n} t_{i-j,k-j} \alpha^{p^k} \right) \\ &= \sum_{0 \leq k < n} \left(\sum_{0 \leq i,\ j < n} A_i \cdot B_j \cdot t_{i-j,k-j} \right) \alpha^{p^k} \\ &= \sum_{0 \leq k < n} C_k \alpha^{p^k} \end{aligned} \tag{1.4}$$

where $C_k = \sum_{0 \leq i,\ j < n} A_i \cdot B_j \cdot t_{i-j,k-j}$ are the coefficients $C_0, \ \ldots \ , \ C_{n-1}$ of the product $C = A \cdot B$.

If we consider the identities (1.1) and (1.3), we obtain that

$$t_{i,j}^{(k)} = t_{i-j,k-j}$$

Therefore the number of non-zero entries in T_0 is the number of non-zero entries in $T_{\mathcal{N}}$ and so the multiplication between A and B in the normal basis representation of $\mathbb{F}_{p^n}$ is completely determined by $T_{\mathcal{N}}$.

Definition 1.5 (see [14]). *Let* $d_{\mathcal{N}} = \#\left\{(i,k) \in \{0, \ldots, n-1\}^2 : t_{i,k} \neq 0\right\}$ *be the number of non-zero entries in* $T_{\mathcal{N}}$.

Then $d_{\mathcal{N}}$ is the complexity of the normal basis $\mathcal{N}$ and the product of two elements in $\mathbb{F}_{p^n}$ with respect to this basis can be computed with at most $n \cdot d_{\mathcal{N}}$ multiplication and at most $(d_{\mathcal{N}} - 1) \cdot n$ additions in $\mathbb{F}_p$ as in general case.

But while in the last one we must memorize a $n \times n$-matrix, in the first case we must recall n $n \times n$-matrix.

Obviously we wish to find a normal basis such that $d_{\mathcal{N}}$ is the smallest possible. The following theorem gives as a lower bound for $d_{\mathcal{N}}$.

Theorem 1.2 (see [12]). *Let* $\mathcal{N}$ *be a normal basis in* $\mathbb{F}_{p^n}$ *over* $\mathbb{F}_p$ *with density* $d_{\mathcal{N}}$. *Then* $d_{\mathcal{N}} \geq 2n - 1$.

Proof . Let $\mathcal{N} = \{\alpha, \alpha^p, \alpha^{p^2}, \ldots, \alpha^{p^{n-1}}\}$ be the normal basis over $\mathbb{F}_p$. Let $Tr(\alpha) = \sum_{i=0}^{n-1} \alpha^{p^i}$ be the trace of α over $\mathbb{F}_p$ and let $T_{\mathcal{N}} = (t_{i,k})_{0 \leq i,k < n}$ be the multiplication matrix of $\mathcal{N}$

$$Tr(\alpha) \cdot \alpha = \sum_{0 \leq i < n} (\alpha^{p^i} \cdot \alpha)$$

Since $(\alpha^{p^i} \cdot \alpha) = \sum_{0 \leq k < n} t_{i,k} \alpha^{p^k}$ we have

$$\begin{aligned} Tr(\alpha) \cdot \alpha &= \sum_{0 \leq i < n} \sum_{0 \leq k < n} t_{i,k} \alpha^{p^k} \\ &= \sum_{0 \leq k < n} \left(\sum_{0 \leq i < n} t_{i,k} \right) \alpha^{p^k} \end{aligned} \tag{1.5}$$

and comparing coefficients proves

$$\sum_{0 \leq i < n} t_{i,k} = \begin{cases} Tr(\alpha) & \text{if } k = 0 \\ 0 & else \end{cases} \tag{1.6}$$

Thus the sum of rows of $T_{\mathcal{N}}$ is an n-tuple with $Tr(\alpha)$ in position 1 and zeros elsewhere.

Since α is nonzero and $\{\alpha\alpha^{p^i} : 0 \leq i < n\}$ is also a basis of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$, then the rows of $T_{\mathcal{N}}$ are linearly indipendent. Therefore each column of $T_{\mathcal{N}}$ must contain at least one nonzero element. Moreover with respect to (1.6) there are at least two nonzero elements in each column with the possible exception of column one.

So the total number of nonzero elements in $T_{\mathcal{N}}$ is at least $2n-1$ $\square$

Definition 1.6 (Optimal normal bases (1), see [12, 14]). *We say that* $\mathcal{N}$ *is an optimal normal basis of* $\mathbb{F}_{p^n}$ *if* $d_{\mathcal{N}} = 2n-1$.

Now we illustrate through an example a common way to compute the multiplication matrix $T_{\mathcal{N}}$ with respect to a normal basis $\mathcal{N} = \{\alpha, \ \ldots \ , \alpha^{p^{n-1}}\}$.

EXAMPLE (Construction of $T_{\mathcal{N}}$)

1. We consider an irreducible minimal polynomial $f \in \mathbb{F}_2[x]$ and let α be one of its roots. Thus we have the field

$$\mathbb{F}_{2^5} = \frac{\mathbb{F}_2}{(x^5 \ + \ x^2 \ + \ 1)} = \\ = \left\{ \sum_{0 \leq i < 5} a_i \cdot \alpha^i : a_i \ \in \ \mathbb{F}_2 \text{ and } f(\alpha) \equiv 0 \right\}$$

Set $\beta = \alpha^3$, then $\{\beta, \ \beta^2, \ \beta^4, \ \beta^8, \ \beta^{16}\}$ is a normal basis, in fact

$$\begin{aligned} \beta &= \alpha^3 \pmod f \\ \beta^2 &= \alpha^6 \ \equiv \ \alpha^3 \ + \ \alpha \pmod f \\ \beta^4 &= \alpha^{12} \ \equiv \ \alpha^3 \ + \ \alpha^2 \ + \ \alpha \pmod f \\ \beta^8 &= \alpha^{24} \ \equiv \ \alpha^4 \ + \ \alpha^3 \ + \ \alpha^2 \ + \ \alpha \pmod f \\ \beta^{16} &= \alpha^{48} \ \equiv \ \alpha^4 \ + \ \alpha \ + \ 1 \pmod f \end{aligned}$$

Then we have:

$$(\beta,\ \beta^2,\ \beta^4,\ \beta^8,\ \beta^{16}) = (1,\ \alpha,\ \alpha^2,\ \alpha^3,\ \alpha^4) \cdot \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

and

$$\begin{vmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{vmatrix} \neq 1$$

2. The computation of the products $\beta^{p^i} \cdot \beta$ for $0 \leq i < 5$ in $\mathbb{F}_{2^5}$ gives

$$\begin{aligned} \beta \cdot \beta &= \alpha^3 \cdot \alpha^3 \equiv \alpha^3 + \alpha \pmod{f} \\ \beta^2 \cdot \beta &= \alpha^6 \cdot \alpha^3 \equiv \alpha^4 + \alpha^3 + \alpha \pmod{f} \\ \beta^4 \cdot \beta &= \alpha^{12} \cdot \alpha^3 \equiv \alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1 \pmod{f} \\ \beta^8 \cdot \beta &= \alpha^{24} \cdot \alpha^3 \equiv \alpha^3 + \alpha + 1 \pmod{f} \\ \beta^{16} \cdot \beta &= \alpha^{48} \cdot \alpha^3 \equiv \alpha^3 + \alpha^2 \pmod{f} \end{aligned}$$

3. Now we write a matrix T such that in the left part contains the coefficient of the polynomial basis representation of β^{2^i} *for* $i = 0, ..., 4$, while in the right part contains the coefficient of the polynomial repre-

sentation of $\beta^{2^i} \cdot \beta$.

$$T = \left(\begin{array}{ccccc|ccccc} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{array}\right)$$

We can perform Gaussian elimination on T to write

$$\beta \cdot \beta,\ \beta^2 \cdot \beta,\ \beta^4 \cdot \beta,\ \beta^8 \cdot \beta,\ \beta^{16} \cdot \beta$$

as a linear combination of the conjugates of β. The output is then $\left(I_4 | \overline{T}\right)$ with I_5 the 5×5-identity matrix and

$$\overline{T} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

4. The multiplication matrix $T_{\mathcal{N}}$ is the transposed matrix of $\overline{T}$. Then

$$\begin{aligned} \beta \cdot \beta &= \beta^2 \\ \beta^2 \cdot \beta &= \beta^2 + \beta^{2^2} + \beta^{2^3} \\ \beta^{2^2} \cdot \beta &= \beta + \beta^2 + \beta^{2^2} + \beta^{2^4} \\ \beta^{2^3} \cdot \beta &= \beta + \beta^{2^2} + \beta^{2^3} + \beta^{2^4} \\ \beta^{2^4} \cdot \beta &= \beta + \beta^2 + \beta^{2^2} \end{aligned}$$

and so the density of the matrix $T_{\mathcal{N}}$ is $d_{\mathcal{N}} = 15$.

1.3 Construction of the optimal normal bases

We will now consider one construction of the optimal normal bases with the help of roots of unity and cyclotomic polynomials, for this reason we review some classical results:

Definition 1.7 (see [10]). *Let n be a positive integer and let $K^{(n)}$ the splitting field of $x^n - 1$ over a field K. The roots of $x^n - 1$ in $K^{(n)}$ are called the n-th roots of unity over K and the set of this roots is denoted by $E^{(n)}$.*

Definition 1.8 (see [10, 14]). *Let K a field of characteristic p and n a positive integer not divisible by p. An element ς in a extension field of K is called a primitive n-th root of unity if $\varsigma^n = 1$ and $\varsigma^s \neq 1$ for $0 < s < n$.*

Thus the *primitive n-th root of unity* over K is a generator of the cyclic group $E^{(n)}$.

Under the same condition of Definition 1.8 the polynomial

$$\Phi_n(x) = \prod_{\substack{s=1 \\ gcd(s,n)=1}}^{n} (x - \varsigma^s)$$

is called the *n-th cyclotomic polynomial* over K and $\varsigma^0, \varsigma^1, \ldots, \varsigma^{\varphi(n)}$ are the $\varphi(n)$ *primitive n-th roots of unity.*

Moreover every *n-th root of unity* over K is a *primitive d-th root of unity* over K, for exactly one positive divisor d of n.

In fact, let ς be a *primitive n-th root of unity* over K and let ς^s be an arbitrary *n-th root of unity* over K, then $d = n/gcd(s, n)$ is the order of ς^s, but since:

$$x^n - 1 = \prod_{s=1}^{n} (x - \varsigma^s)$$

If we collect the factors $(x - \varsigma^s)$ for which ς^s is a primitive dth root of unity over K, we obtain the formula:

$$x^n - 1 = \prod_{d|n} Q_d(x)$$

and so it follows that

$$\Phi_n(x) = \frac{x^n - 1}{\prod_{\substack{d|n \\ d \neq n}} \Phi_d(x)}$$

Theorem 1.3 (see [10, 13]). *Let p be a prime power and let n be a positive integer such that $gcd(n,p) = 1$. Let d be the last positive integer such that $p^d \equiv 1 \pmod{n}$. Then $\mathbb{F}_{p^d}$ is the splitting field of Φ_n and Φ_n is the product of $\phi(n)/d$ distinct monic irreducible polynomials of degree d.*

Proof . Let α be any primitive n-th root of unity over $\mathbb{F}_p$. If $\alpha \in \mathbb{F}_{p^k}$, $\alpha^{p^k} = \alpha$ and so $p^k \equiv 1 \pmod{n}$. Since d is the least positive integer exponent such that $p^d \equiv 1 \pmod{n}$ then k is a multiple of d and so $\alpha \in \mathbb{F}_{p^d}$.

Now, since n divides $p^d - 1$ then the cyclic group $\mathbb{F}_{p^d}^*$ has a unique subgroup of order n and so this subgroup contains all n-th roots of unity, i.e. all the roots of Φ_n are in $\mathbb{F}_{p^d}$, then $\mathbb{F}_{p^d}$ is the splitting field of Φ_n.

Since d is the least positive integer such that $\alpha \in \mathbb{F}_{p^d}$ then the minimal polynomial of α over $\mathbb{F}_p$ has degree d and because there are $\phi(n)$ primitive n-th roots of unity then Φ_n can be factorized in $\phi(n)/d$ distinct irreducible polynomials of degree d, each one is a minimal polynomial of a primitive n-th roots of unity □

We note that from the last theorem it follows that

Theorem 1.4. *Φ_n is irreducible over $\mathbb{F}_p$ if and only if $q^{\phi(n)} \equiv 1 \pmod{n}$.*

Then from this we know when Φ_n is irreducible over a finite field and we shall use these results in the next theorem that describes some optimal normal bases.

Theorem 1.5 (Optimal normal bases (2), see [14]). *Let $n+1$ be a prime and p be a primitive root modulo $n+1$ (i.e. p modulo $n+1$ generates $\mathbb{F}_{n+1}^*$). Let ς be a primitive $(n+1)$-th root of unity in an extension field of $\mathbb{F}_p$ and $\alpha = \varsigma$. Then $N = \{\alpha, \alpha^p, \alpha^{p^2}, \ldots, \alpha^{p^{n-1}}\}$ is an optimal normal basis of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$.*

Proof . From Theorem 1.2 and Theorem 1.3 we have that since p is a primitive root in $\mathbb{F}_{n+1}$, the minimal polynomial of α is

$$\Phi_{n+1} = \frac{(X^{n+1} - 1)}{(X - 1)} = X^n + X^{n-1} + \ldots + 1 \in \mathbb{F}_p[X]$$

and the $(n+1)$-th roots of unity are linearly indipendent. Thus the set of the conjugates of α, $\mathcal{N} = \{\alpha,\ \alpha^p,\ ...,\ \alpha^{p^n}\}$, is a normal basis for $\mathbb{F}_{p^n}$. This basis, as we have shown, is equal to the basis $\{\alpha,\ \alpha^2,\ ...\ ,\ \alpha^n\}$.

Now we denote with $\mathcal{M} = \{\alpha \cdot \alpha^i : i = 1,\ ...\ , n\}$ the set of rows of $T_{\mathcal{N}}$ and we note that the number of nonzero terms in the matrix $T_{\mathcal{N}}$ is also the number of nonzero terms in the expansion of the set $\mathcal{M}$ in the basis $\mathcal{N}$.

Thus we have

$$\alpha \cdot \alpha^i = \alpha^{i+1} = \alpha^j \text{ for } 1 \leq j < n$$

and

$$\alpha \cdot \alpha^n = 1 = -\sum_{1 \leq i \leq n} \alpha^i$$

Therefore there are exactly $(n-1)+n = 2n-1$ nonzero terms in $T_{\mathcal{N}}$ and so $\mathcal{N}$ is an optimal normal basis □

Theorem 1.6 (see [4]). *Let $n+1$ be a prime, let p be a primitive root modulo $n+1$ so that $\alpha^i = \alpha^{i \pmod{n+1}}$. The basis $\{\alpha,\ \alpha^p,\ \alpha^{p^2},\ ...\ ,\ \alpha^{p^{n-1}}\}$ for $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$ is the same as the basis $\{\alpha,\ \alpha^2,\ ...\ ,\ \alpha^n\}$ for $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$.*

Proof . It is sufficient to show that $\{1,\ p,\ p^2,\ ...\ ,\ p^{n-1}\} = \{1,\ 2,\ ...\ ,\ n\}$ in $\mathbb{F}_{n+1}$. Suppose $0 \leq j \leq i \leq n-1$ and $p^i \equiv p^j \pmod{n+1}$.

Then $p^j(p^{i-j}-1) \equiv 0 \pmod{n+1}$ and so $p^{i-j} \equiv 1 \pmod{n+1}$. Furthermore the order of p, that is n, divides $i-j$.

But $i-j < n$ then $i = j$.

Hence $\{1,\ p,\ p^2,\ ...\ ,\ p^{n-1}\}$ are all distinct and $\{1,\ p,\ p^2,\ ...\ ,\ p^{n-1}\} = \{1,\ 2,\ ...\ ,\ n\}$ □

We note, also, that the basis $\{\alpha,\ \alpha^2,\ ...\ ,\ \alpha^n\}$ is different but similar to the traditional power basis $\{1,\ \alpha,\ ...\ ,\alpha^{n-1}\}$ for $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$. For switch quickly between these two bases we use the identities:

$$\alpha^n = -\alpha^{n-1} - \alpha^{n-2} - ... - \alpha - 1 \text{ and } \alpha^0 = -\alpha^n - \alpha^{n-1} - ... - \alpha$$

Moreover we can reduce the exponents of α that are $> n$ using the relation $\alpha^i = \alpha^{i \pmod{n+1}}$ and the relation $\alpha^0 = -\alpha^n - \alpha^{n-1} - ... - \alpha$ can be used

for exponents of α that are zero modulo $n+1$. Thus as a consequence of the optimality of the bases the reduction stage of the multiplication in $\mathbb{F}_{p^n}$ requires only $2n-1$ additions in $\mathbb{F}_p$ (see [7]).

Chapter 2

Arithmetic operations in $\mathbb{F}_{p^2}$

Now we describe how the computations in $\mathbb{F}_{p^2}$ can be done efficiently; for this purpose a rapresentation of $\mathbb{F}_{p^2}$ is needed that allows efficient arithmetic operations.

Let $p \equiv 2 \pmod 3$, it follows that $(X^3 - 1)/(X - 1) = X^2 + X + 1$ is irriducibele over $\mathbb{F}_p$.

In fact we have that one root of the polynomial has the form

$$\alpha = \frac{-1 + \sqrt{-3}}{2}$$

and so $X^2 + X + 1$ is irriducible if and only if -3 is a quadratic non-residue modulo p. Using the Legendre symbol[1] it follows that:

$$\left(\frac{-3}{p}\right) = \left(\frac{-1}{p}\right) \cdot \left(\frac{3}{p}\right) = (-1)^{\frac{p-1}{2}} \cdot \left(\frac{3}{p}\right)$$

then

[1] We recall that *if a is an integer and $p > 2$ a prime, we define the* ***Legendre symbol*** $\left(\frac{a}{p}\right)$ *to equal* $0, 1$ *or* -1*, as follows:*

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{if } p|a \\ 1 & \text{if } a \text{ is a quadratic residue} \pmod p \\ -1 & \text{if } a \text{ is a nonresidue} \pmod p \end{cases}$$

(see [5])

- if $\frac{p-1}{2}$ is even $\Rightarrow p \equiv 1 \pmod 4$ and so $(-1)^{\frac{p-1}{2}} \cdot \left(\frac{3}{p}\right) = -1$
- if $\frac{p-1}{2}$ is odd $\Rightarrow p \equiv 3 \pmod 4$ and so $(-1)^{\frac{p-1}{2}} \cdot \left(\frac{3}{p}\right) = (-1) \cdot (1) = -1$

Therefore -3 is a quadratic non-residue modulo p and $X^2 + X + 1$ is irriducible over $\mathbb{F}_p[X]$, so $\mathbb{F}_{p^2} \cong \mathbb{F}_p[X]/(X^2 + X + 1)$.

Moreover, since $p \equiv 2 \pmod 3$, p is a primitive root modulo 3, i.e. $p \pmod 3$ generates $\mathbb{F}_3^*$. Then considering Theorem 1.5 the zeros α and α^p of the polynomial $(X^3 - 1)/(X - 1) = X^2 + X + 1$ form an optimal normal basis for $\mathbb{F}_{p^2}$ over $\mathbb{F}_p$, i.e.

$$\mathbb{F}_{p^2} \cong \{x_1\alpha + x_2\alpha^p : x_1, x_2 \in \mathbb{F}_p\}.$$

From $\alpha^i = \alpha^{i \pmod 3}$ it follows that $\alpha^p = \alpha^{2+3k} = \alpha^2 \cdot 1^k = \alpha^2$ and so

$$\mathbb{F}_{p^2} \cong \left\{x_1\alpha + x_2\alpha^2 : \ \alpha^2 + \alpha + 1 = 0 \ , \ x_1, \ x_2 \in \mathbb{F}_p\right\}.$$

Note that in this representation of $\mathbb{F}_{p^2}$ an element t of $\mathbb{F}_p$ is represented by $-t\alpha - t\alpha^2$, e.g. 3 is represented by $-3\alpha - 3\alpha^2$ (see [8]).

Arithmetic operations in $\mathbb{F}_{p^2}$ are carried out as follows.

Lemma 2.1 (see [8]). *Let $x, y, z \in \mathbb{F}_{p^2}$ with $p \equiv 2 \pmod 3$ prime. Not couting additions or subtractions in $\mathbb{F}_p$:*

i. Computing x^p is for free.

ii. Computing x^2 takes two multiplications in $\mathbb{F}_p$.

iii. Computing $x \cdot y$ takes three multiplications in $\mathbb{F}_p$.

iv. Computing $x \cdot z - y \cdot z^p$ takes four multiplications in $\mathbb{F}_p$.

Proof . Let $x = x_1\alpha + x_2\alpha^2 \in \mathbb{F}_{p^2}$ we have that:

$$x^p = x_1^p\alpha^p + x_2^p\alpha^{2p} = x_2\alpha + x_1\alpha^2$$

It follows that p^{th} powering in $\mathbb{F}_{p^2}$ does not require arithmetic operations and thus can be considered to be for free, this prove *i.*

For squaring, we write:

$$(x_1\alpha \ + \ x_2\alpha^2)^2 = x_1^2\alpha^2 \ + \ x_2^2\alpha^4 \ + \ 2x_1x_2\alpha^3 \ (*)$$

because $\alpha^3 \equiv 1 \equiv -\alpha \ -\alpha^2$ and $\alpha^4 \equiv \alpha$ both modulo $\alpha^2 + \alpha + 1$ it follows

$$(*) = x_1^2\alpha^2 + x_2^2\alpha \ - \ 2x_1x_2\alpha \ - \ 2x_1x_2\alpha^2 = x_1(x_1 \ - \ 2x_2)\alpha^2 \ + \ x_2(x_2 \ - \ 2x_1)\alpha$$

that prove *ii*.

Let $x = x_1\alpha \ + \ x_2\alpha^2$ and $y = y_1\alpha \ + \ y_2\alpha^2 \in \mathbb{F}_{p^2}$ and we compute

$$\begin{aligned} x \cdot y &= (x_1\alpha \ + \ x_2\alpha^2) \cdot (y_1\alpha \ + \ y_2\alpha^2) = \\ &= (x_1y_1\alpha^2 \ + \ x_1y_2 \ + \ x_2y_1 \ + \ x_2y_2\alpha) = \\ &= (x_2y_2 \ - \ x_1y_2 \ - \ x_2y_1)\alpha \ + \ (x_1y_1 \ - \ x_1y_2 \ - \ x_2y_1)\alpha^2 \end{aligned}$$

using Karatsuba method (see Appendix A), first compute $x_1 \cdot y_1$, $x_2 \cdot y_2$ and $(x_1 \ + \ x_2)(y_1 \ + \ y_2)$, after which $x_1 \cdot y_2 \ + \ x_2 \cdot \ y_1$ and thus $x \cdot y$ follow using four subtractions.

Finally let $x = x_1\alpha \ + \ x_2\alpha^2, \ y = y_1\alpha \ + \ y_2\alpha^2, \ z = z_1\alpha \ + \ z_2\alpha^2 \in \mathbb{F}_{p^2}$, it follows that:

$$\begin{aligned} & x \cdot z \ - \ y \cdot z^p = \\ &= (x_1\alpha \ + \ x_2\alpha^2) \cdot (z_1\alpha \ + \ z_2\alpha^2) \ - \ (y_1\alpha \ + \ y_2\alpha^2) \cdot (z_1\alpha \ + \ z_2\alpha^2)^p = \\ &= (x_1\alpha \ + \ x_2\alpha^2) \cdot (z_1\alpha \ + \ z_2\alpha^2) \ - \ (y_1\alpha \ + \ y_2\alpha^2) \cdot (z_1^p\alpha^p \ + \ z_2^p\alpha^{2p}) = \\ &= (x_1\alpha \ + \ x_2\alpha^2) \cdot (z_1\alpha \ + \ z_2\alpha^2) \ - \ (y_1\alpha \ + \ y_2\alpha^2) \cdot (z_1\alpha^2 \ + \ z_2\alpha) \end{aligned}$$

with $\alpha^2 \ + \ \alpha \ + \ 1 \equiv 0$ it follows

$$\begin{aligned} &= x_1 \cdot z_1\alpha^2 \ + \ x_1 \cdot z_2 \ + \ x_2 \cdot z_1 \ + \ x_2 \cdot z_2\alpha \ - \\ &\quad - \ y_1 \cdot z_1 \ - \ y_1 \cdot z_2\alpha^2 \ - \ y_2 \cdot z_1\alpha \ - \ y_2 \cdot z_2 \ = \\ &= x_1 \cdot z_1\alpha^2 \ + \ x_1 \cdot z_2 \cdot (-\alpha \ - \alpha^2) + \ x_2 \cdot z_1 \cdot (-\alpha \ - \alpha^2) + \ x_2 \cdot z_2\alpha \ - \\ &\quad - y_1 \cdot z_1 \cdot (-\alpha \ - \alpha^2) - \ y_1 \cdot z_2\alpha^2 \ - \ y_2 \cdot z_1\alpha \ - \ y_2 \cdot z_2 \cdot (-\alpha \ - \alpha^2) \ = \\ &= (z_1 \cdot (y_1 - x_2 - y_2) \ + \ z_2 \cdot (x_2 - x_1 + y_2))\alpha \ + \\ &\quad + \ (z_1 \cdot (x_1 - x_2 + y_1) \ + \ z_2 \cdot (y_2 - x_1 - y_1)\alpha^2) \end{aligned}$$

that prove *iv.* □

For comparison purposes we review the following results.

Lemma 2.2 (see [9]). [2]

Let $x, y, z \in \mathbb{F}_{p^6}$ with $p \equiv 5 \pmod 7$, and let $s, t \in \mathbb{Z}$ with $0 < s, t < p$.

i. Computing $x \cdot y$ takes 18 multiplications in $\mathbb{F}_p$.

ii. Computing x^2 takes 18 squarings in $\mathbb{F}_p$.

iii. Computing x^s takes an expected $9 \log_2 s$ multiplications and $18 \log_2 s$ squarings in $\mathbb{F}_p$.

iv. Computing $x^s \cdot x^t$ takes an expected $13.5 \log_2 \max(s,t)$ multiplications and $18 \log_2 \max(s,t)$ squarings in $\mathbb{F}_p$.

Proof . Since $p \equiv 5 \pmod 7$, p generates $\mathbb{F}_7^*$ and $\Phi_7(X) = X^6 + X^5 + X^4 + X^3 + X^2 + X + 1$. Let α denote a root of $\Phi_7(X)$, then $\mathbb{F}_{p^6}$ can be represented using the optimal normal bases $\{\alpha,\ \alpha^2,\ \ldots, \alpha^6\}$ over $\mathbb{F}_p$.

So multiplication and squarings in $\mathbb{F}_{p^6}$ can be done in 18 multiplications and squarings in $\mathbb{F}_p$ respectively, rather than 36 as it is required by the classic algorithm for the multiplications.

In fact, let

$$x = \sum_{i=0}^{5} x_i \alpha^{i+1} \text{ and } y = \sum_{j=0}^{5} y_j \alpha^{j+1}$$

using the Karatsuba algorithm, with proper adjustment of the powers of α, i.e.

$$x = X_0\ \alpha + X_1\ \alpha^4 \text{ and } y = Y_0 \alpha + Y_1 \alpha^4$$

where $X_0,\ X_1,\ Y_0,\ Y_1$ are the second degree polynomials, we have that:

$$\begin{aligned} x \cdot y &= (X_0\alpha\ +\ X_1\alpha^4) \cdot (Y_0\alpha\ +\ Y_1\alpha^4) \\ &= X_1Y_1\alpha^8\ +\ (X_0Y_1\ +\ X_1Y_0)\alpha^5\ +\ X_0Y_0\alpha^2 \end{aligned}$$

[2] We remark that in the article [9] p is 2 (mod 3).

Instead, our choice to take $p \equiv 2 \pmod 7$ is a "trick" to have an optimal normal basis, as confirmed by a e-mail sent us by Martijn Stam, a A. K. Lenstra's student.

so that with $z_0 = X_0Y_0,\ z_1 = X_1Y_1$ and $z_2 = (X_0 + X_1)\cdot(Y_0 + Y_1)$

$$x \cdot y = z_1\alpha^8 + (z_2 - z_1 - z_0)\alpha^5 + z_0\alpha^2$$

Each of this z_i can be computed using 6 multiplication, in fact, because $X_0 = x_0 + x_1\alpha + x_2\alpha^2$ and $Y_0 = y_0 + y_1\alpha + y_2\alpha^2$ we have that:

$$\begin{aligned} z_0 = X_0Y_0 = \\ = x_0y_0 + (x_0y_1 + x_1y_0)\alpha + (x_0y_2 + x_1y_1 + x_2y_0)\alpha^2 + \\ + (x_1y_2 + x_2y_1)\alpha^3 + x_2y_2\alpha^4 \end{aligned}$$

so that using Karatsuba method again, we set:

$$\begin{aligned} &\overline{z}_0 = x_0y_0,\ \overline{z}_1 = x_1y_1,\ \overline{z}_2 = x_2y_2, \\ &\overline{z}_3 = (x_0 + x_1)\cdot(y_0 + y_1), \\ &\overline{z}_4 = (x_0 + x_2)\cdot(y_0 + y_2), \\ &\overline{z}_5 = (x_1 + x_2)\cdot(y_1 + y_2) \end{aligned}$$

thus

$$\begin{aligned} z_0 = \overline{z}_0 + (\overline{z}_3 - \overline{z}_0 - \overline{z}_1)\alpha + (\overline{z}_1 + \overline{z}_4 - \overline{z}_0 - \overline{z}_2)\alpha^2 + \\ + (\overline{z}_5 - \overline{z}_1 - \overline{z}_2)\alpha^3 + \overline{z}_2\alpha^4 \end{aligned}$$

z_1 and z_2 can be computed in a similar way. It follows that 18 multiplications (or squarings) are suffice to compute $x \cdot y$ (or x^2) and this proves *i* and *ii*.

For *iii* we use the ordinary square and multiply method. Therefore we get $\log_2 s$ squarings and expected $0.5\log_2 s$ multiplications in $\mathbb{F}_{p^6}$ and so considering the result of *i* and *ii* we have $9\log_2 s$ multiplications and $18\log_2 s$ squarings in $\mathbb{F}_p$.

For *iv* we use standard multi-exponentiation (see Appendix B) and so we obtain $\log_2(\max(s,t))$ squarings and $0.75\log_2(\max(s,t))$ multiplications in $\mathbb{F}_{p^6}$ which are equivalent to $13.5\log_2(\max(s,t))$ multiplications and $18\log_2(\max(s,t))$ squarings in $\mathbb{F}_p$. □

Chapter 3

XTR supergroup and XTR subgroup

The XTR work environment is the multiplicative group of the finite field $\mathbb{F}_{p^6}$. This cyclic group has order $p^6 - 1$, therefore only one cyclic subgroup with order $p^2 - p + 1$ exsist (see [8]).

The latter group is reffered to as the *XTR supergroup,* $\boldsymbol{X}$. Because $p^2 - p + 1$ not divide any $p^s - 1$ for $s = 1, 2, 3$, $\boldsymbol{X}$ cannot be embedded in the multiplicative group of any true subfield of $\mathbb{F}_{p^6}$.

Let g be an element of order $q > 3$ dividing $p^2 - p + 1$. The order q subgroup $\boldsymbol{Y} = \langle g \rangle$ genereted by g is referred to as *XTR subgroup* (see [8]).

We show, here below, that arbitrary powers of g can be represented using a single element of the subfield $\mathbb{F}_{p^2}$, and that such powers can be computed efficiently using arithmetic operations in $\mathbb{F}_{p^2}$, avoiding arithmetic in $\mathbb{F}_{p^6}$ (see [9]).

3.1 Traces

Definition 3.1. *The trace* $Tr(h)$ *over* $\mathbb{F}_{p^2}$ *of* $h \in \mathbb{F}_{p^6}$ *is the sum of the conjugates over* $\mathbb{F}_{p^2}$ *of* h*, i.e.,*

$$Tr(h) = h + h^{p^2} + h^{p^4}$$

Because $h \in \mathbb{F}_{p^6}$, the order of h divides $p^6 - 1$ and so $h^{p^6} = h^{p^6-1} \cdot h = h$, it follows that

$$Tr(h)^{p^2} = (h + h^{p^2} + h^{p^4})^{p^2} = h^{p^2} + h^{p^4} + h^{p^6} = h^{p^2} + h^{p^4} + h = Tr(h)$$

so that $Tr(h) \in \mathbb{F}_{p^2}$ (see [8, 9]).

Proposition 3.1. *The function Tr from $\mathbb{F}_{p^6}$ to $\mathbb{F}_{p^2}$ is an $\mathbb{F}_{p^2}$-linear application, i.e.*

$$Tr(h + k) = Tr(h) + Tr(k)$$

$$Tr(c \cdot h) = c \cdot Tr(h)$$

Proof . Let $c \in \mathbb{F}_{p^2}$ such that $c^{p^2} = c$, and let $h, k \in \mathbb{F}_{p^6}$ we have that

$$\begin{aligned}
Tr(h \ + \ k) &= (h \ + \ k) \ + \ (h \ + \ k)^{p^2} \ + \ (h \ + \ k)^{p^4} \\
&= h \ + \ k \ + \ h^{p^2} + \ k^{p^2} + \ h^{p^4} + \ k^{p^4} \\
&= Tr(h) \ + \ Tr(k) \\
Tr(c \ \cdot \ h) &= c \ \cdot \ h \ + \ c^{p^2} \ \cdot \ h^{p^2} \ + \ c^{p^4} \ \cdot \ h^{p^4} \\
&= c \ \cdot \ h \ + \ c \ \cdot \ h^{p^2} \ + \ c \ \cdot \ h^{p^4} \\
&= c \ \cdot \ (\ h \ + \ h^{p^2} \ + \ h^{p^4}) \\
&= c \ \cdot \ Tr(h)
\end{aligned}$$

□

Now let $g \in \mathbb{F}_{p^6}^*$ be an element of order $q > 3$ and such that q divides $p^2 - p + 1$. The conjugates of g which order divide $p^2 - p + 1$ are g, $g^{p^2} = g^{p-1}$, since $p^2 \equiv p - 1 \pmod{p^2 - p + 1}$, $g^{p^4} = g^{-p}$, since $p^4 \equiv (p-1)^2 \equiv p^2 - 2p + 1 \equiv p - 1 - 2p + 1 \equiv -p \pmod{p^2 - p + 1}$ and so

$$Tr(g) = g \ + \ g^{p^2} \ + \ g^{p^4} = g \ + \ g^{p-1} \ + \ g^{-p}$$

Mereover, since the order of g divides $p^6 - 1$ the trace over $\mathbb{F}_{p^2}$ of g equals the trace of the conjugates over $\mathbb{F}_{p^2}$ of g:

$$Tr(g) = Tr(g^{p^2}) = Tr(g^{p^4})$$

From this last result and since in XTR elements of $\boldsymbol{X}$ are represented by their trace over $\mathbb{F}_{p^2}$, it follows that XTR makes no distinction between an element of $\boldsymbol{Y} = \langle g \rangle$ and its conjugates over $\mathbb{F}_{p^2}$ (see [16]).

3.2 The polynomials $F(c, X)$

Definition 3.2 (see [9]). *For $c \in \mathbb{F}_{p^2}$ define the polynomial*

$$F(c, X) = X^3 - cX^2 + c^p X - 1 \in \mathbb{F}_{p^2}[X]$$

with (not necessarily distinct) roots $h_0, h_1, h_2 \in \mathbb{F}_{p^6}[X]$, and define

$$c_n = h_0^n + h_1^n + h_2^n \text{ for } n \in \mathbb{Z}.$$

Now we considered c of the form $Tr(g)$ for g of order > 3 and dividing $p^2 - p + 1$. We have the following result:

Theorem 3.1 (see [9]). *The roots of $X^3 - Tr(g)X^2 + Tr(g)^p X - 1$ are the conjugates of g.*

Proof . We compare the coefficients of $X^3 - Tr(g)X^2 + Tr(g)^p X - 1$ with the coefficients of the polynomial $(X - g)(X - g^{p-1})(X - g^{-p})$.

The coefficients of X^2 follows from $g + g^{p-1} + g^{-p} = Tr(g)$.

The coefficients of X equals:

$$g \cdot g^{-p} + g \cdot g^{p-1} + g^{-p} \cdot g^{p-1} = g^p + g^{1-p} + g^{-1}$$

Since $1 - p \equiv -p^2 \mod p^2 - p + 1$ and $-1 \equiv p^2 - p \mod p^2 - p + 1$ we find that:

$$g^p + g^{1-p} + g^{-1} = g^p + g^{-p^2} + g^{p^2-p} = (g + g^{-p} + g^{p-1})^p = Tr(g)^p$$

The constant coefficients is the product of the conjugates:

$$g \cdot g^{-p} \cdot g^{p\,-\,1} = g^{1\,-\,p\,+\,p\,-\,1} = g^0 = 1$$

which completes the proof. □

Therefore

$$(X\,-\,g)(X-\,g^{p\,-\,1})(X\,-\,g^{-p}) = X^3\,-\,Tr(g)X^2\,+\,Tr(g)^pX\,-\,1 \in \mathbb{F}_{p^2}[X]$$

is actually the minimal polynomial of g over $\mathbb{F}_{p^2}$ and so this polynomial and the conjugates of g are fully determined by $Tr(g) \in \mathbb{F}_{p^2}$.

The same holds for any power of g, in fact for any integer n the conjugates of g^n are the roots of $X^3\,-\,Tr(g^n)X^2\,+\,Tr(g^n)^pX\,-\,1 \in \mathbb{F}_{p^2}[X]$, and the latter polynomial and the conjugates of g^n are determined by $Tr(g^n) \in \mathbb{F}_{p^2}$.

This observation is useful for cryptographic purposes if there is a way to efficiently compute $Tr(g^n)$ given $Tr(g)$. In cryptographic protocols, then, $g^n \in \mathbb{F}_{p^6}$ can be replaced by $Tr(g^n) \in \mathbb{F}_{p^2}$, thereby obtaining a saving of a factor 3 in the rapresentation size (see [8]).

We will show that $Tr(g^n)$ can indeed be computed quickly given $Tr(g)$, so that a considerable speed advantage is obtained.

For this purpose now we consider the properties of the polynomial $X^3\,-\,cX^2\,+\,c^pX\,-1$ for general $c \in \mathbb{F}_{p^2}$ as in Definition 3.2. .

3.3 Properties of $F(c, X)$ and its roots

Lemma 3.1. *"Properties of $F(c, X)$, see [9]"*

i. $F(c, h_j^{-p}) = 0$ *for* $j = 0, 1, 2$.

ii. Either all h_j *have order dividing* $p^2\,-\,p\,+\,1$ *and* > 3 *or all* $h_j \in \mathbb{F}_{p^2}$.

Proof *i.* From $F(c, h_j) = h_j^3\,-\,ch_j^2\,+\,c^ph_j\,-\,1 = 0$ it follows that $h_j \neq 0$

and that

$$\begin{aligned} 0 = F(c,h_j)^p = \\ = h_j^{3p} \ - \ c^p h_j^{2p} \ + \ c^{p^2} h_j^p \ - \ 1 = \\ = -h_j^{3p} \cdot \ (-1 \ + \ c^p h_j^{-p} \ - \ c h_j^{-2p} \ + \ h_j^{-3p}) = \\ = -h_j^{3p} \ \cdot F(c, h_j^{-p}) \end{aligned}$$

where $h_j \neq 0$ and $c^{p^2} = c$. Therefore $F(c, h_j^{-p}) = 0$ and thus for all root h_j, its power h_j^{-p} is also root of the polynomial $F(c, X)$.

Proof *ii.* From i it follows, without loss of generality, that we have three possibility:

1. $h_j = h_j^{-p}$ for $j = 0, 1, 2$.
2. $h_0 = h_0^{-p}, h_1 = h_2^{-p}, h_2 = h_1^{-p}$.
3. $h_j = h_{j+1 \mod 3}^{-p}$ for $j = 0, 1, 2$.

In the first case all h_j have order dividing $p+1$ and thus in $\mathbb{F}_{p^2}$, in fact we have:

$$h_j^{p+1} = h_j \cdot h_j^p = h_j \cdot h_j^{-1}$$

In the second case, h_0 has order dividing $p+1$ because $h_0 = h_0^{-p}$. While $h_1 = h_2^{-p} = h_1^{p^2}$ and $h_2 = h_1^{-p} = h_2^{p^2}$, so that h_1 and h_2 both have orders dividing $p^2 - 1$ and thus $h_1, h_2 \in \mathbb{F}_{p^2}$.

In the last case it follows from $1 = h_0 \cdot \ h_1 \cdot \ h_2$ that

$$1 = \ h_0 \cdot h_1 \cdot h_2 = h_0 \cdot h_2^{-p} \cdot h_0^{-p} = h_0 \cdot h_0^{p^2} \cdot h_0^{-p} = h_0^{p^2 \ - \ p \ + \ 1}$$

so that h_0 and similarly h_1 and h_2 have order dividing $p^2 \ - \ p \ + \ 1$

Moreover if either one, say h_0, has order at most 3, then h_0 has order 1 or 3 since $p^2 \ - \ p \ + \ 1$ is odd.

Since 3 divide $p^2 - 1$, in fact $p^2 \equiv 1 \mod 3$, it follows that the order of h_0 divides $p^2 - 1$ so that $h_0 \in \mathbb{F}_{p^2}$. But then h_1 and h_2 are in $\mathbb{F}_{p^2}$ as well, because they are power of h_0.

It follows that while in the first and second case the roots are in $\mathbb{F}_{p^2}$, in the last case either all h_j have order dividing $p^2 - p + 1$ and > 3, i.e. $h_j \in \boldsymbol{X}$ for $j = 0, 1, 2$, or all h_j are in $\mathbb{F}_{p^2}$.

Recall that, as in Definition 3.2, $c_n = h_0^n + h_1^n + h_2^n$. Now we show its properties.

Lemma 3.2. *"Properties of c_n, see [9]"*

i. $c = c_1$.

ii. $c_{-n} = h_0^n \cdot h_1^n + h_0^n \cdot h_2^n + h_1^n \cdot h_2^n$ *for* $n \in \mathbb{Z}$.

iii. $c_{-n} = c_{np} = c_n^p$ *for* $n \in \mathbb{Z}$.

iv. $c_n \in \mathbb{F}_{p^2}$ *for* $n \in \mathbb{Z}$.

v. $c_{u+v} = c_u \cdot c_v - c_v^p \cdot c_{u-v} + c_{u-2v}$ *for* $u, v \in \mathbb{Z}$.

Proof . The proof follows from the definition of c_n in Definition 3.2, in fact it follows that

$$c_1 = h_0 + h_1 + h_2 = c$$

For *ii*, because $h_0 \cdot h_1 \cdot h_2 = 1$ it follows that

$$\begin{aligned} h_1 \cdot h_2 = h_0^{-1} &\Rightarrow (h_1 \cdot h_2)^n = h_0^{-n} \\ h_0 \cdot h_2 = h_1^{-1} &\Rightarrow (h_0 \cdot h_2)^n = h_1^{-n} \\ h_0 \cdot h_1 = h_2^{-1} &\Rightarrow (h_0 \cdot h_1)^n = h_2^{-n} \end{aligned}$$

therefore

$$\begin{aligned} (h_1 \cdot h_2)^n + (h_0 \cdot h_2)^n + (h_0 \cdot h_1)^n &= h_1^n \cdot h_2^n + h_0^n \cdot h_2^n + h_0^n \cdot h_1^n = \\ &= h_0^{-n} + h_1^{-n} + h_2^{-n} = c_{-n} \end{aligned}$$

Now we consider the three cases of the proof of *ii* in Lemma 3.1 (Properties of $F(c, X)$):

1. If $h_j = h_j^{-p}$ for $j = 0, 1, 2$ then $h_j^{np} = h_j^{-n}$ and thus $c_{-n} = h_0^{-n} + h_1^{-n} + h_2^{-n} = h_0^{np} + h_1^{np} + h_2^{np} = c_{np}$.

2. If $h_0 = h_0^{-p}, h_1 = h_2^{-p}$ and $h_2 = h_1^{-p}$ then $c_{-n} = h_0^{-n} + h_1^{-n} + h_2^{-n} = h_0^{np} + h_1^{np} + h_2^{np} = c_{np}$.

3. if $h_j = h_{j+1 \bmod 3}^{-p}$ for $j = 0, 1, 2$ then $c_{-n} = h_0^{-n} + h_1^{-n} + h_2^{-n} = h_1^{np} + h_2^{np} + h_0^{np} = c_{np}$.

Moreover $c_{np} = (h_0^n + h_1^n + h_2^n)^p = c_n^p$ that proves *iii*.

If all $h_j \in \mathbb{F}_{p^2}$, then *iv* is immediate. Otherwise, it follows from Lemma 3.1.ii that[1] $F(c, X) \in \mathbb{F}_{p^2}[X]$ is irreducible if and only if its roots have order dividing $p^2 - p + 1$ and are > 3, so we have that $F(c, X)$ is irreducible and its roots are the conjugates of h_0. Thus $c_n = Tr(h_0^n) \in \mathbb{F}_{p^2}$, which concludes the proof of *iv*.

By the definition of c_n, $c_{-n} = c_n^p$ (cf. *Property iii*) and the relationship $h_0^n + h_1^n + h_2^n = 1$, the proof of *v* follows from a staightforward computation.

In fact we have that

$$\begin{aligned} & c_u \cdot c_v - c_v^p \cdot c_{u-v} + c_{u-2v} = \\ &= (h_0^u + h_1^u + h_2^u) \cdot (h_0^v + h_1^v + h_2^v) - (h_0^v + h_1^v + h_2^v)^p \cdot \\ & \cdot (h_0^{u-v} + h_1^{u-v} + h_2^{u-v}) + (h_0^{u-2v} + h_1^{u-2v} + h_2^{u-2v}) = \end{aligned}$$

By the relationship $c_{-n} = c_n^p$ it follows

$$\begin{aligned} &= h_0^{u+v} + h_1^{u+v} + h_2^{u+v} + h_0^u \cdot h_1^v + h_0^u \cdot h_2^v + h_1^u \cdot h_0^v + h_1^u \cdot h_2^v + \\ &\quad + h_2^u \cdot h_0^v + h_2^u \cdot h_1^v - (h_0^{-v} + h_1^{-v} + h_2^{-v}) \cdot (h_0^{u-v} + h_1^{u-v} + h_2^{u-v}) + \\ &\quad + (h_0^{u-2v} + h_1^{u-2v} + h_2^{u-2v}) = \\ &= h_0^{u+v} + h_1^{u+v} + h_2^{u+v} + h_0^u \cdot h_1^v + h_0^u \cdot h_2^v + h_1^u \cdot h_0^v + h_1^u \cdot h_2^v + \\ &\quad + h_2^u \cdot h_0^v + h_2^u \cdot h_1^v - h_0^{u-2v} - h_1^{u-2v} - h_2^{u-2v} - h_0^{-v} \cdot h_1^{u-v} + \\ &\quad - h_0^{-v} \cdot h_2^{u-v} - h_1^{-v} \cdot h_0^{u-v} - h_1^{-v} \cdot h_2^{u-v} - h_2^{-v} \cdot h_0^{u-v} + \\ &\quad + h_2^{-v} \cdot h_1^{u-v} + h_0^{u-2v} + h_1^{u-2v} + h_2^{u-2v} = (*) \end{aligned}$$

[1] We remember that *The polynomial $f \in F[X]$ of degree 2 or 3 is irriducible in $F[X]$ if and only if f has no root in F (see [10], Theorem 1.69).*

Now if we use the relation $h_0^n + h_1^n + h_2^n = 1$, it follows that

$$\begin{aligned} -h_0^{u-2v} - h_1^{u-2v} - h_2^{u-2v} &= -1 \\ h_0^{u-2v} + h_1^{u-2v} + h_2^{u-2v} &= 1. \end{aligned}$$

Moreover

$$h_0^{-v} = h_1^v \cdot h_2^v,\ h_1^{-v} = h_0^v \cdot h_2^v,\ h_2^{-v} = h_0^v \cdot h_1^v$$

thus

$$\begin{aligned} (*) = {} & h_0^{u+v} + h_1^{u+v} + h_2^{u+v} + h_0^u \cdot h_1^v + h_0^u \cdot h_2^v + h_1^u \cdot h_0^v + h_1^u \cdot h_2^v + \\ & + h_2^u \cdot h_0^v + h_2^u \cdot h_1^v - h_1^u \cdot h_2^v - h_1^v \cdot h_2^u - h_0^u \cdot h_2^v - h_0^v \cdot h_2^u + \\ & + h_0^u \cdot h_1^v - h_0^v \cdot h_1^u = \\ = {} & h_0^{u+v} + h_1^{u+v} + h_2^{u+v} = c_{u+v}. \end{aligned}$$

□

Theorem 3.2 (see [9]). *$F(c_n, h_j^n) = 0$ for $j = 0, 1, 2$ and $n \in \mathbb{Z}$.*

Proof . We compute the coefficients of $(X - h_0^n)(X - h_1^n)(X - h_2^n)$ and compare this coefficients with the coefficients of the polynomial

$$F(c_n, X) = X^3 - c_n X^2 + c_n^p X - 1 \in \mathbb{F}_{p^2}.$$

We have:

$$\begin{aligned} & (X - h_0^n)(X - h_1^n)(X - h_2^n) = \\ = {} & (X^2 - h_1^n X - h_0^n X + h_0^n h_1^n)(X - h_2^n) = \\ = {} & X^3 - h_1^n X^2 - h_0^n X^2 - h_2^n X^2 + h_0^n h_1^n X + h_1^n h_2^n X + h_0^n h_2^n X + \\ & + h_0^n h_1^n h_2^n = \\ = {} & X^3 - (h_0^n + h_1^n + h_2^n) X^2 + (h_0^n h_1^n + h_0^n h_2^n + h_1^n h_2^n) X - h_0^n h_1^n h_2^n \end{aligned}$$

We find that the coefficient of X^2 equals $-(h_0^n + h_1^n + h_2^n) = -c_n$,the coefficient of X equals $(h_0^n h_1^n + h_0^n h_2^n + h_1^n h_2^n) = c_{-n} = c_n^p$ (cf. Lemma 3.2.iii) and finally the constant coefficient equals $-h_0^n h_1^n h_2^n = -(h_0 h_1 h_2)^n = -1$

Thus it follows that $(X - h_0^n)(X - h_1^n)(X - h_2^n) = F(c_n, X)$ and so $F(c_n, h_j^n) = 0$ for $j = 0, 1, 2$. □

Theorem 3.3 (see [9]). *$F(c, X)$ is riducible over $\mathbb{F}_{p^2}$ if and only if $c_{p+1} \in \mathbb{F}_p$.*

Proof .($\Rightarrow$) If $F(c, X)$ is riducible then all h_j are in $\mathbb{F}_{p^2}$ for $j = 0, 1, 2$ (Lemma 3.1.ii) (We remember the equivalence in proof *iv* of the (Lemma 3.2)).

It follows that

$(h_j^{p+1})^p = h_j^{p^2+p} = h_j^{p+1} \Rightarrow h_j^{p+1} \in \mathbb{F}_p$ for $j = 0, 1, 2$ and so also their sum $h_0^{p+1} + h_1^{p+1} + h_2^{p+1} = c_{p+1} \in \mathbb{F}_p$.

($\Leftarrow$) If $c_{p+1} \in \mathbb{F}_p$ then $c_{p+1}^p = c_{p+1}$ and $F(c_{p+1}, X) = X^3 - c_{p+1}X^2 + c_{p+1}X - 1$ thus 1 is root of $F(c_{p+1}, X)$, i.e. $F(c_{p+1}, 1) = 0$.

Since the roots of $F(c_{p+1}, X)$ are the $(p+1)^{th}$ powers of the roots of $F(c, X)$ (cf. Theorem 3.2), it follows that $F(c, X)$ has a root of order dividing $p + 1$ and so $p^2 - 1$, i.e. an element of $\mathbb{F}_{p^2}$, so that $F(c, X)$ is riducible over $\mathbb{F}_{p^2}$. □

We conclude this section observing that the Theorem 3.2 is fundamental. From this theorem follows that for $c_n = Tr(g^n)$ the conjugates of g^n are the roots of $X^3 - Tr(g^n)X^2 + Tr(g^n)^p X - 1 \in \mathbb{F}_{p^2}[X]$ and this polynomial with g^n's conjugates are fully determined by $Tr(g^n)$, as we claimed on the other hand with this result we have a compact rappresentation of every powers of g replacing $g^n \in \mathbb{F}_{p^6}^*$ with $Tr(g^n) \in \mathbb{F}_{p^2}$.

3.4 The computation of c_n

In the last section, we have show the properties of c_n, the same properties that lead to a fast algorithm to compute c_n for any $n \in \mathbb{Z}$.

In fact we have the following results:

Corollary 3.1 (see [9]). *Assume that c, c_{n-1}, c_n and c_{n+1} are given. Then*

i. Computing $c_{2n} = c_n^2 - 2c_n^p$ takes two multiplications in $\mathbb{F}_p$.

ii. Computing $c_{n+2} = c \cdot c_{n+1} \ - \ c^p \cdot c_n \ + \ c_{n-1}$ *takes four multiplications in* $\mathbb{F}_p$.

iii. Computing $c_{2n-1} = c_{n-1} \cdot c_n \ - \ c^p \cdot c_n^p \ + \ c_{n+1}^p$ *takes four multiplications in* $\mathbb{F}_p$.

iv. Computing $c_{2n+1} = c_{n+1} \cdot c_n \ - \ c \cdot c_n^p \ + \ c_{n-1}^p$ *takes four multiplications in* $\mathbb{F}_p$.

Proof . The identities follows from Lemma 3.2 *.iii, .iv* .

In fact from

$$c_{u+v} = c_u \cdot c_v \ - \ c_v^p \cdot c_{u-v} \ + \ c_{u-2v}$$

for i, let $u = v = n$ and $c_0 = 3$. We have

$$c_{2n} = c_n \cdot c_n \ - \ c_n^p \cdot 3 \ + \ c_{-n}$$

and since $c_{-n} = c_n^p$ following $c_{2n} = c_n^2 \ - \ 2c_n^p$.
For *ii*, let $u = n + 1$ and $v = 1$. We have

$$c_{n+2} = c_{n+1} \cdot c \ - \ c^p \cdot c_n \ + \ c_{n-1}$$

For *iii*, let $u = n - 1$ and $v = n$. We have

$$c_{2n-1} = c_{n-1} \cdot c_n \ - \ c_n^p \cdot c_{-1} \ + \ c_{-(n+1)}$$

and since $c_{-n} = c_n^p$ it follows $c_{2n-1} = c_{n-1} \cdot c_n \ - \ c \cdot c_n^p \ + \ c_{n-1}^p$.

Finally for *iv*, let $u = n + 1$ and $v = n$. We have

$$c_{2n+1} = c_{n+1} \cdot c_n \ - \ c_n^p \cdot c \ + \ c_{-(n-1)}$$

and since $c_{-n} = c_n^p$ it follows $c_{2n+1} = c_{n+1} \cdot c_n \ - \ c_n^p \cdot c \ + \ c_{(n-1)}^p$.

Now we compute the computational costs of these last four identities, considering the Lemma 2.1.

In the first identity, since $p \equiv 2 \mod 3$, we compute only c_n^2 and that takes two multiplications in $\mathbb{F}_p$, while to compute the cost of the three last identities we consider the relation $x \cdot z \ - \ y \cdot z^p$ that takes four multiplications in $\mathbb{F}_p$, setting $x = c_{n+1}$, $y = c_n$ and $z = c$ for *ii*, $x = c_{n-1}$, $y = c^p$ and $z = c_n$ for *iii*, and finally $x = c_{n+1}$, $y = c$ and $z = c_n$ for *iv*.

Definition 3.3. *Let* $S_n(c) = (c_{n-1},\ c_n,\ c_{n+1}) \in (\mathbb{F}_{p^2})^3$

Algorithm 3.1 (Computation of $S_n(c)$ given n and c, see [8, 9])

1. If $\underline{n < 0}$ apply this algorithm to $-n$ and c, and apply the relation $c_{-n} = c_{np} = c_n^p$ to the resulting value, obtaining

$$S_n(c) = \left(c_{-(n+1)},\ c_{-n},\ c_{-(n-1)}\right) = \left(c_{n+1}^p,\ c_n^p,\ c_{n-1}^p\right)$$

2. If $\underline{n = 0}$ apply the relation $c_{-n} = c_{np} = c_n^p$ and with $c_0 = 3$, obtaining

$$S_0(c) = (c_{-1},\ c_0,\ c) = (c^p,\ 3,\ c)$$

3. If $\underline{n = 1}$ apply the identity $c_{2n} = c_n^2 - 2c_n^p$, obtaining

$$S_1(c) = (c_0,\ c,\ c_2) = \left(3,\ c,\ c^2 - 2c^p\right)$$

4. if $\underline{n = 2}$ use the identity $c_{n+2} = c \cdot c_{n+1} - c^p \cdot c_n + c_{n-1}$ and $S_1(c)$ to compute c_3 and thereby

$$S_2(c) = (c,\ c_2,\ c_3)$$

5. if $\underline{n > 2}$ to compute $S_n(c)$ let $m = n$

   If m is even then replace m by $m - 1$.

   Let $m = 2\overline{m} + 1$, $\overline{S}_{\overline{m}}(c) = S_m(c)$, $k = 1$ and compute $\overline{S}_k(c) = S_3(c)$ using the identity $c_{n+2} = c \cdot c_{n+1} - c^p \cdot c_n + c_{n-1}$ and $S_2(c)$.

   Let $\overline{m} = \sum_{j=0}^{r} \overline{m}_j 2^j$ with $\overline{m}_j \in \{0,\ 1\}$ and $\overline{m}_r = 1$.

   For $j = r-1,\ r-2,\ ....,\ 1,\ 0$ in succession do the following:

   - If $\overline{m}_j = 0$ then use $\overline{S}_k(c) = (c_{2k},\ c_{2k+1},\ c_{2k+2})$ to compute $\overline{S}_{2k}(c) = (c_{4k},\ c_{4k+1},\ c_{4k+2})$, using:

- The identity $c_{2n} = c_n^2 - 2c_n^p$ to compute c_{4k} and c_{4k+2}.
- The identity $c_{2n-1} = c_{n-1} \cdot c_n - c^p \cdot c_n^p + c_{n+1}^p$ to compute c_{4k+1}.

and replace k by $2k$.

- If $\overline{m}_j = 1$ then use $\overline{S}_k(c) = (c_{2k},\ c_{2k+1},\ c_{2k+2})$ to compute $\overline{S}_{2k+1}(c) = (c_{4k+2},\ c_{4k+3},\ c_{4k+4})$, using:

- The identity $c_{2n} = c_n^2 - 2c_n^p$ to compute c_{4k+2} and c_{4k+4}.
- The identity $c_{2n+1} = c_{n+1} \cdot c_n - c \cdot c_n^p + c_{n-1}^p$.

and replace k by $2k+1$.

After this iterations we have that $k = \overline{m}$ and $S_m(c) = \overline{S}_{\overline{m}}(c)$.

If n is even use $S_m(c) = (c_{m-1},\ c_m,\ c_{m+1})$ to compute $S_{m+1}(c) = (c_m,\ c_{m+1},\ c_{m+2})$ using the identity $c_{n+2} = c \cdot c_{n+1} - c^p \cdot c_n + c_{n-1}$ and replace m by $m+1$.

As a result we have $S_n(c) = S_m(c)$.

Remark 3.1. The only difference between the two cases in the last algorithm is the application of the identity $c_{2n-1} = c_{n-1} \cdot c_n - c^p \cdot c_n^p + c_{n+1}^p$ if the bit is off and of the identity $c_{2n+1} = c_{n+1} \cdot c_n - c \cdot c_n^p + c_{n-1}^p$ if the bit is on. The two computations are very similar and take the same number of instructions. This great similarity makes this algorithm much less susceptible to attacks (see [9]).

Theorem 3.4 (see [8]). *Given the sum c of the roots of $F(c,\ X)$, computing the sum c_n of the n^{th} powers of the roots takes $8 \cdot \log_2(n)$ multiplications in $\mathbb{F}_p$.*

Proof . The proof of this theorem follows immediately from Algorithm above and Corollary 3.1 .

In fact $\log_2(n)$ is the number of bits necessary to represent n in binary base, and so it is the number of iterations in algorithm necessary to compute $S_n(c)$, moreover for each iteration are necessary 8 multiplications to compute the two c_k, four multiplications for each c_k computed with the identities of the Corollary 3.1 .

Now setting $c = Tr(g)$ and remembering the two Theorem 3.1 and 3.2, it follows that

$$S_n(Tr(g)) = \big(Tr(g^{n-1}),\ Tr(g^n),\ Tr(g^{n+1}))\big)$$

Therefore, the algorithm above can be used to compute $S_n(Tr(g))$ for any n, given $Tr(g)$.

Moreover, considering the Theorem 3.4, given the representation $Tr(g) \in \mathbb{F}_{p^2}$ of the conjugates of g, the representation $Tr(g^n) \in \mathbb{F}_{p^2}$ of the n^{th} power of g can be computed at the cost of $8\log_2(n)$ multiplications in $\mathbb{F}_p$, for any integer n.

If we assume, as the experts, that a squaring in $\mathbb{F}_p$ takes 80% of the time of a multiplication in $\mathbb{F}_p$ (see [2]), then, according to Lemma 2.2., computing g^n given g can be expected to take $23.4\log_2(n)$ multiplications in $\mathbb{F}_p$.

Thus, comparing these two last results we obtain that computing $Tr(g^n)$ given $Tr(g)$ is almost three times faster than computing g^n given g.

We will see ahead that the ability to quickly compute $Tr(g^n)$ based on $Tr(g)$ suffices for the implementation of many cryptographic protocols.

Chapter 4

XTR parameter and key selection

In the previous chapters we have seen that, to construct XTR, primes $p,\ q$ must be satisfy the following conditions:

1. p and $n+1$ are prime numbers and p is a primitive root $\pmod{n+1}$.
2. $\Phi_6(p) = p^2 - p + 1$ has a prime factor q.

By the first condition we obtain that $\mathbb{F}_{p^2}$ has an optimal normal basis (cf. Theorem 1.5), while by the second condition the *XTR subgroup* $\boldsymbol{Y}$, with order q, cannot be embedded in the multiplicative group of any true subfield of $\mathbb{F}_{p^6}$(see [7], Lemma 2.4).

Therefore in the first section we are going to present any algorithm to find the primes p and q that satisfy the two conditions above, while in the second section we are going to consider the problem to find, with an element $g \in \mathbb{F}_{p^6}$ with order q, a proper $Tr(g)$.

4.1 Selection of p and q

Now let us describe fast and practical methods that satisfy the two previous conditions. Moreover, to use the fast $\mathbb{F}_{p^2}$ arithmetic described in chapter 2, the prime p should be $2 \pmod 3$.

Let P and Q denote the bit lengths of the primes p and q to be generated, respectively.

Algorithm 4.1 (Selection of "nice" p and q, see [8])

1. Find $r \in \mathbb{Z}$ such that $q = r^2 - r + 1$ is a Q-bit prime [1].
2. Find $k \in \mathbb{Z}$ such that $p = r + k \cdot q = kr^2 + (1-k)r + k$ is a P-bit prime that is $2 \pmod 3$.

Trough this algorithm we can find primes p that satisfy a second degree polynomial with small coefficients.

We note that if we find $k = 1$, then we search r such that both $r^2 - r + 1$ and $r^2 + 1$ are prime and such that $p \equiv r^2 + 1 \equiv 2 \pmod 3$ and so the prime p has a "very nice" form.

In this case to obtain p prime, r must be even, i.e. $r = 2k$ with $k \in \mathbb{Z}$, therefore $p = r^2 + 1 = 4k^2 + 1$ and so $p \equiv 1 \pmod 4$.

Another method to generate p and q is the following:

Algorithm 4.2 (Selection of p and q, see [8, 9])

1. Select a Q-bit prime $q \equiv 7 \pmod{12}$ [2].
2. Find the roots r_1 andr_2 of $X^2 - X + 1 \pmod q$.
3. Find $k \in \mathbb{Z}$ such that

$$p = r_i + k \cdot q \quad \text{for} \quad i = 1,\ 2$$

[1] It is conjectured there are infinitely many primes of the form $r^2 - r + 1$.

[2] Exists for the Dirichlet's theorem, this theorem states that for any two positive coprime integers a and b, there are infinitely many primes of the form a + nb, i.e. there are infinitely many primes which are congruent to a mod b.

`is a` P-`bit prime that is` $2 \pmod 3$.

It follows from $q \equiv 7 \pmod{12}$ that $q \equiv 1 \pmod 3$ and $q \equiv 3 \pmod 4$.

Now let us compute the roots of $X^2 - X + 1$ which have the form:

$$r_{1/2} = \frac{1 \pm \sqrt{-3}}{2} \pmod q;$$

then r_1 and r_2 exsist if and only if -3 is a quadratic residue modulo q, i.e

$$\left(-\frac{3}{q}\right) = \left(-\frac{1}{q}\right) \cdot \left(\frac{3}{q}\right) = (-1)^{\frac{q-1}{2}} \left(\frac{3}{q}\right) = -\left(\frac{3}{q}\right).$$

The last identity follows since $q \equiv 3 \pmod 4$ and so $\frac{q-1}{2}$ is odd. Moreover from Quadratic Reciprocity Law (see [5]), since both 3 and q are $\equiv 3 \pmod 4$, we have that:

$$-\left(\frac{3}{q}\right) = \left(\frac{q}{3}\right)$$

that is equal to 1 since $q \equiv 1 \pmod 3$.

Thus we have proved that the roots exsist.

Moreover, since $q \equiv 3 \pmod 4$, r_1 and r_2 can be found using a single $((q+1)/4)^{th}$ powering modulo q (see [5]).

Finally if $p = r_i + kq$ for $i = 1,\ 2$ then q divide $p^2 - p + 1$.

4.2 Selection of the subgroup

Given p and $q > 3$, we must find $Tr(g) \in \mathbb{F}_{p^2}$ for an element $g \in \mathbb{F}_{p^6}$ of order q.

We observe that an actual representation of g is not required and it suffices to know its trace.

But if we know $Tr(g)$ for an unspecified g, we can compute a generator of *XTR subgroup* determining any root of $F(Tr(g), X)$ (cf. Definition 3.2).

One of the approach to find $Tr(g)$ is to represent $\mathbb{F}_{p^6}$ using a third degree irreducible polynomial over $\mathbb{F}_{p^2}$ and to choose an element $k \in \mathbb{F}_{p^6}$ such that $k^{(p^6-1)/q} \neq 1$. Set $g = k^{(p^6-1)/q}$, it is possible to compute $Tr(g)$ (see [9]).

We obtain another approach recalling that $F(c,X) \in \mathbb{F}_{p^2}[X]$ is irreducible if and only if its roots in $\mathbb{F}_{p^6}^*$ have order dividing $p^2 - p + 1$ and are > 3 (cf. Proof *iv* Lemma 3.2). It follows that if $c \in \mathbb{F}_{p^2}$ is such that $F(c,X)$ is irriducible, then c is the trace of an element in $\mathbb{F}_{p^6}^*$ of order > 3 dividing $p^2 - p + 1$.

If furthermore $c_{(p^2-p+1)/q} \neq 3$ then $c_{(p^2-p+1)/q}$ (cf. Definition 3.2) is the trace of an element of order q. In fact, we have:

let k an element in *XTR supergroup* $\boldsymbol{Y}$ such that $c = Tr(k)$, then $h = k^{(p^2-p+1)/q} \neq 1$ since $Tr(h) = c_{(p^2-p+1)/q} \neq 3$. Moreover, since $k \in \boldsymbol{Y}$, $h^q = k^{(p^2-p+1)} = 1$ and so the order of h is q (see [9]).

Now it remains to find $c \in \mathbb{F}_{p^2}$ such that $F(c,X)$ is irreducible. For this purpose let us consider the following result:

Lemma 4.1. *(see [8]) For a randomly selected $c \in \mathbb{F}_{p^2}$ the probability that $F(c,X) \in \mathbb{F}_{p^2}[X]$ is irreducible is about one third.*

Proof . Recalling that almost all the elements of the *XTR supergroup* $\boldsymbol{Y}$ are roots of a monic irreducible polynomials of the form $F(c,X)$, except the elements with order ≤ 3 (cf. Lemma 3.1, Theorem 3.1 and 3.3). There are about $p^2 - p$ elements of the subgroup of order $p^2 - p + 1$ of $\mathbb{F}_{p^6}^*$ that are roots of the irreducible polynomials $F(c,X)$.

But, since each of these polynomials has three distinct roots (cf. Definition 3.2), there must be $(p^2 - p)/3$ different values for $c \in \mathbb{F}_{p^2} \setminus \mathbb{F}_p$ such that $F(c,X)$ is irreducible. □

Now we present an algorithm that is based on the fact that $F(c,x)$ is reducible if and only if $c_{p+1} \in \mathbb{F}_p$ (cf. Theorem, 3.3) and on the method to find $Tr(g)$ above.

Algorithm 4.3 (Computation of $Tr(g)$, see [8])

1. `Pick` $c \in \mathbb{F}_{p^2} \setminus \mathbb{F}_p$ `at random.`

2. Compute c_{p+1} using Algorithm 3.1, if $c_{p+1} \in \mathbb{F}_p$ then return to Step 1.

3. Compute $c_{(p^2-p+1)/q}$ using Algorithm 3.1, if $c_{(p^2-p+1)/q} = 3$ then return to Step 1.

4. Let $Tr(g) = c_{(p^2-p+1)/q}$.

The Algorithm 4.3 computes $Tr(g) \in \mathbb{F}_{p^2}$ for some $g \in \mathbb{F}_{p^6}$ of order q. By the following theorem we compute the number of applications that this Algorithm requires.

Theorem 4.1. *The Algorithm 4.3 can be expected to require:*

- $\frac{3q}{(q-1)}$ *applications of Algorithm 3.1 with* $n = p + 1$.
- $\frac{q}{(q-1)}$ *applications of Algorithm 3.1 with* $n = (p^2 - p + 1)/q$.

Then the total number of multiplications that the Algorithm 4.3 requires is:

$$\frac{q}{q-1}\left(24 \cdot \log_2 (p - 1 \bmod q) + 8 \cdot \log_2 \left(\frac{p^2 - p + 1}{q} \bmod q\right)\right)$$

See [9] for the proof.

Chapter 5

Cryptographic applications

XTR can be used in any cryptosystem that relies on the subgroup discrete logarithm problem. Therefore, we consider some applications of XTR, in particular we are going to present the XTR version of the Diffie Hellman key agreement protocol and ElGamal encryption and decryption protocols (see [8, 9]).

In the first section of this chapter we will present the XTR versions of these protocols. They follow by replacing the ordinary representation of subgroup elements with the XTR representation of subgroup elements of a multiplicative group. In the XTR version, in fact, the XTR group elements are replaced by their traces.

In the second section we will show the Discrete Logarithm problem and other two problems that are related to it and we will consider their XTR version.

5.1 XTR - Diffie Hellman and XTR - ElGamal

The Diffie-Hellman key agreement protocol is the first published solution to the key distribution problem, that allows two parties, never met, to establish a shared secret key by exchanging information over an open channel (see [9]).

In the traditional DH scheme there are two users that agree on a multiplicative group $\mathbb{F}_p^*$ of a prime field $\mathbb{F}_p$ and on a generator g of $\mathbb{F}_p^*$. Each user sends a random power of g to the other one and raises, the value received from the other part, to the same power and so each user obtains the shared secret key.

In the protocols that we present below there are two users **A** and **B** that have already agreed upon *XTR subgroup* $\boldsymbol{Y}$ and on the XTR public key data containing two primes, p and q as in Chapter 4 (cf. Section 4.1), and the trace of a generator of the *XTR subgroup* (cf. Section 4.2).

Algorithm 5.1 (**XTR - DH key agreement**, see [9])

Let $(p, q, Tr(g))$ be XTR public key data. We suppose that this key is shared by two users, A and B, that want to agree on a secret key K. Then they must proceed as follows:

1. A selects a random integer a with $1 < a < q - 2$ and computes

   $$S_a(Tr(g)) = (Tr(g^{a-1}), Tr(g^a), Tr(g^{a+1})) \in (\mathbb{F}_{p^2})^3$$

   using Algorithm 3.1 with $n = a$ and $c = Tr(g)$.

   A sends $Tr(g^a) \in \mathbb{F}_{p^2}$ to B.

2. B selects a random integer b with $1 < b < q - 2$ and computes

   $$S_b(Tr(g)) = (Tr(g^{b-1}), Tr(g^b), Tr(g^{b+1})) \in (\mathbb{F}_{p^2})^3$$

   using Algorithm 3.1 with $n = b$ and $c = Tr(g)$.

   B sends $Tr(g^b) \in \mathbb{F}_{p^2}$ to A.

3. A receives $Tr(g^b)$ from B and computes

   $$S_a(Tr(g)^b) = (Tr(g^{(a-1)b}), Tr(g^{ab}), Tr(g^{(a+1)b})) \in (\mathbb{F}_{p^2})^3$$

   using Algorithm 3.1 with $n = a$ and $c = Tr(g^b)$.

   Then A obtains a shared secret key $K = Tr(g^{ab}) \in \mathbb{F}_{p^2}$.

4. B receives $Tr(g^a)$ from A and computes

$$S_b(Tr(g)^a) = (Tr(g^{(b-1)a}), Tr(g^{ab}), Tr(g^{(b+1)a})) \in (\mathbb{F}_{p^2})^3$$

using Algorithm 3.1 with $n = b$ and $c = Tr(g^a)$.

Then B obtains the shared secret key $K = Tr(g^{ab}) \in \mathbb{F}_{p^2}$.

We recall that in Chapter 3 we have seen that, using Algorithm 3.1, to compute $Tr(g^n)$ given $Tr(g)$ is almost three times faster than to compute g^n given g.

We note that in each step of the Algorithm 5.1 we compute $Tr(g^h)$, where g^h is an element in XTR subgroup with $0 < h < q - 2$, using the Algorithm 3.1. Then each step of the Algorithm 5.1 is almost three times faster than each step of the traditional Diffie-Hellman protocol that is based on subgroup of multiplicative group of finite field and that in each step computes the power of the elements of this last subgroup (see [6]).

Now, we consider the ElGamal cryptosystem, named after its inventor, Taher ElGamal, this cryptosystem consists of two components:

the encryption algorithm and the decryption algorithm.

In the traditional ElGamal encryption the public key, of a user **A**, is the triplet (p, g, g^x), where p is a prime, g is a generator of $\mathbb{F}_p^*$ and the integer x, such that $1 < x < p - 1$, is a private key.

If the user **B** want to send a message M to **A** then must to encrypt a plaintext M, then he selects a random integer k relatively prime to $p-1$ and computes the following values:

$$K \longleftarrow g^k \pmod p$$
$$E \longleftarrow M(g^x)^k \pmod p$$

Then the ciphertext consists of the pair (K, E) computed above.

To retrieve the plaintext M, **A** uses the ElGamal decryption scheme, computing:

$$M \longleftarrow E \cdot (K^x)^{-1} \pmod{p}$$

It is easy to verify that:

$$E \cdot (K^x)^{-1} = Mg^{xk}(g^{xk})^{-1} = M \pmod{p}$$

(see [1])

Now we describe the XTR version of ElGamal cryptosystem and we will show the differences respect to the traditional version.

In the protocol below we have that XTR public key data may also contain a public point $Tr(g^k)$ for an integer k that is kept secret, the so-called private key (see [9]).

Algorithm 5.2 (see [8])

Let $(p, q, Tr(g), Tr(g^k))$ be XTR public key data of A, where $Tr(g^k)$ is a value computed and made public by A, for some integer k selected by A.

Then the private key of A is k.

(XTR - ElGamal encryption).

We suppose that B wants to encrypt a message M intended for A, then B proceeds as follows:

1. B selects a random b such that $1 < b < q - 2$ and computes:

$$S_b(Tr(g)) = (Tr(g^{b-1}), Tr(g^b), Tr(g^{b+1})) \in (\mathbb{F}_{p^2})^3$$

   using Algorithm 3.1 with $n = b$ and $c = Tr(g)$, to obtain $Tr(g^b)$.

2. B computes

$$S_b(Tr(g^k)) = (Tr(g^{(b-1)k}), Tr(g^{bk}), Tr(g^{(b+1)k})) \in (\mathbb{F}_{p^2})^3$$

   using Algorithm 3.1 with $n = b$ and $c = Tr(g^k)$, to obtain $Tr(g^{bk})$.

3. B determines a symmetric encryption key K based on $Tr(g^{bk}) \in \mathbb{F}_{p^2}$ and uses an agreed upon symmetric encryption method with key K, to encrypt M. Then B obtains the encrypt message E.

4. B sends $(Tr(g^b), E)$ to A

(XTR - ElGamal decryption).

A decrypts the message $(Tr(g^b), E)$ that he has received by B in the following way:

1. A computes

$$S_k(Tr(g^b)) = (Tr(g^{b(k-1)}), Tr(g^{bk}), Tr(g^{(b+1)k})) \in (\mathbb{F}_{p^2})^3$$

using Algorithm 3.1 with $n = k$ and $c = Tr(g^b)$, to obtain $Tr(g^{bk})$.

2. A determines the symmetric encryption key K based on $Tr(g^{bk}) \in \mathbb{F}_{p^2}$

3. A decrypts E using the agreed upon symmetric encryption method with key K and so A obtain the message M that B has sent him.

Now if we compare XTR - ElGamal with the traditional ElGamal encryption we note that, while in the first the key K is used in conjunction with an agreed upon symmetric key encryption method, in the traditional ElGamal encryption the message is encrypt using multiplication by the key. This property of the traditional ElGaml encryption would amount to require in XTR - ElGamal encryption that $E = K \cdot M$.

For ElGamal - decryption it is implicitly assumed that the first component of an ElGamal encrypted message represents $Tr(g^b)$.

As we have noted for XTR - DH Key agreement protocol, in XTR - ElGamal encryption and decryption we use the Algorithm 3.1 to compute

the trace $Tr(g^k)$ given $Tr(g)$. So Algorithm 5.2 is three times faster than the traditional ElGamal encryption and decryption that computes g^k given an element g of subgroup of multiplicative of finite field.

5.2 Discrete logarithm

We recall that the traditional version of the two protocols, that we have seen before, bases their security on the difficulty to solve the discrete logarithm problem.

Now we are going to show how there is, also, the XTR logarithm problem version on which the security of XTR - Diffie Hellman and XTR - ElGamal protocols is based.

Let G be a multiplicative group of order n. Given $g,\ y \in G$, the discrete logarithm of y to the base g is an integer x such that:

$$g^x \equiv y \in G$$

and it is written as

$$x = \log_g y$$

If g generates G then such an x with $0 \leq x < n$ exsists.

Now we consider a moltiplicative group $G = \langle g \rangle$ of prime order q.

We know that it is easy to raise numbers to large power using repeted squaring, while to compute x, given g and y, can be quite hard.

We define the *Discrete Logarithm (DL)* problem in $\langle g \rangle$, the computation, for a given $y \in \langle g \rangle$, of the unique $x \in \{0, 1,\ \ldots\ , q-1\}$ such that $g^x = y$ and we write $x = DL(y)$ (see [8, 16]).

Related to the *Discrete Logarithm* (DL) problem are *Diffie Hellman* (DH) problem and *Diffie Hellman Decision* (DHD) problem.

The *Diffie Hellman* (DH) problem is the problem, given g^x and g^y, to compute g^{xy}. We write $DH(g^x, g^y) = g^{xy}$.

Instead, the *Diffie Hellman Decision* (DHD) problem is the problem, given $a,\ b,\ c \in G$, to determine whether $c = DH(a, b)$.

It is widely believed and assumed that if DL problem in G is intractable, then the other two are so.

On the other hand, it is easy to observe that if one is able to solve the *Discrete Logarithm* (DL) problem, then himself can solve *Diffie Hellman* (DH) problem and *Diffie Hellman Decision* (DHD) problem (see [8]).

As we have seen before, the XTR version of the protocols follows by replacing elements of the XTR group by their traces and so the sicurity of these protocols is no longer based on the original DH, DHD or DL problems but on their XTR versions (see [9]).

For this reason we will deal with the XTR-DL, XTR-DH and XTR-DHD problems.

The XTR-DL problem, given $Tr(y)$ with $y \in \boldsymbol{Y} = \langle g \rangle$, is to find $x \in \{0, 1, , \ ... \ , q-1\}$ such that $Tr(g^x) = Tr(y)$. We write that $x = XDL(Tr(y))$.

We observe that, since $Tr(g) = Tr(g^{p^2}) = Tr(g^{p^4})$ (see Chapter 3) for a given $y \in \langle g \rangle$, if $x = XDL(Tr(y))$, then so are $x \cdot p^2 \pmod q$ and $x \cdot p^4 \pmod q$.

The XTR-DH problem is to compute $Tr(g^{xy})$, given $Tr(g^x)$ and $Tr(g^y)$ and we can write $XDH(Tr(g^x), Tr(g^y)) = Tr(g^{xy})$.

Finally the last problem is to determine whether $XDH(a, b) = c$ for $a, \ b, \ c \in Tr(\langle g \rangle)$

Before we prove the equivalence between old and new problems, we consider the following result:

Definition 5.1. *(see [9]) We say that problem $\mathcal{A}$ is (a, b)-equivalent to problem $\mathcal{B}$, if any istance of problem $\mathcal{A}$ (or $\mathcal{B}$) can be solved by at most a (or b) calls to an algorithm solving problem $\mathcal{B}$ (or $\mathcal{A}$).*

Theorem 5.1. *(see [9]) The following equivalences hold:*

i. The XTR-DL problem is $(1, 1)$-equivalent to the DL problem in G.

ii. The XTR-DH problem is $(1, 2)$-equivalent to the DH problem in G.

iii. The XTR-DHD problem is $(3, 2)$-equivalent to the DHD problem in G.

Proof . We denote with $r(Tr(y))$ the root of $F(Tr(y), X)$, given $y \in \boldsymbol{Y} = \langle g \rangle$ (see Definition 3.2). In order to prove i, we must to compute $DL(y)$.

Let $x = XDL(Tr(y))$, then $DL(y) = x \cdot p^{2j} \pmod{q}$ for either $j = 0,\ j = 1$ or $j = 2$.

Conversely for $Tr(y) \in \mathbb{F}_{p^2}$ then $XDL(Tr(y)) = DL(r(Tr(y)))$, that proves i.

Now we set $a = g^x$ and $b = g^y$ and we want to compute $DH(a, b) = DH(g^x, g^y)$.

So we compute $d_i = XDH(Tr(a \cdot g^i), Tr(b)) = XDH(Tr(g^x \cdot g^i), Tr(g^y)) = Tr(g^{(x+i)y})$ for $i = 0, 1$, then

$$r(d_i) \in \{(DH(g^x, g^y) \cdot g^{iy})^{p^{2j}} : j = 0,\ 1,\ 2\}$$

from which $DH(g^x, g^y)$ follows.

Conversely $XDH(Tr(g^x), Tr(g^y)) = Tr(DH(r(g^x), r(g^y)))$, that completes the proof of ii.

Finally, the fact that the XTR-DHD problem is $(3, 2)$-equivalent to the DHD problem in G, follows from these equivalences:

$$DH(g^x, g^y) = g^{xy} \text{ if and only if } \begin{array}{l} XDH(Tr(g^x), Tr(g^y)) = Tr(g^{xy}) \text{ and} \\ XDH(Tr(g^x \cdot g), Tr(g^y)) = Tr(g^{xy} \cdot g^y). \end{array}$$

Conversely, let $a,\ b,\ c \in Tr(\langle g \rangle)$

$$XDH(a, b) = c \text{ if and only if } \begin{array}{l} DH(r(a), r(b)) = r(c)^{p^{2j}} \\ \text{for either } j = 0, j = 1 \text{ or } j = 2. \end{array}$$

These two equivalences prove iii. □

We recall that, thanks to Pohling-Hellman algorithm (see [15]), given a group G of order an integer n, the Discrete Logarithm problem in G can be reduced to the Discrete logarithm problem in all prime order subgroups of G.

Algorithm 5.3 (Pohling-Hellman's algorithm)

Let $q = p^n, q-1 = \prod_{1 \le i \le s} {p_i}^{e_i}$, let g be a generator of $\mathbb{F}_q^*$ and let $y \in \mathbb{F}_q^*$. We will compute $x = \log_g y$.

1. To compute x_i for $i = 1, 2, \ \ldots \ , s$ as follows:

   1.1 Set $\alpha = 1, l_{-1} = 0$ and $\overline{g} = g^{\frac{q-1}{p_i}} \pmod q$.

   1.2 For $j = 0, \ \ldots \ , e_i - 1$, compute:

$$\begin{aligned} \alpha &\longleftarrow \alpha \cdot g^{l_{j-1} {p_i}^{j-1}} \pmod q \\ \alpha &\longleftarrow (\alpha^{-1} \cdot y)^{\frac{q-1}{p_i}^{j+1}} \pmod q \\ l_j &\longleftarrow \log_{\overline{g}} \overline{y} \pmod q \end{aligned}$$

   1.3 Then $x_i = l_0 + l_1 p_i + \ \ldots \ + l_{e_i - 1} {p_i}^{e_i - 1}$.

2. Find a solution of the system of congruance equation:

$$\begin{cases} x \equiv x_1 \pmod {{p_1}^{e_1}} \\ \vdots \\ x \equiv x_s \pmod {{p_s}^{e_s}} \end{cases}$$

3. Return $x = \log_g y$.

It follows that, to obtain a secure cryptosystem, the group order should contain at least one large prime factor.

Now we are going to show that, if q is sufficiently large, DL problem in XTR subgroup $\boldsymbol{Y}$ is as hard as in $\mathbb{F}_{p^6}^*$.

This last multiplicative group has order $p^6 - 1$ and its factors are $\Phi_t(p)$ where t is a divisor of 6 and $\Phi_t(p)$ is the t-th cyclotomic polynomial evaluated in p (see Chapter 1).

So

$$p^6 - 1 = (p-1)(p+1)(p^2+p+1)(p^2-p+1).$$

We observe that the subgroups of order $\Phi_t(p)$ for $t = 1, 2, 3$, i.e. $(p-1)$, $(p+1)$ and (p^2+p+1), can be efficently embedded in multiplicative group of the proper subfields $\mathbb{F}_{p^t}$ of $\mathbb{F}_{p^6}$, for $t = 1, 2, 3$.

In fact we know that:

- The subgroup of order $p-1$ can be embedded in $\mathbb{F}_p$.
- Since $p+1$ divides p^2-1, the subgroup of order $p+1$ can be embedded in $\mathbb{F}_{p^2}$.
- Since p^2+p+1 divides p^3-1, the subgroup of order p^2+p+1 can be embedded in $\mathbb{F}_{p^3}$.

Instead the subgroup of order $\Phi_6(p) = p^2-p+1$ cannot be embedded in any proper subfield of $\mathbb{F}_{p^6}$.

Then if we want to solve the discrete logarithm problem in $\mathbb{F}_{p^6}^*$, we must be solve four discrete logarithm problems.

Three of these discrete logarithm problems can be reformulated as the discrete logarithm problems in multiplicative groups of proper subfields $\mathbb{F}_p$, $\mathbb{F}_{p^2}$ and $\mathbb{F}_{p^3}$ of $\mathbb{F}_{p^6}$ and these problems are believed to be considerably easier than the problem in $\mathbb{F}_{p^6}^*$ (see [17]).

For this reason we can conclude that the hardness of computing discrete logarithm in the group $\mathbb{F}_{p^6}^*$ must reside in its cyclotomic subgroup of order $\Phi_6(p) = p^2-p+1$.

Now we consider the XTR subgroup $\boldsymbol{Y}$ that has order a large prime q which divides p^2-p+1.

The subgroup $\boldsymbol{Y}$ cannot be embedded into any proper subfield of $\mathbb{F}_{p^6}$ and this fact is due the following lemma (see [16]):

Lemma 5.1. *(see [7])*

Let $q > s$ be a prime factor of $\Phi_s(p)$. Then q does not divide any $\Phi_t(p)$ for divisors t of s with $t < s$.

Proof . Since q divides $\Phi_s(p)$ then $p^s - 1 \equiv 0 \pmod{q}$.

Moreover since q is prime and $q > s$, it follows that s divides $q - 1$.

Therefore $\Phi_s(X)$ and the $\Phi_t(X)$'s are among the distinct irreducible factors of $X^{q-1} - 1$.

But this last polynomial factors into $q-1$ distinct linear factors in $\mathbb{F}_q[X]$ and one of these is $(X - p \bmod q)$, since q divides $\Phi_s(p)$ and so p is a root of $\Phi_s(X)$ in $\mathbb{F}_q[X]$.

We obtain that only one factor of $X^{q-1} - 1$ is equal $(X - p \mod q)$ and so p is not a root of any of the $\Phi_t(X)$'s, then lemma follows. □

From this last result we can conclude that, with a proper choice of q dividing $p^2 - p + 1$, the discrete logarithm problem in $\boldsymbol{Y}$ is as hard as it is in $\mathbb{F}_{p^6}$.

Finally, recalling that XTR - DL problem is equivalent to the DL problem (cf. Theorem 5.1) and considering the XTR - DH key agreement and XTR - ElGamal encryption and decryption , we note that the security of these protocols is based on the difficulty to solve the discrete logarithm problem in $\boldsymbol{Y}$.

Then as we have just shown, the security of the XTR - DH key agreement and XTR - ElGamal encryption and decryption protocols is based on the hardness to solve the discrete logarithm in $\mathbb{F}^*_{p^6}$.

So we can conclude that the necessity to use XTR subgroup is due, not only for the speed to execute of the calculations, in particular that of n^{th} powers of an element, but also for the greater security of the protocols which implement these calculations.

Appendix A

Karatsuba Algorithm

We will describe the Karatsuba's trick for the multiplication of two polynomials of degree less than n and we will show how, through this trick, the computational costs, to compute this multiplication, improve.

For this purpose, we first consider the classic multiplication algorithm.

Let

$$A = \sum_{0 \leq i < n} A_i x^i \text{ and } B = \sum_{0 \leq i < m} B_i x^i \in \mathbb{F}_p[x]$$

then

$$\begin{aligned} A \cdot B &= \left(\sum_{0 \leq i < n} A_i x^i \right) \cdot \left(\sum_{0 \leq i < m} B_i x^i \right) \\ &= \sum_{0 \leq i \leq (n-1)+(m-1)} \left(\sum_{j \in I_i} A_i \cdot B_{i-j} \right) x^i \end{aligned}$$

where $I_i = \{0 \leq j \leq m : 0 \leq i - j < n\}$.

So we can compute the product between A and B with at most $m \cdot n$ multiplication and $(m-1) \cdot (n-1)$ additions in $\mathbb{F}_p$.

Definition A.1. *(Multiplication time, see [14])*

The function

$$M : \mathbb{N}_{>0} \to \mathbb{R}_{>0}$$

is called a ***multiplication time*** *for* $\mathbb{F}_p[x]$*, if two polynomials in* $\mathbb{F}_p[x]$ *of degree less than* n *can be multiplied using at most* $M(n)$ *operations in* $\mathbb{F}_p$*.*

Then if we consider two polynomials of degree $n-1$ and $m-1$ that, as we have seen before, can be multiplied with $n \cdot m$ multiplications and $(m-1)(n-1)$ additions in $\mathbb{F}_p$ and we suppose that $n > m$, it follows that the number of operations in $\mathbb{F}_p$ is

$$n^2 + (n-1)^2 = 2n^2 - 2n + 1$$

and so

$$M(n) = O(n^2)$$

Thus, we are going to show, how, with the Karatsuba algorithm, the number of operations to compute the multiplications between this two polynomials of degree less than n decrease.

A.1 Karatsuba's algorithm

Algorithm A.1 (Karatsuba's multiplication algorithm, [14])

Let A and B be polynomials over $\mathbb{F}_p[x]$ of degree less than $n \in \mathbb{N}_{\geq 1}$.

1. If $n = 1$ then compute $C = A \cdot B$ in $\mathbb{F}_p[x]$.

2. Else if $n \neq 1$:

   2.1 Set $m = \lceil \frac{n}{2} \rceil$.

   2.2 Write

   $$A = a_1 x^m + a_0 \text{ and } B = b_1 x^m + b_0$$

   with $a_0,\ a_1,\ b_0,\ b_1 \in \mathbb{F}_p[x]$ of degree less than m.

   Then

   $$\begin{aligned} A \cdot B &= (a_1 x^m + a_0) \cdot (b_1 x^m + b_0) \\ &= (a_1 \cdot b_1) x^{2m} + (a_1 \cdot b_0 + a_0 \cdot b_1) x^m + a_0 \cdot b_0 \end{aligned} \tag{A.1}$$

2.3 Call the algorithm recursively to compute:

$$c_0 \leftarrow a_0 \cdot b_0$$
$$c_1 \leftarrow a_1 \cdot b_1$$
$$c_2 \leftarrow (a_0 + a_1) \cdot (b_0 + b_1)$$

(Note: $(a_1 \cdot b_0 + a_0 \cdot b_1) = c_2 - c_1 - c_0$)

2.4 Compute

$$c_2 \leftarrow c_2 - c_1 - c_0$$

2.5 Compute

$$C \leftarrow c_1 x^{2m} + c_2 x^m + c_0$$

3. Return C.

Correctness of this last Algorithm follows directly by these two relations:

$$C = A \cdot B = a_1 \cdot b_1 x^{2m} + ((a_0 + a_1) \cdot (b_0 + b_1) - a_0 b_0 - a_1 b_1) x^m + a_0 \cdot b_0$$

and

$$(a_1 \cdot b_0 + a_0 \cdot b_1) = (a_0 + a_1)(b_0 + b_1) - a_1 \cdot b_1 - a_0 \cdot b_0$$

Through this algorithm, Karatsuba reduce the number of polynomial multiplication in (A.1) from 4 to 3.

The Algorithm A.1 is recursive since it can be applied again for the multiplications of the elements $a_0,\ a_1,\ b_0,\ b_1 \in \mathbb{F}_p[x]$ in step 2.

Lemma A.1. *(see [14])*

Given two polynomials in $\mathbb{F}_q[x]$ of degree less than 2^k, where $k \in \mathbb{N}_{\geq 1}$.

The Algorithm A.1 computes theirs product with

$$M(2^k) \leq 9 \cdot (3^k) - 8 \cdot 2^k \tag{A.2}$$

operations in $\mathbb{F}_q$.

Proof .

We prove the bound

$$M(2^k) \leq 9 \cdot (2^k)^{\log_2 3} - 8 \cdot 2^k = 3^{k+2} - 2^{k+3}$$

by induction on $k \in \mathbb{N}_{\geq 0}$.

For $k = 0$ we have $M(2^0) = 3^2 - 2^3 = 1$ multiplication in step 1.

Now we suppose that the relation (A.2) is true for $0 \leq k' < k$.

Since $m = \left\lceil \frac{2^k}{2} \right\rceil = 2^{k-1}$, to compute $(a_0 + a_1)$ and $(b_0 + b_1)$ we have $2 \cdot 2^{k-1}$ additions in $\mathbb{F}_q$ and three recursive calls of input size 2^{k-1} in step (2.3), then we must perform $3M(2^{k-1})$ operations in $\mathbb{F}_q$.

In step (2.4), for the computation of c_2, we count at most $2 \cdot 2^k$ additions of coefficients.

Finally the computation of $c_1 x^{2m} + c_2 x^m + c_0$ causes 2^k additions in $\mathbb{F}_q$, we note that since $\deg(c_0) < 2 \cdot 2^{k-1}$, c_0 not causes operations in $\mathbb{F}_q$.

Summing up, we obtain:

$$M(2^k) \leq 3M(2^{k-1}) + 4 \cdot 2^k$$

and by induction hypothesis it follows:

$$\begin{aligned} M(2^k) &\leq 3(3^{k+1} - 2^{k+2}) + 2^{k+2} = 3^{k+2} - 3 \cdot 2^{k+2} + 2^{k+2} \\ &= 3^{k+2} - 2^{k+2} \end{aligned}$$

as claimed. □

From the Lemma A.1 it follows:

Corollary A.1. *Two polynomials in $\mathbb{F}_q[x]$ of degree less than $n \in \mathbb{N}_{\geq 1}$ can be multiplied with at most:*

$$M(n) \leq 9 \cdot (n)^{\log_2 3} - 8 \cdot 2^k = O(n^{1.59})$$

operations in $\mathbb{F}_q$.

So we can conclude that if we use the Karatsuba's algorithm, for the multiplication of two polynomials, the number of operations in $\mathbb{F}_q$ decrease.

Appendix B

Multi-exponentiation algorithm

It is well known that modular exponentiations or module multi-exponentiations can be used to implement fast moderm cryptosystems (see [19]). The major task to obtain an efficient performance of these cryptosystems, is to reduce the times needed to perform the operations of modular exponentiation and module multi-exponentiation (see [19]).

Given the base X, the exponent a and the module N, the modular exponentiation problem is to compute $X^a \bmod N$.

One of the common method to compute $X^a \bmod N$ is binary method that only use square and multiply operations.

With this method the exponent X must be represented in a binary form and each of these bits are scanned from left to right or right to left.

If $a = (a_{n-1},\ a_{n-2},\ \ldots\ , a_1,\ a_0)_2$, then we have the following algorithm.

Algorithm B.1 (see [3])

1. `Set` $A = 1$ `and` $i = 0$.
2. `If` $a_i = 1$ `then compute` $A = A \cdot X \bmod N$.
3. `Compute` $X = X \cdot X \pmod N$.

4. `Set` $i = i + 1$, `if` $i \leq n - 1$ `return to step 2 otherwise output` A.

Since the average number of ones in a binary representation of a is equal to $\frac{\log_2 a}{2}$, to compute $X^a \bmod N$ through Algorithm B.1 are necessary $\log_2 a$ squarings and $\frac{\log_2 a}{2}$ multiplications and so

$$\frac{3}{2}\log_2 a$$

multiplications (see [3]).

While, given the module N, the bases X_i and the exponents a_i for $i = 1, 2, \ldots, k$, the multi-exponentiation problem is to compute $Z = \prod_{0<i\leq k} X_i^{a_i} \bmod N$ (see [19]).

In this appendix we concentrate our attention on the case $k = 2$, that is, given integers a and b, compute $Z = X^a Y^b \bmod N$.

The most common solution of this last problem is to compute exponentiations individually and then to multiply these computations together.

Then, if we use Algorithm B.1 to compute the two modular exponentiations $X^a \bmod N$ and $Y^b \bmod N$, the number of multiplications required is

$$\frac{3}{2}\log_2 a + \frac{3}{2}\log_2 b < 3\log_2(\max(a, b))$$

.

To decrease this number of multiplications for multi-exponentiation, we present the so called Pekmestzi's algorithm that computes $X^a Y^b \bmod N$ faster than the method above.

Algorithm B.2 (Pekmestzi's algorithm , see [19])

`Let` X, Y, a, b, N `positive integers, to compute` $Z = X^a Y^b \bmod N$:

1. Compute $a = (a_{k-1},\ a_{k-2},\ \ldots\ ,\ a_0)$ and $b = (b_{k-1},\ b_{k-2},\ \ldots\ ,\ b_0)$, the binary representation of a and b, with $k = \max(log_2 a,\ log_2 b)$;
2. Set $Z = 1$ and $Z' = X \cdot Y \mod N$;
3. Set $h = k - 1$
4. Consider the following cases:
   - if $(a_h,\ b_h) = (1,\ 0)$ then $Z = Z \cdot X$;
   - if $(a_h,\ b_h) = (0,\ 1)$ then $Z = Z \cdot Y$;
   - if $(a_h,\ b_h) = (1,\ 1)$ then $Z = Z \cdot Z'$;
   - if $h \geq 1$ then $Z = Z \cdot Z$;
5. Set $h = h - 1$
6. If $h \geq 0$ go to step 4 otherwise output Z.

Then, with the last algorithm, to compute $X^a Y^b \bmod N$ are necessary $\log_2(\max(a,b))$ squarings and $\frac{3}{4}\log_2(\max(a,b))$ multiplications and so

$$\frac{7}{4}\log_2(\max(a,b))$$

multiplications.

My thanks

Il mio primo ringraziamento va al professore Pappalardi per la sua infinita disponibilità, pazienza e per avermi incoraggiata nei momenti di sfiducia totale.

Ringrazio in modo particolare i miei genitori, per non avermi mai fatto mancare nulla, per avermi permesso di studiare fuori sede facendo tanti sacrifici, per aver avuto tanta fiducia in me e nelle mie capacità.

Ringrazio mio fratello Carlo e mia sorella Lucia, perché nonostante i nostri litigi, so che posso sempre contare su di loro.

Un ringraziamento speciale va al mio amore Stefano, per essermi stato sempre vicino e avermi aiutata a reagire nei momenti difficili, per aver sempre creduto in me e soprattutto per aver sempre sopportato i miei sbalzi d'umore.

Grazie alla mia amica Valeria, per essere stata sempre presente in questi cinque anni, per avermi sempre capita e ascoltata. Non potrei mai fare a meno della tua amicizia.

Grazie a tutti i miei amici romani e non, perché ho sempre potuto contare su di loro, per avermi consigliata ed aiutata sempre. Grazie per avermi fatto sentire a casa, grazie perché con voi in questa immensa città non mi sento mai sola.

Infine ringrazio mia nonna Lucia, per avermi cresciuta con immenso amore, per avermi insegnato i veri valori, per avermi insegnato ad essere forte e a non mollare. Non preoccuparti nonna li terrò sempre a mente e non ti deluderò mai.

Bibliography

[1] M.W.Baldoni, C.Ciliberto, G.M. Piacentini Cattaneo, *Aritmetica, crittografia e codici*, Springer-Verlag Italia, Milano 2006.

[2] H. Cohen, A. Miyaji, T. Ono,*Efficient elliptic curve exponentiation using mixed coordinates*, Prooceedings Asiacrypt'98, LNCS 1514, Springer-Verlag 1998, 51-65.

[3] V.S.Dimitrov, G.A.Jullien and W.C.Miller, *Algorithms for multiexponentiation based on complex arithmetic*, VLSI Research Group, University of Windsor, Ontario Canada N9B 3P4.

[4] Dong-Guk Han, Ki Soon Yoon, Young-Ho Park, Chang Han Kim and Jongin Lim,*Optimal Extension Fields for XTR*, K.Nyberg and H.Heys (Eds.): SAC 2002, LNCS 2595, pp.369 − 384, Springer-Verlag Berlin Heidelberg 2003.

[5] Neal Koblitz,*A course in number theory and cryptography*, volume 114 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition, 1994.

[6] Alessandro Languasco - Alessandro Zaccagnini, *Introduzione alla crittografia moderna*, Padova e Parma 2004.

[7] A.K. Lenstra, *Using Cyclotomic Polynomials to Construct Efficient Discrete Logrithm Cryptosystems over Finite Fields*, Proceedings ACISP97, LNCS 1270, Springer-Verlag 1997, 127-138.

[8] A.K. Lenstra, E.R. Verheul, *An overview of the XTR public key system.* In K. Alster, J. Urbanowicz, and H.C. Williams, editors, *The proceedings of the Public-key Cryptography and Computational Number Theory Conference*, pages 151-180. Verlages Walter de Gruyter, 2001.

[9] A.K. Lenstra, E.R. Verheul, *The XTR public key system*, In T. Okamoto, editor, *Advances in Cryptography-Asiacrypt'00*, volume 1976 of *Lecture Notes in Computer Science*, pages 220-223, Springer-Verlag 2000.

[10] R. Lidl, H. Niederreiter, *Finite Fields*, Number 20 Encyclopedia of mathematics and its applications. Addison-Wesley, Reading MA, (1983).

[11] A.J.Menez, *Applications of Finite Fields*, Waterloo, 1993.

[12] R.C. Mullin, I.M. Onyszchuk, S.A. Vanstone, R.M. Wilson, *Optimal normal bases in* $\mathbb{F}_{p^n}$, Discrete Appl. Math., 22, 149-161 (1988/89).

[13] Munoz-Connolly, Leonela *Characterization of optimal normal bases*, Florida Atlantic University, 2000.

[14] Michael Nöcker, *Data structures for parallel exponentiation in finite fields*, Universitöt Paderborn, Fachbereich 17 Mathematik-Informatik 33095 Paderborn.

[15] S.C.Pohlig, M.E. Hellman, *An improved algorithm for computing logarithms over* $\mathbb{F}_p$ *and its cryptographic significance*, IEEE Trans. on IT, 24 (1978), 106-110.

[16] Martijn Stam, *Speeding up Subgroup Cryptosystems*, Technische Universiteit Eindhoven, 2003.

[17] Martijn Stam, A.K. Lenstra, *Speeding Up XTR*, Proceedings of Asiacrypt 2001, LNCS 2248, Springer-Verlag, 2001, 125-143; available from www.ecstr.com .

[18] S.M. Yen, C.S. Laih, A.K. Lenstra, *Multi-exponentiation*,Vol 141, 6 November 1994

[19] Yeu-Pong Lai, Chin-Chen Chang, *An efficient multi-exponentiation scheme based on modified Booth's method*, International Journal of Electronics, vol 90, 221 – 233.