Generalizations of the Diffie-Hellman Protocol: Exposition and implementation

by

J.S. van der Berg

Submitted in partial fulfillment of the requirements for the degree

Magister Scientiae

in the Department of Mathematics and Applied mathematics
in the Faculty of Natural and Agricultural Sciences

University of Pretoria
Pretoria

March 2007

Contents

1 Introduction **1**
1.1 Symmetric Cryptography 2
1.2 Public Key Distribution Systems 3
1.3 Computation 5

2 Diffie-Hellman Protocol **8**
2.1 The Protocol 8
2.2 Security of the Diffie-Hellman Protocol 10
2.3 Attacks on the Discrete Logarithm Problem 13
2.4 Security Improvements 31

3 The Compact Subgroup Protocol **34**
3.1 Representations and the Protocol 35
3.2 Implementation Advantages 48
3.3 Security 53

4 Examples of the Compact Subgroup Protocol **56**
4.1 Doing More with Fewer Bits 56
4.2 Cubic Field Extension 60
4.3 Extended XTR 69

5 Protocol Implementations **103**
5.1 Time Measurement and Data Collection 104
5.2 Analysis of Finite Field Operations 105
5.3 Modeling the CFE protocol 111
5.4 Modeling the EXTR protocol 119

List of Tables

2.1 Diffie-Hellman Protocol. 9
2.2 Sequence generated in Example 2.7. 19
2.3 Intermediate results in Example 2.10. 27

3.1 Multi Group representation. 36
3.2 Polynomial representation 38
3.3 Compact Subgroup Protocol. 46

4.1 Computing the public key of Alice in CFE 69
4.2 Computing the public key of Alice in EXTR 98

5.1 SAS output of the field operation analysis, where $x = log_2 p$ . . 106

List of Figures

1.1 Two parties using a symmetric algorithm. 3
1.2 Two parties using a public key distribution system. 5

2.1 Evident relations between problems Diffie-Hellman Problems. . 11

3.1 Key exchange using the Compact Subgroup Protocol 45
3.2 Proportion of bits saved in the Compact Subgourp Protocol. . 52

4.1 Two parties using a public key distribution system. 72

5.1 Field Addition in the Field $\mathbb{F}_p$. 107
5.2 Field Addition error in the Field $\mathbb{F}_p$. 107
5.3 Field Addition in the Extension Field $\mathbb{F}_{p^3}$. 108
5.4 Field Addition error in the Extension Field $\mathbb{F}_{p^3}$. 109
5.5 Field Multiplication in the Field $\mathbb{F}_p$. 109
5.6 Field Multiplication in the Field $\mathbb{F}_{p^3}$. 110
5.7 Field Multiplication error in the Field $\mathbb{F}_p$. 110
5.8 Field Multiplication error in the Field $\mathbb{F}_{p^3}$. 111
5.9 Model for the combining function of CFE. 116
5.10 The error of CFE combining function's model. 117
5.11 The average Hamming weight of the index of the sequence element in CFE. 117
5.12 Model for system parameter generation in CFE. 118
5.13 The error for CFE system parameter generation model. 118
5.14 Model for generation of the system parameters in XTR. 122
5.15 The error of XTR system parameter generation's model. . . . 123
5.16 Model of XTR combining function. 123
5.17 The error of XTR combining function's model. 124
5.18 The XTR Combining function's model with second data set. . 124
5.19 The Hamming weight of the index of the sequence element in data set 1 and 2. 125

Abstract.

A generalisation of the Diffie-Hellman protocol is studied in this dissertation. In the generalisation polynomials are used to reduce the representation size of a public key and linear shift registers for more efficient computations. These changes are important for the implementation of the protocol in constrained environments. The security of the Diffie-Hellman protocol and its generalisation is based on the same computations problems. Lastly three examples of the generalsation and their implementation are discussed. For two of the protocols, models are given to predict the execution time and it is determined how well these model predictions are.

DECLARATION

I, the undersigned, hereby declare that the dissertation submitted herewith for the degree Magister Scientiae to the University of Pretoria contains my own, independent work and has not been submitted for any degree at any other university.

Signature:

Name: Johan Sarel van der Berg
Date: 2006-03-18

Chapter 1

Introduction

Communication is essential in any situation where a group of people work together. Today communication takes place through various methods, like e-mail and real time chatting, using programs such as MSN® and VOIP[1]. A requirement for communication is privacy, i.e. secure communication. The need for secure communication is not a new one; as early as the Roman Empire secure communication was used. It is only recently that secure communication was needed and accessible by the general public. The study of encrypting and decrypting a message in a secure form is called *cryptography*.

The best contemporary example of secure communication is Internet banking. When one does Internet banking, it must not be possible for anyone else to understand or change the communication. Secure communication based on encryption is ensured by two types of algorithms: *symmetric* and *asymmetric*. The symmetric algorithm uses a shared secret key for encryption and decryption. This shared key is obtained from a *public key distribution system*[2] which uses an asymmetric algorithm to derive the shared key. In 1976, Diffie and Hellman introduced the first practical public key distribution system, i.e. the Diffie-Hellman Protocol.

In Chapter 2 the Diffie-Hellman Protocol is given, and security improvement of the protocol. In this dissertation three variants of the Diffie-Hellman

[1] A protocol used to transport speech over the Internet.

[2] Today the term is *key agreement system*. The original term is used since the difference between the two terms is not considered here.

protocol are discussed. The variants have advantages over the Diffie-Hellman protocol, namely an increase in speed and a reduction in the data needed to represent the keys. From these variations common ideas are identified and a general theory is created in Chapter 3. The three variations are described in detail in Chapter 4.

The practical part of this dissertation determines if knowledge of the number of field operations performed is sufficient to predict the execution time of a protocol. In Chapter 5 models are constructed and it is determined how accurate the predictions are.

The remainder of this chapter will provide some background knowledge to the rest of the dissertation. In Section 1.1 and 1.2 symmetric cryptography and public key distribution systems are explained. In Section 1.3 a short discussion of complexity is given, which will be used in the comparison of the protocols.

1.1 Symmetric Cryptography

When two people, for instance, Alice and Bob want to communicate securely over an insecure channel, they can use an encryption algorithm. With a symmetrical algorithm, the users need a shared secret that has to be agreed upon before communication can start. The shared secret is used as a key for a symmetric algorithm to enable encryption and decryption for the secure communication between Alice and Bob to take place, see Figure 1.1. As soon as Alice wants to communicate with more people securely, a large number of secret keys are needed. For n number of people to communicate securely with each other, the total number of shared keys is $n(n-1)/2$. The same number of communication channels are needed. As the number of participants in the protocol increases, the number of keys, and communication channels becomes impractical.

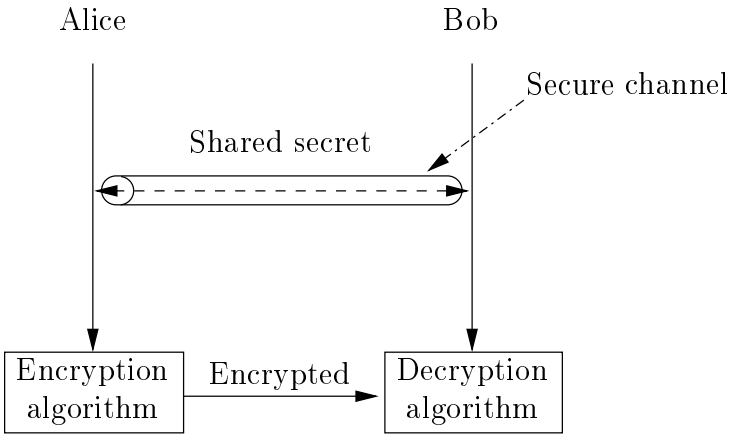

Figure 1.1: Two parties using a symmetric algorithm.

Public key distribution systems offer an approach which eliminates the need for a secure key distribution channel. Diffie and Hellman introduced the first practical public key distribution system, [3]. A current example of a public key distribution system is the program PGP[3] written by Phil Zimmerman [21].

1.2 Public Key Distribution Systems

An algorithm used for a public key distribution system is called an *asymmetric algorithm*. A public key system consists of the following components:

- System parameters: fix some settings in the algorithm.
- Private key, *priv*: the value the user must keep secret.
- Public key, *pub*: the value that is publicly known and used for communication with the owner of the associated private key.
- Combining function, $F(priv, pub)$: the function used to create the shared secret.

[3]Pretty Good Privacy

The introduction of the public key distribution systems removes the need for initial secure communication. However the public keys must belong to the user whom the communication is intended for. Obtaining an authenticated public key is not a trivial problem, but it is not the focus of this dissertation and therefore will not receive further attention.

Before the protocol can be initiated the users, Alice and Bob, must first generate their public and private key pairs, called $(pub_A, priv_A)$ and $(pub_B, priv_B)$ respectively. The public keys are then distributed over an insecure channel to the other users where the combining function computes the shared secret. By selecting any symmetric algorithm (C) a secure channel can be created by using the shared secret as the key for the symmetric algorithm. An illustration of the communication between Alice and Bob is given in Figure 1.2. The shared secret computed by Bob is given by $F(priv_B, pub_A)$, and must be the same as the shared secret computed by Alice, which is given by $F(priv_A, priv_B)$. For this reason a requirement for the combining function is that

$$F(priv_A, pub_B) = F(priv_B, pub_A) \tag{1.1}$$

for all users Alice and Bob.

For a public key distribution system to be secure, it should not be computationally feasible to deduce a private key, or any key equivalent of it, from a public key. Furthermore, the combining function must at least satisfy the following conditions to ensure that the shared secret is in fact a secret.

- It must not be computationally feasible to determine information about the shared secret from the information exchanged in the protocol.
- The combining function must appear as though acting randomly with respect to its input.
- It must be infeasible to compute the private key from the public key.

All known asymmetrical algorithms are significantly slower than symmetrical algorithms. Hence, a combination of symmetrical and asymmetrical

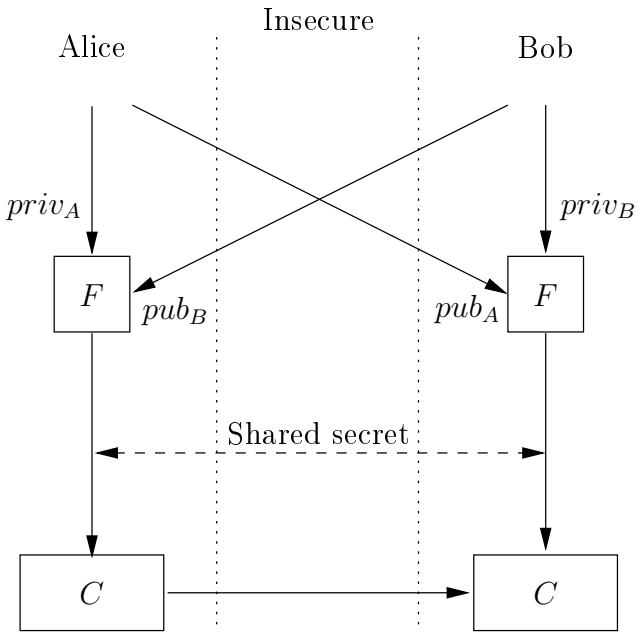

Figure 1.2: Two parties using a public key distribution system.

algorithms are used in order to optimise the speed without compromising security.

1.3 Computation

The suitability of a protocol for an implementation, is determined by its computational efficiency and its data transmitting efficiency during communication. A theoretical measure, the O-notation, is used to determine how efficient a protocol is. An advantage of using the O-notation is that it can be used independently from the machine the protocol is implemented on. The O-notation measures the complexity of an algorithm.

Definition 1.1.

(1) A *partial* function $f : \mathbb{N} \mapsto \mathbb{R}$ is a function that needs not be defined for all $n \in \mathbb{N}$ and is called *eventually positive* if there is a constant $N \in \mathbb{N}$ and such that $f(n)$ is defined and strictly positive for all $n \geq N$.

(2) Let $g : \mathbb{N} \mapsto \mathbb{R}$ be eventually positive. Then $O(g)$ is the set of all eventually positive functions $f : \mathbb{N} \mapsto \mathbb{R}$ for which there exist $N, c \in \mathbb{N}$ such that $f(n)$ and $g(n)$ are defined and $f(n) \leq cg(n)$ for all $n \geq N$.

Complexity analysis of an exponentiation algorithm is given in the next example.

Example 1.2 (Repeated Squaring)**.** The power of a number is frequently computed. In this example a simple but efficient algorithm is demonstrated, namely the repeated squaring technique, see Algorithm 1.

Algorithm 1: Left-to-right exponentiation

Data: $\alpha \in \mathbb{Z}, n = \left(n'_{r-1}, \ldots, n'_0\right)_2$
Output: $M = \alpha^n$

```
1 M ← 1;
2 for i = r − 1 to 0 do
3     if n'_i = 1 then
4         M ← M · α;
5     end
6     M ← M · M;
7 end
```

The input of the algorithm is taken as the length of the exponent in its binary representation, i.e. $r = \log_2(n)$. The operations that are of interest in the complexity analysis are multiplication and squaring.

Squaring and multiplication is implemented in the current context as multiprecision integers. In the squaring operation of a multiprecision integer, symmetries arise that can be used to reduce the number of single precision multiplications. Thus it is assumed that squaring takes 80% of the time to perform multiplication. The bit of the exponent under consideration in line 3 determines whether multiplication is also being performed. Therefore, the time needed to perform the exponentation is

$$H_n M + r(1.8M) = (H_n + 1.8r)M$$

where H_n is the Hamming weight of the exponent and M the time needed to perform multiplication. Only using repeated multiplication would require

n multiplication, which is larger than or equal to $2r$. Note that $H_n \leq r$. Thus, repeated squaring is more efficient. In terms of the O-notation, the complexity is linear, i.e. $O(r)$.

The O-notation can be seen as an asymptotic measure, but the protocol will only be implemented with finite possible inputs. This difference raises the question about the validity of using the O-notation.

The comparison of the complexity analysis with implementations with practical inputs will be investigated here, Chapter 5. The operations of interest in the analysis are multiplication, squaring and addition. These were selected, as they are the operations used in the finite field.

Chapter 2

Diffie-Hellman Protocol

In [3] the Diffie-Hellman protocol is introduced; it was the first practical public key distribution protocol. Despite its age, some of the mainstream asymmetric protocols are still based on it, such as Elliptic Curves and XTR. One of the criticisms of the Diffie-Hellman protocol is that it is more complicated than RSA, as knowledge of finite fields is needed for the implementation of the Diffie-Hellman protocol.

In this chapter, the protocol is given as well as how the security of the protocol is evaluated. The security of the Diffie-Hellman Protocol is formalised in the Diffie-Hellman, Discrete Logarithm and Diffie-Hellman Decision Problems. Four attacks on these problems will be discussed and each of these attacks highlights a vulnerability of the Diffie-Hellman Protocol. Currently, the security of the protocol is determined by the choice of parameters that make known attacks not feasible.

2.1 The Protocol

The protocol requires a cyclic group G of order n and a primitive element α in G that generates all the elements $1, \alpha, \alpha^2, \ldots, \alpha^{n-1}$ of G. The system's parameters α and G are public knowledge. The group $<\alpha>$ is referred to as the *Diffie-Hellman group*.

The private keys of Alice and Bob are integers in the interval $[2, \text{ord}(\alpha))$,

$priv_A$ and $priv_B$ respectively. The public keys of Alice and Bob are now calculated as follows:

$$pub_A = \alpha^{priv_A}$$
$$pub_B = \alpha^{priv_B}.$$

The public keys are exchanged between Alice and Bob via an insecure channel.

To compute the shared secret between Alice and Bob, the combining function $F : \mathbb{Z}_{n-1} \setminus \{0, 1\} \times G \to G$ is defined by

$$F(x, y) = y^x.$$

Now the combining function satisfies (1.1):

$$\begin{aligned} F(priv_A, pub_B) &= \left(\alpha^{priv_B}\right)^{priv_A} \\ &= \left(\alpha^{priv_A}\right)^{priv_B} \\ &= F(priv_B, pub_A). \end{aligned}$$

A summary of the protocol is given in Table 2.1.

	Alice	Bob
Common	$<\alpha> = G$	
Private	$priv_A$	$priv_B$
Public	α^{priv_A}	α^{priv_B}
Shared Secret	$\alpha^{priv_A priv_B}$	

Table 2.1: Diffie-Hellman Protocol.

Example 2.1. The Diffie-Hellman key exchange for $\alpha = 7$, using $\mathbb{Z}_{13}^*$ as the cyclic group.

Let the private keys for Alice and Bob be:

$$priv_A = 10$$
$$priv_B = 5.$$

The public keys are then

$$\begin{aligned} pub_A &= 7^{10} \pmod{13} \\ &= 4 \end{aligned}$$

and

$$\begin{aligned} pub_B &= 7^{5} \pmod{13} \\ &= 11. \end{aligned}$$

The shared secret for Alice and Bob is thus

$$7^{5\cdot 10} \pmod{13} = 10.$$

2.2 Security of the Diffie-Hellman Protocol

In Section 1.2 three necessary properties for a secure public key distribution system are stipulated. In the case of the Diffie-Hellman Protocol, these properties translate into the following three problems.

Definition 2.2 (Discrete Logarithm Problem)**.** [18, Definition 2] Let G be a finite cyclic group generated by α. The problem of computing from $\beta \in G$ a number s such that $\alpha^s = \beta$ is called the *Discrete Logarithm Problem*. Notation: $s = DL_\alpha(\beta)$.

Definition 2.3 (Diffie-Hellman Problem)**.** [18, Definition 1] Let G be a finite cyclic group with generator α. The problem of computing α^{ab} from α^a and α^b is called the *Diffie-Hellman Problem*. Notation: $\alpha^{ab} = DH(\alpha^a, \alpha^b)$.

Definition 2.4 (Diffie-Hellman Decision Problem)**.** [18, Definition 3] Let G be a finite cyclic group generated by α. Let $\alpha^a, \alpha^b, \alpha^c$ be chosen independently and randomly in G according to the uniform distribution. Given the triples $(\alpha^a, \alpha^b, \alpha^{ab})$ and $(\alpha^a, \alpha^b, \alpha^c)$ in random order, the *Diffie-Hellman Decision Problem* is to decide, with a probability greater than $1/2$, which of the triples is the correct Diffie-Hellman triple.

The intractability of these problems will ensure that the Diffie-Hellman Protocol has the properties mentioned in Section 1.2. Although all three problems are important for the security of the Diffie-Hellman Protocol, the Discrete Logarithm Problem will be the main focus of this dissertation.

It is evident that solving the Discrete Logarithm Problem is sufficient for solving the Diffie-Hellman Problem and the Diffie-Hellman Decision Problem. It is also evident that solving the Diffie-Hellman problem is sufficient for solving for the Diffie-Hellman Decision Problem.

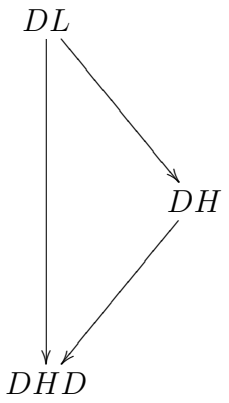

Figure 2.1: Evident relations between problems Diffie-Hellman Problems.

Indeed, if α^a and α^b are given, then

$$\alpha^{ab} = (\alpha^a)^{DL_\alpha(\alpha^b)}$$

and if a triple $(\alpha^a, \alpha^b, \alpha^c)$ is given, then it can be determined whether this is a Diffie-Hellman triple by testing whether

$$\alpha^c = (\alpha^a)^{DL_\alpha(\alpha^b)}$$

or

$$\alpha^c = DH(\alpha^a, \alpha^b).$$

Whether one can solve the Diffie-Hellman Decision Problem by methods other the discrete logarithm poses an important problem. The following example shows that in certain cases it is very easy to solve the Diffie-Hellman Decision Problem without making use of solutions from the Discrete Logarithm Problem or the Diffie-Hellman Problem.

Example 2.5 (Difficulty of Diffie-Hellman Decision)**.** This example shows that in certain cases, solving the Diffie-Hellman Decision Problem is easy. Use the notation of Definition 2.4 and let the order of the group G be $2p$, where p is prime. For a triple $\left(\alpha^a, \alpha^b, \alpha^c\right)$ to be a valid Diffie-Hellman triple, it must be the case that

$$2p|(ab - c),$$

hence

$$ab \equiv c \pmod 2. \tag{2.1}$$

Relation (2.1) holds if, and only if, one of the following holds:

(1) $(2|a$ or $2|b)$ and $2|c$,

(2) $2 \nmid a$ and $2 \nmid b$ and $2 \nmid c$.

Relation (2.1) fails if, and only if, one of the following relations holds:

(1) $(2|a$ or $2|b)$ and $2 \nmid c$,

(2) $2 \nmid a$ and $2 \nmid b$ and $2|c$.

To determine if a triple is a Diffie-Hellman triple, Algorithm 2 is used. The algorithm determines if (2.1) is satisfied. If relation (2.1) is not satisfied the triple is not a valid Diffie-Hellman pair. If the relation is not valid the output of the algorithm is random. Thus the probability that the algorithm is correct is 0.5 if relation (2.1) holds and 1 if the relation does not hold. Since the relation holds with a probability of 0.5, the algorithm provides the correct answer is obtained with a probability of 3/4.

If the relation is valid no information is obtained and a random decision is made. Thus for a random input the relation will be false of 50 percent of the input and for the other triple the probability that a correct answer is give, is 0.5. Thus for any triple the correct answer is obtained with a probability of 0.5.

Algorithm 2: Solve the Diffie-Hellman Decision Problem, Example 2.5

Data: $\left(\alpha^a, \alpha^b, \alpha^c\right)$ the triple to test
Output: If the Triple is valid

$g_a \leftarrow (\alpha^a)^p$;
$g_b \leftarrow \left(\alpha^b\right)^p$;
$g_c \leftarrow (\alpha^c)^p$;
if $g_a = g_b = 1$ *and* $g_c \neq 1$ **then**
    return Not Valid Triple;
end
if $g_a = g_b \neq 1$ *and* $g_c = 1$ **then**
    return Not Valid Triple;
end
Randomly select $r \in \mathbb{Z}_2$;
if r **then**
    return Not Valid Triple;
end
return Valid Triple;

2.3 Attacks on the Discrete Logarithm Problem

Certain attacks, as indicated below, are considered for determining the parameters of the Diffie-Hellman protocol. These attacks are not the most efficient variants, but illustrate the weakness of the protocol. In certain attacks, the order of the Diffie-Hellman group needs to be factorised. It is therefore assumed that the factors can be computed in a reasonable time.

The attacks considered are:

- Pollard ρ: Random search.

- Chinese Remainder Theorem: Factorisation of the order of the Diffie-Hellman group.
- Pohlig-Hellman: Provides a method to compute the discrete logarithm in a cyclic group of prime order.
- Index Calculus: Using the smallest field containing the Diffie-Hellman group.

A detailed survey of attacks on the Discrete Logarithm Problem can be found in [19].

In each of the following attacks, let G be a cyclic group of order n, with generator α, and let $\beta \in G$ be any element of which the logarithm needs to be computed, i.e. for the equation $\alpha^r = \beta$, (r is an integer) where r must be solved.

2.3.1 Pollard ρ

The Pollard ρ algorithm solves for r by first finding integers m and s such that

$$\beta^m = \alpha^s.$$

The above integers are found by constructing a sequence $x_0, x_1, x_2, \ldots$ of elements in G and then determining an index m such that

$$x_{2m} = x_m,$$

a collision. This method of finding an m such that $x_m = x_{2m}$ is called *Floyd's cycle-finding* algorithm [24].

Floyd's cycle-finding algorithm adds computation complexity, by removing the need to store all the elements in the sequence until a collision is found. For any sequence $x_0, x_1, x_2, \ldots$, if j is the smallest index such that $x_j = x_i$ for $0 \leq i < j$, then there exists an m' with $i \leq m' \leq j$ such that $x_{m'} = x_{2m'}$: If $k = j - i$ then for any $l \geq i$ and $t \geq 0$, it follows that $x_l = x_{l+tk}$. In

particular, if

$$l = m' = k\left\lceil \frac{i}{k} \right\rceil$$

and

$$t = \left\lceil \frac{i}{k} \right\rceil$$

then $x_{m'} = x_{2m'}$ and $i \leq m' \leq j$.

Divide the group G into three disjoint and roughly equal, in cardinality, sets S_1, S_2 and S_3, from which the sequence $x_0, x_1, x_2, \ldots$ is obtained by setting $x_0 = 1$ and

$$x_i = \begin{cases} \beta x_{i-1} & \text{if } x_{i-1} \in S_1 \\ x_{i-1}^2 & \text{if } x_{i-1} \in S_2 \\ \alpha x_{i-1} & \text{if } x_{i-1} \in S_3 \end{cases}$$

with $i \geq 1$. The elements of the sequence can be written as $x_i = \beta^{b_i}\alpha^{a_i}$ for $i \geq 0$ and $a_0 = b_0 = 0$. Therefore, for each $i \geq 1$ it follows that

$$(a_i, b_i) = \begin{cases} (a_{i-1}, b_{i-1} + 1) & \text{if } x_{i-1} \in S_1 \\ (2a_{i-1}, 2b_{i-1}) & \text{if } x_{i-1} \in S_2 \\ (a_{i-1} + 1, b_{i-1}) & \text{if } x_{i-1} \in S_3. \end{cases}$$

Compute the 6-tuple $(x_i, a_i, b_i, x_{2i}, a_{2i}, b_{2i})$ for $i = 1, 2 \ldots$ until $x_i = x_{2i}$. Then

$$\beta^{b_i}\alpha^{a_i} = \beta^{b_{2i}}\alpha^{a_{2i}}.$$

Let $m = b_i - b_{2i} \pmod{\text{ord}(\alpha)}$ and $s = a_{2i} - a_i \pmod{\text{ord}(\alpha)}$, then

$$\beta^m = \alpha^s \tag{2.2}$$

and

$$mDL_\alpha(\beta) \equiv s \pmod{\text{ord}(\alpha)}.$$

Case 1: If m and n are relatively prime, then the discrete logarithm can be solved directly by

$$DL_\alpha(\beta) \equiv sm_{-1} \pmod{\text{ord}(\alpha)},$$

where m_{-1} is such that

$$mm_{-1} \equiv 1 \pmod{\text{ord}(\alpha)}.$$

Case 2: If $(m, \text{ord}(\alpha)) = m' > 1$ use the extended Euclidean Algorithm to compute u and v such that

$$m' = u \cdot m + v \cdot \text{ord}(\alpha). \tag{2.3}$$

Then from (2.2)

$$\begin{aligned} \beta^{m'} &= \beta^{u \cdot m + v \cdot \text{ord}(\alpha)} \\ &= \beta^{u \cdot m} \\ &= \alpha^{us}. \end{aligned}$$

By substituting β by α^r in the above equation,

$$\alpha^{rm'} = \alpha^{us},$$

with r and m' fixed. While u can be selected arbitrarily large, it can thus be assumed that us is divisible by m'. The above equation can be stated as follows:

$$\beta^{m'} = \alpha^{m'l}$$

thus

$$\beta = \alpha^{l+(i\text{ord}(\alpha)/m')} \text{ with } 1 \leq i \leq m'$$

and

$$\begin{aligned} DL_\alpha(\beta) &= r \\ &\equiv l + (in/m') \pmod{n} \end{aligned}$$

for a value of i.

This indicates how to calculate the value r, such that $\beta = \alpha^r$.

The following theorem is used to estimate the complexity of the Pollard ρ algorithm.

Theorem 2.6 (The birthday paradox)**.** [24, Theorem 2.4] *Suppose that $0 < k \ll n$ and independently selects k random integers between 1 and n, with equal probability. The probability $P(n,k)$, that of two selected numbers are equal, is approximately $1 - e^{-k(k-1)/(2n)}$.*

Proof. Randomly select k integers independently and uniformly from the set $[1,n]$ and let $P(n,k)$ denote the probability that two of the selected numbers are equal. The probability that all the selected elements are distinct is

$$\frac{n(n-1)\cdots(n-k+1)}{n^k},$$

thus

$$P(n,k) = 1 - \frac{n(n-1)\cdots(n-k+1)}{n^k}.$$

The probability can be written as

$$P(n,k) = 1 - \left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right)\cdots\left(1 - \frac{k-1}{n}\right).$$

The factors $(1 - i/n)$ for $1 \leq i < k$ will be approximated. If i/n is small,

then $1 - i/n \approx e^{-i/n}$. The probability can now be approximated by

$$\begin{aligned} P(n,k) &\approx 1 - e^{-1/n}e^{-2/n}\cdots e^{-(k-1)/n} \\ &= 1 - e^{-(1/n+2/n+\cdots+(k-1)/n)} \\ &= 1 - e^{-k(k-1)/2n}. \end{aligned}$$

□

Assume that the sequence $x_1, x_2, \ldots$ is chosen in the Pollard ρ algorithm such that it behaves like a random walk through the group G. Theorem 2.6 is then used to estimate the complexity. If a success rate of approximately 90% is needed, then from $1 - e^{-2.5} \approx 0.9$ it is required that

$$\begin{aligned} -k(k-1)/(2\text{ord}(\alpha)) &\approx -2.5 \\ \therefore k^2 &\approx 5 \cdot \text{ord}(\alpha), \end{aligned}$$

where the collision in the sequence of elements occurs after k iterations. Note that k is the index of the largest element computed in the sequence. As Floyd's algorithm is used, it is needed that k is replaced by $2k$. Thus,

$$\begin{aligned} k^2 &\approx (1.25)\text{ord}(\alpha) \\ \text{i.e. } k &= O(\sqrt{\text{ord}(\alpha)}). \end{aligned}$$

This gives a complexity of $O(\sqrt{\text{ord}(\alpha)})$. Since the success rate only changes the number of iterations needed, the order of the complexity of the algorithm will not change. In [22] it is proven that the lower bound of complexity of a generic method to solve the Discrete Logarithm Problem is $O(\sqrt{\text{ord}(\alpha)})$.

Example 2.7. Consider the group $\mathbb{Z}_{13}^*$ with generator $\alpha = 7$. Let $\beta = 5$ be the element of which the discrete logarithm needs to be computed.

Divide the group into three sets as follows

$$S_1 = \{3, 4, 9, 10\}$$
$$S_2 = \{2, 6, 11, 12\}$$
$$S_3 = \{1, 5, 7, 8\}.$$

The sequence is now defined by

$$x_i = \begin{cases} 5x_{i-1} & \text{if } x_{i-1} \in S_1 \\ x_{i-1}^2 & \text{if } x_{i-1} \in S_2 \\ 7x_{i-1} & \text{if } x_{i-1} \in S_3. \end{cases}$$

Table 2.2 gives the values of the 6-tuples $(x_i, a_i, b_i, x_{2i}, a_{2i}, b_{2i})$ for the first four iterations.

i	x_i	a_i	b_i	x_{2i}	a_{2i}	b_{2i}
0	1	0	0	1	0	0
1	7	0	1	10	0	2
2	10	0	2	4	2	4
3	11	1	2	10	3	5
4	4	2	4	4	8	10

Table 2.2: Sequence generated in Example 2.7.

This table indicates that the first collision occurs when $i = 4$ using Floyd's method. Also notice that the first collision occurs when $i = 2$ and $j = 6$, if Floyd's cycle-finding technique is not used. With a collision at $i = 4$, the variables in (2.2) are

$$\begin{aligned} m &= a_4 - a_8 \\ &= 2 - 8 \\ &\equiv 6 \pmod{12} \end{aligned}$$

and

$$\begin{aligned} s &= b_8 - b_4 \\ &= 10 - 4 \\ &\equiv 6 \pmod{12}. \end{aligned}$$

Since $(6, 12) = 6 = m' > 1$ the extended Euclidean Algorithm is used to find the next set of variables as defined in (2.3):

$$\begin{aligned} u &= -1 \\ v &= 1. \end{aligned}$$

Thus

$$\beta^6 = \alpha^{-6},$$

and

$$l = 1.$$

The solution is in now in the set

$$\begin{aligned} &\{1 - i \cdot 12/6 | i = 0, 1, \ldots, 11\} \\ =&\{1, 10, 9, 7, 5, 3\}. \end{aligned}$$

By exhaustive search it follows that the logarithm is 3.

2.3.2 Chinese Remainder Theorem

The Discrete Logarithm Problem will be solved in a group by considering the same problem in its subgroups. This is achieved through a divide and conquer attack that uses the factorisation of ord($<\alpha>$) to divide the problem. The Chinese Remainder Theorem uses the solutions in these subgroups to provide the solution in the group $<\alpha>$.

Theorem 2.8 (Chinese Remainder Theorem)**.** [4, Theorem 2.3.1] *Let* $\{n_1, \ldots, n_k\}$ *be a set of positive integers which are relatively prime. For any set of integers* $\{a_1, \ldots, a_k\}$ *the system of congruences*

$$x \equiv a_i \pmod{n_i}, \quad i = 1, 2, \ldots, k,$$

has exactly one solution modulo $n = n_1 n_2 \cdots n_k$.

Proof. Define

$$c_i = a_i (M_i)_{-1} \pmod{n_i}$$

where

$$M_i = \prod_{\substack{j=1 \\ j \neq i}}^{k} n_j \text{ and } M_i (M_i)_{-1} \equiv 1 \pmod{n_i}.$$

Then the value

$$y = \sum_{i=1}^{k} c_i M_i \pmod{n},$$

is a solution since

$$\begin{aligned} y &\equiv c_i M_i \pmod{n_i} \\ &\equiv \big(a_i (M_i)_{-1}\big) M_i \pmod{n_i} \\ &\equiv a_i \pmod{n_i}. \end{aligned}$$

If x and y are two solutions, then $x - y \equiv 0 \pmod{n_i}$ for $i = 1, \ldots, k$. It follows that $x \equiv y \pmod{n}$. This proves uniqueness of the solution. □

Let $n_1 n_2 \cdots, n_k$ be a factorisation of n, where the n_i's are relatively prime and let $M_i = n/n_i$, i.e $(M_i, n_i) = 1$. Define a function

$$f : \mathbb{Z}_n \to \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$$

by

$$f(m) = (m \pmod{n_1}, \ldots, m \pmod{n_k}).$$

By the Chinese Remainder Theorem, the function f is a surjection and since the cardinalities of the domain and range of f are the same, f is a bijection. The inverse of f is given by

$$f^{-1}(a_1, \ldots, a_k) = \sum_{i=1}^{k} c_i M_i \pmod{n},$$

where

$$c_i = a_i (M_i)_{-1} \pmod{n_i}.$$

To compute the discrete logarithm of $\beta = \alpha^r$ where $G = <\alpha>$ and $\text{ord}(\alpha) = n$, factorise n into relatively prime numbers, $n_1, n_2 \ldots, n_k$. Then there exists an isomorphism

$$<\alpha> \cong < \alpha_1 > \times \cdots \times < \alpha_k >$$

where $\alpha_i \in <\alpha>$ and $\text{ord}(\alpha_i) = n_i$ for $i = 1, 2, \ldots, k$. Define a function

$$I : <\alpha> \mapsto <\alpha_1> \times \cdots \times <\alpha_k>$$

by

$$I : \alpha^m \mapsto \left(\alpha_1^{m \ (mod \ n_1)}, \ldots, \alpha_k^{m \ (mod \ n_k)}\right).$$

The function I is a bijection as it has an inverse defined by

$$\left(\alpha_1^{a_1}, \ldots, \alpha_{n_k}^{a_{n_k}}\right) \mapsto \alpha^m,$$

where $m = f^{-1}(a_1, \ldots, a_{n_k})$. Now it follows from

$$\begin{aligned} I\left(\alpha^m \cdot \alpha^l\right) &= \left(\alpha_1^{m+l \pmod{n_1}}, \ldots, \alpha_k^{m+l \pmod{n_k}}\right) \\ &= \left(\alpha_1^{m \pmod{n_1}}, \ldots, \alpha_k^{m \pmod{n_k}}\right) \cdot \left(\alpha_1^{l \pmod{n_1}}, \ldots, \alpha_k^{l \pmod{n_k}}\right) \\ &= I(\alpha^m) \cdot I\left(\alpha^l\right) \end{aligned}$$

that I is an isomorphism.

Algorithm 3 is used to compute the discrete logarithm in G. The 'for loop' computes the discrete logarithm of the i^{th} component of $I(\beta)$ in $< \alpha_i >$. The last line of the algorithm uses the Chinese Remainder Theorem to compute the discrete logarithm of β.

Algorithm 3: Chinese Remainder Theorem computing the discrete logarithm.

Data: $n = \prod_{i-1}^{k} n_i$ and α^c where $<\alpha> = G$ and $\text{ord}(G) = n$
Output: Ensure $c = DL_\alpha(\alpha^c)$

```
1 β ← α^c;
2 for i = 1 to k do
3     β_i ← β^{n/n_i};
4     c_i ← DL_{α_i}(β_i);
5     x_i ← c_i · (n/n_i (mod n_i))^{-1};
6 end
7 c = x_1 · n/n_1 + ··· + x_k · n/n_k (mod n);
```

The time needed to compute the discrete logarithm can be estimated by

$$\begin{aligned} time(DL_\alpha(\beta)) &= \sum_{i=1}^{k} \left(time(DL_\alpha(\beta_i)) + m_i\right) \\ &\leq k \cdot \max\left\{DL_\alpha(\beta_i) + m_i\right\} \\ &\approx k \cdot \max\left\{DL_\alpha(\beta_i)\right\}, \end{aligned}$$

where m_i is a small value for the time needed to execute the other instructions in the 'for loop'. This is an efficient divide and conquer algorithm, where the complexity is dependent on the largest factor of the order of the group.

Algorithm 3 can be used before applying most of the other algorithms used to solve the Discrete Logarithm Problem.

Example 2.9. Consider the group $G = \mathbb{Z}_{13}^*$ with generator $\alpha = 7$ and $\beta = 6$. The order of $\mathbb{Z}_{13}^*$ is $12 = 2^2 \cdot 3$ and the factorisation is

$$< 7 > \cong < 5 > \times < 9 >,$$

where $\text{ord}(< 5 >) = 4$ and $\text{ord}(< 9 >) = 3$. The element 6 is represented in $< 5 > \times < 9 >$ by $(8, 9)$. The logarithm of 9 in $< 9 >$ is 1. The power list of 5 in $< 5 >$ is

$$5, 12, 8, 1. \tag{2.4}$$

Thus the logarithm of 8 in $< 5 >$ is 3 and

$$\begin{aligned} x_1 &= 3 \cdot 3_{-1} \pmod 4 \\ &\equiv 1, \\ x_2 &= 4_{-1} \pmod 3 \\ &\equiv 1. \end{aligned}$$

The logarithm of 6 in $\mathbb{Z}_{13}^*$ is now given by

$$12/4 + 12/3 = 7 \pmod{12}.$$

2.3.3 Pohlig-Hellman

The Pohlig-Hellman algorithm is similar to the Chinese Remainder Theorem. Both the attacks divide the cyclic group $G = <\alpha>$ into subgroups. In the case of the Pohlig-Hellman algorithm, the order of these cyclic groups must be a prime power, i.e.

$$<\alpha> \cong <\alpha_1> \times \cdots \times <\alpha_k>$$

where each α_i is a generator of a cyclic group of order $p_i^{\lambda_i}$, with p_i a prime factor of the ord(α). Thus the Pohlig-Hellman algorithm can be seen as a special case of the Chinese Remainder Theorem attack. The Pohlig-Hellman algorithm is Algorithm 3 where the n_i's are prime numbers and the method of solving DL_α in line 4 is specified next.

For each prime factor p of ord($<\alpha>$) the exponent r in $\beta = \alpha^r$, can be written in its radix expansion,

$$r = \sum_{i=0}^{\lambda-1} s_i p^i \text{ with } 0 \leq s_i \leq p-1. \tag{2.5}$$

To remove cluttering of notation the subscript i is removed. Note that the value of r is not used in the algorithm, it is in fact the values of s_i that are determined. After all the s_i for every prime factor of ord($<\alpha>$) are determined, the Chinese Remainder Theorem is used to compute the value of r.

The values of the s_i are determined iteratively. In the subgroup of order p^l notice that

$$\begin{aligned}
\beta^{\text{ord}(\alpha)/p} &= \alpha^{r\cdot\text{ord}(\alpha)/p} \\
&= \alpha^{\text{ord}(\alpha)\sum_{j=0}^{\lambda-1} s_j p^{j-1}} \\
&= \alpha^{\text{ord}(\alpha)s_0/p} \\
&= c_0^{s_0}
\end{aligned}$$

where $c_0 = \alpha^{\text{ord}(\alpha)/p}$ is a primitive p^{th} root of unity in the current group $<\alpha^{\text{ord}(\alpha)/p}>$. Through exhaustive search on all possible values of $s_0 \in [1, p)$, the correct value can be determined.

To determine the value of an s_i, a similar computation is performed in

the group $<\alpha^{\text{ord}(\alpha)/p^{i+1}}>$. Suppose $s_0, \ldots, s_{i-1}$ are known and

$$\begin{aligned} r_i &= \sum_{m=i}^{\lambda-1} s_m p^m \text{ with } 0 \leq s_m \leq p-1 \\ &= r - \sum_{m=0}^{i-1} s_m p^m. \end{aligned}$$

Then s_i can be computed by noticing that

$$\begin{aligned} a_i^{\text{ord}(\alpha)/p^{i+1}} &= \alpha^{r_i \cdot \text{ord}(\alpha)/p^{i+1}} \\ &= \alpha^{\text{ord}(\alpha)\sum_{j=0}^{\lambda-1-i} s_{j+i}p^{j-1}} \\ &= \alpha^{\text{ord}(\alpha)s_i/p} \\ &= c_i^{s_i}, \end{aligned}$$

where $c_0 = \alpha^{\text{ord}(\alpha)/p}$ is a primitive p^{th} root of unity in the current group. The value of s_i is determined by testing all possible values until

$$a_i^{\text{ord}(\alpha)/p^{i+1}} = c_i^{s_i}.$$

By using an algorithm for fast exponentiation and improving the search for the exponent s_i, the complexity [19] of solving the Discrete Logarithm Problem is given by

$$O\left(\sum_{i=1}^{k} \lambda_i(\log \text{ord}(\alpha) + p_i)\right).$$

For a detailed discussion of the algorithm see [19].

Example 2.10. In Example 2.9 the logarithm of 6 with respect to 7 is computed by considering the logarithm in $<5>$ and $<9>$. Since the logarithm is trivial in $<9>$, only the logarithm of 8 in $<5>$ is considered, i.e. $DL_5(8)$.

Since $\text{ord}(<5>) = 4$, the only two subgroups that need to be considered are those of order 2 and 4. The intermediate values of the algorithm are given in Table 2.3.

i	a_i	c_i	s_i	p^i
0	8	12	1	2
1	12	12	1	4

Table 2.3: Intermediate results in Example 2.10.

The logarithm can be written in the form

$$r = s_0 + s_1 \cdot 2 \text{ where } s_i \in \{0, 1\}.$$

The first iteration gives the relation

$$\begin{aligned} 8^{4/2} &\equiv 12 \\ &\equiv c_0^{s_0} \pmod{13}. \end{aligned}$$

and since 12 is primitive in $<5>$; $c_0 = 12$ and it follows that $s_0 = 1$. The second iteration gives the relation

$$\begin{aligned} 8 \cdot 5^{-1} &\equiv 8 \cdot 5^3 \\ &\equiv 12 \\ &\equiv c_1^{s_1} \pmod{13} \end{aligned}$$

and thus $s_1 = 1$.

The logarithm is of 8 with generator 5 is

$$1 + 1 \cdot 2 = 3.$$

2.3.4 Index Calculus

The Index Calculus method gives rise to a class of probabilistic algorithms to find the logarithm of an element β in a group $G = <\alpha>$ of order n. The method consists of two stages, namely pre-computation and computation of the individual logarithm.

The pre-computation stage is executed once for a specific group. In the pre-computation stage relations are created, which are dependent on the

group that is attacked. The second stage uses the relations from the pre-computation phase to compute the discrete logarithm.

Pre-computation

Let the set $S \subset G$ consist of elements $\rho_i \in G$. A random positive integer a is selected and it is attempted to write α^a as a product of the elements in S,

$$\alpha^a = \prod_{i=1}^{t} \rho_i^{\lambda_i}, \tag{2.6}$$

where t is the size of S. The above equation gives the congruence

$$a \equiv \sum_{i=1}^{t} \lambda_i DL_\alpha(\rho_i) \pmod{n}. \tag{2.7}$$

By considering the $DL_\alpha(\rho_i)$'s as indeterminate in (2.7), a system of equations can be constructed. A solution for such a system of equations is necessary for the success of the pre-computation phase.

By increasing the size of the set S, the work needed for the pre-computation phase increases and the probability that the computation of the individual logarithm will succeed is increased.

Computing an individual logarithm

To compute the logarithm of a given β, integers s are randomly selected until $\alpha^s\beta$, can be written as a product of elements in S, i.e.

$$\alpha^s\beta = \prod_{i=1}^{t} \rho_i^{b_i}. \tag{2.8}$$

If such a relation cannot be found, the method fails. It follows that

$$DL_\alpha(\beta) \equiv \sum_{i=1}^{t} b_i DL_\alpha(\rho_i) - s \pmod{n}.$$

Consider an implementation in $\mathbb{F}_{p^n}^*$. Since the elements of S are being used to 'factorise' numbers, it is natural to select irreducible elements for the set S. But there are no irreducible elements in the $\mathbb{F}_{p^n}$ since every element

has an inverse. However, the irreducible elements are selected from $\mathbb{F}_p[x]$ since the polynomials in $\mathbb{F}_p[x]$ of degree less than n represent the elements in $\mathbb{F}_{p^n}$ uniquely. Therefore the irreducible polynomials in $\mathbb{F}_p[x]$ of degree less than n generate the elements in $\mathbb{F}_{p^n}^*$. The set S will consist of irreducible polynomials of degree less then k, $k \leq n$, where the parameter k determines the probability that the algorithm will be successful. By reducing the value of k, the number of elements in S decreases which in turn reduces the chance that the relation in (2.6) will be found. Reduction in the size of S also reduces the computations needed to complete the pre-computation phase.

To solve the system of equations given by (2.7), Gauss elimination or other techniques as illustrated in Example 2.11 can be used.

Example 2.11. Consider the field $\mathbb{F}_{2^6}$ with defining primitive polynomial $f(x) = x^6 + x + 1 \in \mathbb{F}_2[x]$. Then $\alpha = x$ is a primitive element of $\mathbb{F}_{2^6}^*$, being a root of the polynomial. We determine the logarithm of $\beta = x^4+x^3+x^2+x+1$ to the base x.

Let $S = \{x, x+1, x^2+x+1\}$ be the set of irreducible polynomials of degree at most 2 in $\mathbb{F}_p[x]$. To compute the logarithm of these elements let $1 \leq a \leq 63$. Evidently $DL_\alpha(x) = 1$. Let $a = 6$ then

$$x^6 \equiv x + 1 \pmod{f(x)},$$

giving $DL_\alpha(x+1) = 6$. Before computing the case $a = 32$, note that

$$\begin{aligned} x^{64} &\equiv x \\ &\equiv x^6 + 1 \\ &\equiv \left(x^3 + 1\right)^2 \pmod{f(x)} \end{aligned}$$

giving

$$\begin{aligned} x^{32} &\equiv x^3 + 1 \\ &\equiv (x+1)\left(x^2 + x + 1\right) \pmod{f(x)} \end{aligned}$$

and

$$\begin{aligned} 32 &\equiv DL_\alpha(x+1) + DL_\alpha(x^2+x+1) \\ &\equiv 6 + DL_\alpha(x^2+x+1) \pmod{63} \end{aligned}$$

thus $DL_\alpha(x^2+x+1) = 26$.

To get a relation in the form of (2.8), let $s = 2$. That is,

$$\begin{aligned} \beta \cdot \alpha^2 &\equiv x^5 + x^4 + x^3 + x^2 + x + 1 \\ &\equiv \left(x^2+x+1\right)^2 (x+1) \pmod{f(x)}. \end{aligned}$$

Since all the factors are in S, the discrete logarithm can be computed as

$$\begin{aligned} DL_\alpha(\beta) &\equiv 2DL_\alpha(x^2+x+1) + DL_\alpha(x+1) - 2 \\ &\equiv 2 \cdot 26 + 6 - 2 \\ &\equiv 56 \pmod{63}. \end{aligned}$$

The implementation given by Weidemann [19, page 67] in $\mathbb{F}_{2^k}$ gives a complexity of

$$e^{\left(c_1 + o(1)(k \log_e k)^{1/2}\right)}$$

for the pre-computation and

$$e^{\left(1/(2c_1) + o(1)(k \log_e k)^{1/2}\right)}$$

for computing the logarithm, where $c_1 = \sqrt{2 \log_e 2}$. The implementer of this attack does not need to use the same representation as the user of the protocol. The alternatives would include the use of an isomorphic field for improved efficiency or the smallest field containing the Diffie-Hellman group.

A detailed discussion about the Index Calculus algorithm and related variants can be found in [19].

2.4 Security Improvements

In the previous section, four attacks were considered. From these attacks the following was concluded, where the Diffie-Hellman group $G = <\alpha>$:

- The size of $\sqrt{\text{ord}(\alpha)}$ must be large enough to prevent the Pollard ρ attack.
- The largest prime factor of $\text{ord}(\alpha)$ must be large enough to prevent the Pohlig-Hellman and Chinese Remainder Theorem attacks.
- The smallest field containing the Diffie-Hellman group must be large enough to prevent the Index Calculus attack.

All but the last constraint can easily be achieved by parameter selection. For the last constraint Lemma 2.13 (below) is needed. This lemma was first given in [11, Lemma 2.4] (slightly differently), and later corrected in [1].

Definition 2.12. [15, Theorem 2.44] Let K be a field with characteristic p and let n be a natural number not divisible by p and let η be a primitive n^{th} root of unity over K. Then the polynomial

$$\phi_n(x) = \prod_{\substack{1 \le s \le n \\ \gcd(s,n)=1}} (x - \eta^s)$$

is called the n^{th} *cyclotomic polynomial* over K.

Lemma 2.13. *Let q be a prime factor of $\phi_n(p)$ with n a natural number, p a prime number with $q \nmid n$, $p \nmid n$. Then the subgroup of $\mathbb{F}_{p^n}^*$ of order q is not contained in any proper subgroup of $\mathbb{F}_{p^n}^*$.*

Proof. Let α be an element in $\mathbb{F}_{p^n}^*$ of order q. It is needed to prove for $s|n$ and $s < n$ that $\text{ord}(<\alpha>) \nmid \text{ord}\left(\mathbb{F}_{p^s}^*\right)$. Note that $q \nmid \prod_{i|s} \phi_i(p)$ if and only if $\text{ord}(<\alpha>) \nmid \text{ord}\left(\mathbb{F}_{p^s}^*\right)$.

From $(x^n - 1, nx^{n-1}) = 1$ it follows that that $x^n - 1$ has no repeated root in the algebraic closure of $\mathbb{F}_q$. Thus ϕ_n and ϕ_s share no common root modulo q, since $x^n - 1 = \prod_{d|n} \phi_n(x)$. Now from $\phi_n(p) \equiv 0 \pmod{q}$ it follows that for any $s|n$, $\phi_s(p) \not\equiv 0 \pmod{q}$ thus $q \nmid \prod_{i|s} \phi_i(p)$. □

The above lemma gives the smallest field containing the Diffie-Hellman group; only the smallest field for a specific characteristic is given.

To determine a cyclic group G in $\mathbb{F}_{p^6}^*$ that is not contained in any subfield of $\mathbb{F}_{p^6}$, the cyclotomic polynomial must first be determined. From [15, Theorem 2.45] it follows that

$$\begin{aligned}\phi_6(x) &= \frac{x^6-1}{\prod_{\substack{0<s<6\\ s|6}} \phi_s(x)} \\ &= \frac{x^6-1}{(x-1)\phi_2(x)\phi_3(x)} \\ &= \frac{(x^6-1)}{(x-1)\left(\frac{(x^2-1)}{(x-1)}\right)\left(\frac{(x^3-1)}{(x-1)}\right)} \\ &= x^2-x+1.\end{aligned}$$

If q is any prime such that $q|(p^2-p+1)$, then q will be the order of the cyclic group G in $\mathbb{F}_{p^6}^*$. Let α be the generator of the group $\mathbb{F}_{p^6}^*$. Then the generator of the cyclic group G is $\alpha^{(p^6-1)/q}$, which generates the cyclotomic subgroup.

Definition 2.14. [1, Definition 1] In a field $\mathbb{F}_{p^k}$ we call a subgroup of prime order q with $q|\phi_k(p)$ and $q \nmid k$ a *cyclotomic subgroup* and denote it by $G_{q,p,k}$.

Example 2.15. Determine a cyclic subgroup G in $\mathbb{F}_{2^6}^*$ such that G is not contained in any subfield of $\mathbb{F}_{2^6}$, i.e. determine $G_{q,2,6}$. The 6^{th} cyclotomic polynomial is $\phi_6(x) = x^2-x+1$ and

$$\begin{aligned}\phi_6(2) &= 2^2-2+1 \\ &= 3.\end{aligned}$$

Since 3 is a prime number, the order of the group $G_{3,2,6}$ is 3.

Consider the field $\mathbb{F}_{p^6}$ defined by the primitive polynomial $f(x) = x^6+x+1 \in \mathbb{F}_2[x]$ Let α be a root of $f(x)$. The order of the group $<\alpha>$ is $63 = 3^2 \cdot 7$

and a generator of the group $G_{3,2,6}$ is

$$\begin{aligned}\alpha^{3\cdot 7} &= x^{21}\\ &= \alpha^6\alpha^6\alpha^6\alpha^3\\ &= (\alpha+1)(\alpha+1)(\alpha+1)\alpha^3\\ &= \alpha^2+\alpha+1.\end{aligned}$$

Chapter 3

The Compact Subgroup Protocol

A few variants of the Diffie-Hellman protocol are based on common ideas. These ideas are the use of a polynomial for representing the data and a linear shift register for the computations. The first example of a computational advantage gained by using shift registers, is given in [8]. The idea of reducing the amount of data that needs to be communicated through an irreducible polynomial was given in [2]. The Efficient and Compact Subgroup Trace Representation (XTR) has both the advantages of more efficient computations and less data that needs to be communicated and was designed by Lenstra in [13]. These examples are discussed in Chapter 4.

The general theory involving all of these aspects was suggested in [1]. The focus is on the use of a polynomial to reduce the number of bits to represent the key data. The use of a polynomial makes the use of a linear shift register attractive for computations. In this chapter a general theory is developed for a variant of the Diffie-Hellman Protocol. The variant which is called the *Compact Subgroup Protocol*, needs less data for key exchange than the Diffie-Hellman Protocol. Efficiency of the computations with the shift register is very dependent on the parameters of the protocol. The method given here shows that it is possible to perform the computations without sending the initial values to the other party.

In the next section, the Diffie-Hellman protocol is modified until the required variation is achieved. Section 3.2 shows the implementation advan-

tages of the variation. Some of the modifications change the security of the representation. The last section proves the Protocol secure.

3.1 Representations and the Protocol

The Compact Subgroup Protocol uses three different representations to achieve advantages in communication overhead and computation efficiency without sacrificing the security of the protocol. The three representations which are given here, each introduces only a small variation. The contributions of the representations are the following:

- The Multi Group Representation which uses more than one element in the Diffie-Hellman group to represent the user's key.
- The Polynomial Representation which fixes some parameters in the multi group representation.
- The Sequence Representation which introduces an alternative method of computations.

Multi Group representation

The Diffie-Hellman Protocol can be generalised to a representation which uses more than one Diffie-Hellman group $<\alpha>$. Instead of working in one group $<\alpha>$, the system is defined by a sequence of cyclic groups:

$$<\alpha^{h_0}>, <\alpha^{h_1}>, \ldots, <\alpha^{h_{k-1}}>,$$

where the h_i are integers in the interval $[1, \text{ord}(\alpha) - 1)$. The system parameters of the system are α, $h_0, \ldots, h_{k-1}$.

The private key of a user is an integer $priv \in [2, \text{ord}(\alpha) - 1)$ and the public key is the k-tuple

$$\left(\alpha^{h_0 \cdot priv}, \ldots, \alpha^{h_{k-1} \cdot priv}\right). \tag{3.1}$$

The shared secret generated by Alice and Bob will be

$$\left(\alpha^{h_0 \cdot priv_A \cdot priv_B}, \alpha^{h_1 \cdot priv_A \cdot priv_B}, \ldots, \alpha^{h_{k-1} \cdot priv_A \cdot priv_B}\right).$$

It is not important at this stage in which manner the shared secret is used. This representation is the Diffie-Hellman Protocol repeated k times, where the generator of the Diffie-Hellman group of each protocol run is different.

	Alice	Bob
Common	$<\alpha>$ and $h_0, \ldots, h_{k-1}$	
Private	$priv_A$	$priv_B$
Public	$\left(\alpha^{h_0 priv_A}, \ldots, \alpha^{h_{k-1} priv_A}\right)$	$\left(\alpha^{h_0 priv_B}, \ldots, \alpha^{h_{k-1} priv_B}\right)$
Shared Secret	$\left(\alpha^{h_0 priv_A priv_B}, \ldots, \alpha^{h_{k-1} priv_A priv_B}\right)$	

Table 3.1: Multi Group representation.

The use of the h_i's might result in a larger probability for a collision in the public keys, i.e.

$$\left\{\alpha^{h_0 priv_A}, \ldots, \alpha^{h_{k-1} priv_A}\right\} \bigcap \left\{\alpha^{h_0 priv_B}, \ldots, \alpha^{h_{k-1} priv_B}\right\} \neq \emptyset, \tag{3.2}$$

for private keys $priv_A$ and $priv_B$. This will be the case if there exist integers i and j such that

$$priv_A \cdot h_i \equiv priv_B \cdot h_j \pmod{\text{ord}(\alpha)}.$$

In the case of a collision, each of the participants in the protocol can compute the other participant's private key, since the above equation only has one unknown. In the next section, when the polynomial representation is defined, constraints will be given for the h_i's to provide results on the probability that public keys are disjoint.

Example 3.1. Let the Diffie-Hellman group be $<13>$ in $\mathbb{Z}_{31}$. The three groups used are generated by 13, 27 and 22 and correspond to $h_i = 1$, 3 and 7 respectively. Selecting the private key of Alice as 13, her public key is

$$\left(13^{13}, 27^{13}, 22^{13}\right) = (11, 29, 13) \pmod{31}$$

and using 17 as the private key of Bob, his corresponding public key is

$$\left(13^{17}, 27^{17}, 22^{17}\right) = (17, 15, 12) \pmod{31}.$$

Their shared secret is

$$\left(17^{13}, 15^{13}, 13^{13}\right) = (3, 27, 17) \pmod{31}.$$

Polynomial representation

A polynomial $f(x) \in \mathbb{F}_q[x]$ which is irreducible will be used to represent the generators of the Diffie-Hellman groups in the Multi Group representation. Let $f(x)$ be an irreducible polynomial of degree k that factors as

$$\begin{aligned} f(x) &= \prod_{i=0}^{k-1} (x - \alpha_i) \\ &= \prod_{i=0}^{k-1} \left(x - \alpha^{q^i}\right) \end{aligned}$$

where $\alpha_i = \alpha^{q^i}$ for $i = 0, 1, \ldots, k-1$ are in the splitting field of $f(x)$ and are of prime order. The roots of the polynomial $f(x)$ form the set of cyclic groups that are used in the Multi Group representation.

The private key is an integer $priv \in [2, \mathrm{ord}(\alpha))$ and the public key is

$$pub = f_{priv}(x) \tag{3.3}$$

$$= \prod_{i=0}^{k-1} (x - \alpha_i^{priv}), \tag{3.4}$$

where the private key must be such that the polynomial $f_{priv}(x)$ is irreducible. A sufficient condition for $f_{priv}(x)$ to be irreducible is that $\mathrm{ord}(\alpha)$ is prime. Indeed, since the order of α is prime, no power of α, except the unit, is contained in a subfield of $\mathbb{F}_{q^k}$ and therefore $f_{priv}(x)$ is irreducible.

The shared secret of users Alice and Bob is given by

$$f_{priv_A \cdot priv_B}(x) = \prod_{i=0}^{k-1} \left(x - \alpha_i^{priv_A \cdot priv_B}\right).$$

	Alice	Bob
Common	$f(x)$ over $\mathbb{F}_q$	
Private	$priv_A$	$priv_B$
Shared Secret	$f_{priv_A priv_B}$	

Table 3.2: Polynomial representation

In the Multi Group representation, a collision occurs if (3.2) is satisfied. A collision in the polynomial representation means that the public keys of Alice and Bob are the same, since a root of an irreducible polynomial defines the polynomial uniquely. Thus, a collision exists for the polynomial representation if there exist two integers i and j such that

$$\alpha_i^{priv_A} = \alpha_j^{priv_B},$$

that is

$$q^i \cdot priv_A \equiv q^j \cdot priv_B \pmod{\left(q^k - 1\right)}.$$

By noticing that

$$q^k \equiv 1 \pmod{\left(q^k - 1\right)}$$

it follows that

$$priv_A \equiv q^l \cdot priv_B \pmod{\left(q^k - 1\right)} \text{ and } \alpha_i^{priv_A} = \alpha_{i+l}^{priv_B}$$

for some integer l. Thus, a collision in the public keys of Alice and Bob occurs if and only if α_i^{priv} is a root of $f_{priv_B}(x)$. The number k of integers resulting in the same public key is therefore equal to the degree of the polyno-

mial $f(x)$. It follows that the number of distinct private keys is $\lfloor \text{ord}(\alpha)/k \rfloor$. Generally $k << \text{ord}(\alpha)$, resulting in a very small reduction in the key space. The probability that two randomly selected private keys are equivalent, is $k/\text{ord}(\alpha)$.

Example 3.2 (LUC)**.** Let α be the generator of a Diffie-Hellman group in $\mathbb{F}_{p^2}$. Let $\text{ord}(\alpha)$ be prime and $\text{ord}(\alpha)|\phi_2(p) = p + 1$. The constraint that $\text{ord}(\alpha)|\phi_2(p)$ together with Lemma 2.13 is used to ensure that the polynomial is irreducible. The use of the constraint is not necessary but it is convenient for implementation and illustrative purposes. This ensures that the minimal polynomial

$$f(x) = x^2 - (\alpha + \alpha^p)x + 1$$

of α is irreducible. The public key related to $priv$ is then

$$f_{priv}(x) = x^2 - \left(\alpha^{priv} + \alpha^{-priv}\right)x + 1,$$

since $p \equiv -1 \pmod{\text{ord}(\alpha)}$. This representation allows the representation of the public key to be represented by

$$pub = \alpha^{priv} + \alpha^{-priv}.$$

The shared secret generated by users Alice and Bob will be

$$f_{priv_A \cdot priv_B}(x) = x^2 - \left(\alpha^{priv_A \cdot priv_B} + \alpha^{-priv_A \cdot priv_B}\right)x + 1$$

or just

$$\alpha^{priv_A \cdot priv_B} + \alpha^{-priv_A \cdot priv_B}.$$

This illustrates that less data is needed to represent the public key, compared to the Diffie-Hellman protocol.

Sequence representation

Let $f(x)$ be an irreducible polynomial over $\mathbb{F}_q$. This polynomial defines a linear recurring relation that defines an associated linear sequence. Let such a sequence be denoted by (s_i), with initial values that are still to be defined. In particular if

$$f(x) = x^k - a_{k-1}x^{k-1} - a_{k-2}x^{k-2} - \cdots - a_0 \in \mathbb{F}_q[x]$$

then $f(x)$ defines the linear recurring relation

$$s_{n+k} = a_{k-1}s_{n+k-1} + a_{k-2}s_{n+k-2} + \cdots + a_0 s_n \text{ for } n \geq 0.$$

Theorem 3.3. [15, Theorem 6.21] *Let $s_0, s_1, \ldots$ be a k^{th}-order homogeneous linear recurring sequence in $\mathbb{F}_q$ with characteristic polynomial $f(x)$. If the roots $\alpha_0, \ldots, \alpha_{k-1}$ of $f(x)$ are distinct, then*

$$s_i = \sum_{j=0}^{k-1} \beta_j \alpha_j^i \text{ for } i = 0, 1, \ldots. \tag{3.5}$$

where $\beta_0, \ldots, \beta_{k-1}$ are elements that are uniquely determined by the initial values of the sequence and belong to the splitting field of $f(x)$ over $\mathbb{F}_q$.

Proof. The constants $\beta_0, \ldots, \beta_{k-1}$ are determined by the system of linear equations

$$\sum_{j=0}^{k-1} \alpha_i^n \beta_j = s_n, \text{ for } n = 0, 1, \ldots, k-1.$$

Since the determinant of this system is a Vandermonde determinant which is non-zero by the condition on $\alpha_0, \ldots, \alpha_{k-1}$, the elements $\beta_0, \ldots, \beta_{k-1}$ are uniquely determined and belong to the splitting field $\mathbb{F}_q(\alpha_0, \ldots, \alpha_{k-1})$ of $f(x)$ over $\mathbb{F}_q$, as seen from Cramer's rule. To prove the identity (3.5) for all $n \geq 0$ it suffices to check whether the elements in the right-hand side of (3.5), with these specific values of $\beta_0, \ldots, \beta_{k-1}$, satisfy the linear recurrence relation

(3.5). It now follows that

$$\begin{aligned}
&\sum_{j=0}^{k-1}\beta_j\alpha_j^{n+k} - a_{k-1}\sum_{j=0}^{k-1}\beta_j\alpha_j^{n+k-1} - \cdots - a_0\sum_{j=0}^{k-1}\beta_j\alpha_j^{n} \\
&\quad = \sum_{j=0}^{k-1}\beta_j\left(\alpha_j^{n+k} - a_{k-1}\alpha_j^{n+k-1} - \cdots - a_0\alpha_j^{n}\right) \\
&\quad = \sum_{j=0}^{k-1}\beta_j f(\alpha_j)\alpha_j^{n} \\
&\quad = 0
\end{aligned}$$

for all $n \geq 0$, and this completes the proof. □

Let $\alpha_0, \ldots, \alpha_{k-1}$ be the distinct roots of $f(x)$. By Theorem 3.3 any sequence with characteristic polynomial $f(x)$ can be written in terms of the roots α_i,

$$s_n = \sum_{j=0}^{k-1}\beta_j\alpha_j^n \text{ for } n = 0, 1, \ldots.$$

where the β_j are uniquely determined by the initial values. The sequence (s_i) with initial values

$$\begin{aligned}
s_0 &= k \\
s_1 &= \sum_{j=0}^{k-1}\alpha_j \\
s_2 &= \sum_{j=0}^{k-1}\alpha_j^2 \\
&\vdots \\
s_{k-1} &= \sum_{j=0}^{k-1}\alpha_j^{k-1}.
\end{aligned}$$

is called the characteristic sequence of $f(x)$. The public key $pub = (s_{priv,i})$ is

now defined by

$$\begin{aligned} s_{priv,i} &= s_{i\cdot priv} \\ &= \sum_{j=0}^{k-1} \alpha_j^{i\cdot priv} \\ &= \sum_{j=0}^{k-1} \left(\alpha_j^{priv}\right)^i \text{ for } i = 0, 1, \ldots. \end{aligned}$$

which is every $priv^{th}$ element of the sequence (s_i). Since the polynomial $f_{priv} = \prod_{i=0}^{k-1} \left(x - \alpha_i^{priv}\right)$ is irreducible, this polynomial is the characteristic polynomial of the sequence $(s_{priv,i})$, see Theorem 3.3.

The shared secret of users Alice and Bob is now given by the characteristic sequence of $f_{priv_A \cdot priv_B}(x)$, which is every ${priv_A priv_B}^{th}$ term of the sequence (s_i). The shared secret that is computed by each user in the system is the same as seen from

$$\begin{aligned} s_{priv_B, i\cdot priv_A} &= \sum_{j=0}^{k-1} \left(\alpha_j^{priv_B}\right)^{i\cdot priv_A} \\ &= \sum_{j=0}^{k-1} \left(\alpha_j^{priv_A}\right)^{i\cdot priv_B} \\ &= s_{priv_A, i\cdot priv_B}. \end{aligned}$$

No attention has been given to the initial values of the public sequences of each user. The initial values for each user's sequence must be such that the β_j's in (3.5) have the value 1, otherwise the shared sequence will not be the same. The initial values need to be sent along with the linear relation.

For user Alice, the initial values are

$$\begin{aligned} s_{priv_A,0} &= k \\ s_{priv_A,1} &= \sum_{j=0}^{k-1} \alpha_j^{priv_A} \\ s_{priv_A,2} &= \sum_{j=0}^{k-1} \alpha_j^{2priv_A} \\ &\vdots \\ s_{priv_A,k-1} &= \sum_{j=0}^{k-1} \alpha_j^{(k-1)priv_A}, \end{aligned}$$

which are sent to user Bob along with the public key.

Example 3.4 (LUC)**.** Continuing with Example 3.2 the initial values of the sequence are given by

$$\begin{aligned} s_0 &= 2 \\ s_1 &= \alpha + \alpha^{-1} \end{aligned}$$

and the linear relation is

$$s_{n+2} = \left(\alpha + \alpha^{-1}\right) s_{n+1} - s_n.$$

User Alice uses her private key $priv_A$ and computes the initial values

$$\begin{aligned} s_{priv_A,0} &= 2 \\ s_{priv_A,1} &= \alpha^{priv_A} + \alpha^{-priv_A} \end{aligned}$$

and the linear relation

$$s_{n+2} = \left(\alpha^{priv_A} + \alpha^{-priv_A}\right) s_{n+1} - s_n.$$

The shared secret that Bob computes is every ${priv_B}^{th}$ element in Alice's

public sequence, i.e.

$$s_{priv_A,i} = \left(\alpha^{priv_A} + \alpha^{-priv_A}\right) s_{i \cdot priv_A - 1} - s_{i \cdot priv_A - 2} \tag{3.6}$$

with initial values

$$\begin{aligned} s_{priv_A,0} &= 2 \\ s_{priv_A,1} &= \alpha^{priv_A} + \alpha^{-priv_A}. \end{aligned}$$

Compact Subgroup Protocol

The three representations given in the previous three sections are now used to construct the Compact Subgroup Protocol. Each of the three representations provides an advantage:

- Multi Group representation gives a basis for security.
- Polynomial representation enables the use of a compact representation.
- Sequence representation provides a method for efficient computation.

As in the sequence representation let

$$f(x) = \prod_{i=0}^{k-1} (x - \alpha_i) \in \mathbb{F}_{p^t}[x]$$

be an irreducible system polynomial over $\mathbb{F}_{p^t}$ of degree k and of prime order q dividing $\phi_{tk}(p)$. The private key of a user is an integer $priv \in [2, q)$ and the associated public key is

$$f_{priv}(x) = \prod_{i=0}^{k-1} \left(x - \alpha_i^{priv}\right). \tag{3.7}$$

Since the polynomial f is irreducible over $\mathbb{F}_{p^t}$ and of prime order the polynomial f_{priv} is also irreducible over $\mathbb{F}_{p^t}$ for $priv \in [1, q)$ and it follows that the orders of f and f_{priv} are equal. The orders of the system and public polynomials are equal and therefore the roots of these polynomials define

the same Diffie-Hellman group, up to isomorphism. Thus the only difference between the system and public polynomials is how the polynomial is used. All the results that are applicable to the system polynomial are therefore also applicable to the public polynomials.

For users Alice and Bob the respective private keys are $priv_A$ and $priv_B$, where the associated shared secret is the coefficient of x^{k-1} in $f_{priv_A priv_B}(x)$, which is $\sum_{i=0}^{k-1} \alpha_i^{priv_A \cdot priv_B}$. The shared secret and the respective public keys are computed by the combining function F. An illustration of the protocol is given in Figure 3.1.

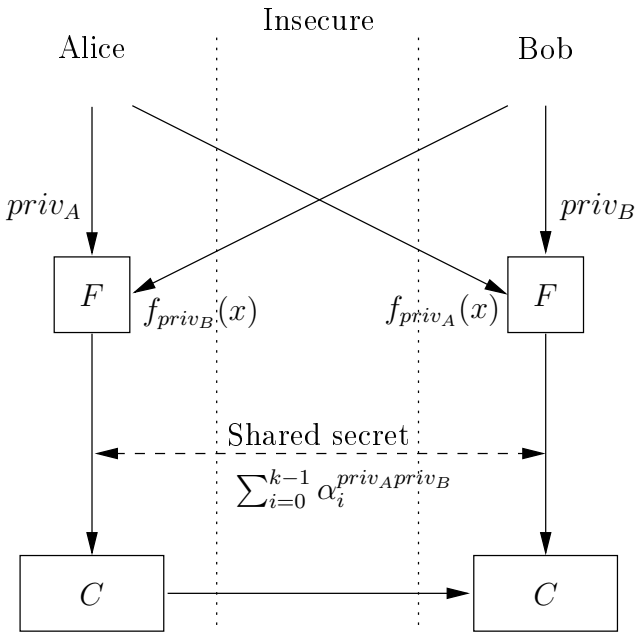

Figure 3.1: Key exchange using the Compact Subgroup Protocol

In this protocol, the Berlekamp-Massey algorithm is used to compute the combining function F. The polynomial $f(x)$ is used to construct a linear relation and initial values are chosen arbitrarily to form a sequence (s_i). From this sequence every $priv^{th}$ element is selected to create a new sequence, $(s_{priv,i})$, with characteristic polynomial $f_{priv}(x)$. The Berlekamp-Massey algorithm produces the minimal polynomial of $(s_{priv,i})$ which must be $f_{priv}(x)$, since $f_{priv}(x)$ is irreducible. It is important to note that in this protocol the

computations are not performed in the Diffie-Hellman group, but in the field over which the polynomial is defined. This gives the advantage that the field operations used, require less resources.

Security of the Compact Subgroup Protocol will follow from Theorem 3.12. Thus all the same measures needed to secure the Diffie-Hellman protocol are also needed for the Compact Subgroup Protocol. These measures are contained in selecting a polynomial of large prime order dividing $\phi_{kt}(p)$.

Reduction in the data that needs to be transmitted is due to the property that $\text{ord}(f)|\phi_{kt}(p)$, see Theorem 3.9.

	Alice	Bob
Common	$f(x)$	
Private	$priv_A$	$priv_B$
Public	$f_{priv_A}(x)$	$f_{priv_B}(x)$
Shared Secret	$\sum_{i=0}^{k-1} \alpha_i^{priv_A priv_B}$	

Table 3.3: Compact Subgroup Protocol.

More attention is given to the computations and security properties of the protocol in Section 3.2.

Example 3.5. Let the system polynomial be the irreducible polynomial $f(x) = x^3 + 2x + 2 \in \mathbb{F}_3[x]$. The order of $f(x)$ is 13, which is the same as $\phi_3(3)$. With initial values $s_0 = 1$, $s_1 = 1$ and $s_2 = 1$ the linear relation

$$s_n = s_{n-2} + s_{n-3}$$

generates the sequence

$$(s_i)_{i=0}^{\infty} = 1112201210010111220121001011122 01 \ldots \ .$$

Let the private keys of users Alice and Bob be 2 and 4 respectively. The public sequence of Alice is then

$$(s_{2i})_{i=0}^{\infty} = 11211001202011121 \ldots$$

with characteristic polynomial

$$\begin{aligned} f_{priv_A}(x) &= f_2(x) \\ &= x^3 + x^2 + x + 2 \end{aligned}$$

determined by the Berlekamp-Massey algorithm.

Similarly, the public sequence of Bob is

$$(s_{4i})_{i=0}^{\infty} = 1210221110100121\ldots$$

with characteristic polynomial

$$\begin{aligned} f_{priv_B}(x) &= f_4(x) \\ &= x^3 + x^2 + 2. \end{aligned}$$

Bob now uses $f_2(x)$ and the initial values $s'_0 = 0$, $s'_1 = 1$ and $s'_2 = 0$, for example, to generate a shift register sequence of (s_{2i}), namely

$$(s'_{2i})_{i=0}^{\infty} = 010222122002101\ldots$$

Taking every 4^{th} element of the above sequence gives

$$(s'_{4\cdot 2i})_{i=0}^{\infty} = 022122201012\ldots$$

with characteristic polynomial

$$\begin{aligned} f_{priv_A priv_B}(x) &= f_{4\cdot 2}(x) \\ &= x^3 + 2x^2 + 2x + 2. \end{aligned}$$

Likewise Alice uses $f_4(x)$ and the initial values $s''_0 = 1$, $s''_1 = 0$ and $s''_2 = 1$, say, to generate a shift register sequence of (s_{4i}), namely

$$(s''_{4i})_{i=0}^{\infty} = 1010012102211101\ldots$$

Taking every second element of the above sequence gives

$$(s''_{2\cdot 4i})_{i=0}^{\infty} = 110202100112111\ldots$$

with characteristic polynomial of

$$\begin{aligned} f_{priv_A priv_B}(x) &= f_{2\cdot 4}(x) \\ &= x^3 + 2x^2 + 2x + 2. \end{aligned}$$

The shared secret is the coefficient of x^2, namely 2.

3.2 Implementation Advantages

The Compact Subgroup Protocol is much more complex than the Diffie-Hellman Protocol. This section will show its advantages: less data is needed to represent the keys and a more efficient computation method.

Compact Representation

In the Compact Subgroup Protocol the coefficients of the polynomials are transmitted. This is in contrast with the Diffie-Hellman protocol where a root of the polynomial is transmitted. If the polynomial is defined over $\mathbb{F}_{p^t}$ and of degree k then $kt\log_2(p)$ bits are needed to represent the polynomial which is the same number of bits needed in the Diffie-Hellman protocol. In Theorem 3.9 it is proved that it is not required to send all the polynomial's coefficients.

The reduction of the number of coefficients that define a polynomial is due to the fact that the roots are in the cyclotomic subgroup of $\mathbb{F}^*_{p^{tk}}$, see Definition 2.14.

Definition 3.6. Let n be a positive integer. For every integer $k \leq n$, the *elementary symmetric polynomial* of degree k in $R[X_1, \ldots, X_n]$ where R is a

ring with unit, which is denoted by σ_k, is given by the formula

$$\sigma_k = \sum_{\substack{H \subset \{1,\ldots,n\} \\ |H|=k}} \left(\prod_{i \in H} X_i \right)$$

Example 3.7. The elementary symmetric polynomials for $n = 1, 2, 3, 4$ and 5 are.

$\sigma_1 : x_1 + x_2 + x_3 + x_4 + x_5$

$\sigma_2 : x_1x_2 + x_1x_3 + x_1x_4 + x_1x_5 + x_2x_3 + x_2x_4 + x_2x_5 + x_3x_4 + x_4x_5$

$\sigma_3 : x_1x_2x_3 + x_1x_2x_4 + x_1x_2x_5 + x_2x_3x_4 + x_2x_3x_5 + x_3x_4x_5$

$\sigma_4 : x_1x_2x_3x_4 + x_1x_2x_3x_5 + x_2x_3x_4x_5$

$\sigma_5 : x_1x_2x_3x_4x_5.$

In the proof of Theorem 3.9, the following property of symmetric polynomials, is needed.

Proposition 3.8. [6, Proposition 3.2.2] *If $f(x) = \sum_{k=0}^{n} a_k x^k$ is a monic polynomial of degree n with roots $x_1, \ldots, x_n$ belonging to a field $\mathbb{F}$, then*

$$a_{n-k} = (-1)^k \sigma_k(x_1, \ldots, x_n),$$

for $0 \leq k \leq n$, where $\sigma_k(x_1, \ldots, x_n)$ is the k^{th} symmetric polynomial.

Theorem 3.9. [1, Theorem 1] *Let α be a generator of a cyclotomic subgroup $G_{p,q,k}$, where p is odd and $k \geq 2$. Let $x^{k/d} + a_{k/d-1}x^{k/d-1} + \cdots + a_1x + a_0$ be the minimal polynomial of α over $\mathbb{F}_{p^d}$, for some d dividing k, with $d < k$. Then $a_0 = (-1)^{k/d}$ and if $k = 2l$ for l an integer, $a_i = (-1)^{k/d} a_{k/d-i}^{p^l}$ for $i = 1, \ldots, k/d - 1$.*

Proof. Write $\alpha_j = \alpha^{p^{d_j}}$ for $j = 0, \ldots, k/d - 1$. Then

$$x^{k/d} + a_{k/d-1}x^{k/d-1} + \cdots + a_1x + a_0 = \prod_{j=0}^{k/d-1} (x - \alpha_j),$$

and using Proposition 3.8 it follows that $a_i = (-1)^{k/d-i}\sigma_{k/d-i}(\alpha_0, \ldots, \alpha_{k/d-1})$ for $i = 0, \ldots, k/d-1$, where $\sigma_n(\alpha_0, \ldots, \alpha_{k/d-1})$ is the n^{th} elementary symmetric polynomial in the conjugates α_j of α. In particular

$$\begin{aligned} a_0 &= (-1)^{k/d}\sigma_{k/d}(\alpha_0, \ldots, \alpha_{k/d-1}) \\ &= (-1)^{k/d}\alpha_0 \cdots h_{k/d-1} \\ &= (-1)^{k/d}\alpha^{1+p^d+p^{2d}+\cdots+p^{k-d}}. \end{aligned}$$

But $1 + p^d + p^{2d} + \cdots + p^{k-d} = (p^k - 1)/(p^d - 1)$, which is divisible by $\phi_k(p)$ and hence by q. Therefore $a_0 = (-1)^{k/d}$.

If $k = 2l$ then $p^k - 1 = (p^l - 1)(p^l + 1)$. Since the order q of h divides $p^k - 1$ but not $p^l - 1$ it follows that $p^l \equiv -1 \pmod{q}$ and therefore $\alpha_j^{-1} = \alpha_i^{p^l}$ for $j = 0, \ldots, k/d-1$. Since $\alpha_0 \cdot h_1 \cdots \alpha_{k/d-1} = 1$ it follows that $\sigma_{k/d-i}(\alpha_0, \ldots, \alpha_{k/d-1}) = \sigma_i\left(\alpha_0^{-1}, \ldots, \alpha_{k/d-1}^{-1}\right)$ and

$$\begin{aligned} \sigma_{k/d-i}(\alpha_0, \ldots, \alpha_{k/d-1}) &= \sigma_i\left(\alpha_0^{-1}, \ldots \alpha_{k/d-1}^{-1}\right) \\ &= \sigma_i\left(\alpha_0^{p^l}, \ldots, \alpha_{k/d-1}^{p^l}\right) \\ &= \sigma_i(\alpha_0, \ldots, \alpha_{k/d-1})^{p^l} \\ &= \left((-1)^i a_{k/d-i}\right)^{p^l} \\ &= (-1)^i a_{k/d-i}^{p^l}. \end{aligned}$$

for $i = 1, \ldots, k/d-1$. Thus

$$\begin{aligned} a_i &= (-1)^{k/d-i}\sigma_{k/d-i}(\alpha_0, \ldots, \alpha_{k/d-1}) \\ &= (-1)^{k/d}a_{k/d-i}^{p^l}. \end{aligned}$$

□

From the above theorem it is seen that both the degree of the polynomial and the field over which the polynomial is defined, determine the obtainable communication reduction. By selecting the degree of the polynomial correctly a saving of 75% is achievable in the communication of the public keys.

Proposition 3.10. [1, Proposition 1] *Let $e = kt$, with $0 < k$. Then for any element h of $G_{q,p,e}$ the minimal polynomial of h over $\mathbb{F}_{p^t}$ can be represented by using the following number of elements of $\mathbb{F}_{p^t}$*

(1) $k-1$ *if kt is odd,*

(2) $\dfrac{k-1}{2}$ *if t is even and k is odd,*

(3) $\dfrac{k}{2}$ *if k is even.*

Proof. Represent the elements of $G_{q,p,e}$ by their minimal polynomials over the subfield of degree t. The constant coefficients are $(-1)^k$, so $k-1$ elements of $\mathbb{F}_{p^t}$ are sufficient to represent elements of $G_{q,p,e}$. This covers the first case.

In the second and third cases e is even and by Theorem 3.9 only half of the coefficients are required. More precisely, if k is odd only $(k-1)/2$ coefficients are required and if k is even, only $k/2$ coefficients are required. Bearing in mind that the first coefficient is $(-1)^k$, the results follow. □

It now follows from Proposition 3.10 that the fraction of coefficients saved is given by

$$\frac{k - \text{number of coefficients needed}}{k}.$$

The coefficients saved in each case, respectively, are

$$\begin{aligned} \frac{k-(k-1)}{k} &= \frac{1}{k}, \\ \frac{k-(k-1)/2}{k} &= \frac{1}{2} + \frac{1}{2k} \quad \text{and} \\ \frac{k-(k/2)}{k} &= \frac{1}{2}. \end{aligned}$$

With only compression in mind, the field used in the protocol should have even extension degree and the degree of the polynomial should be odd.

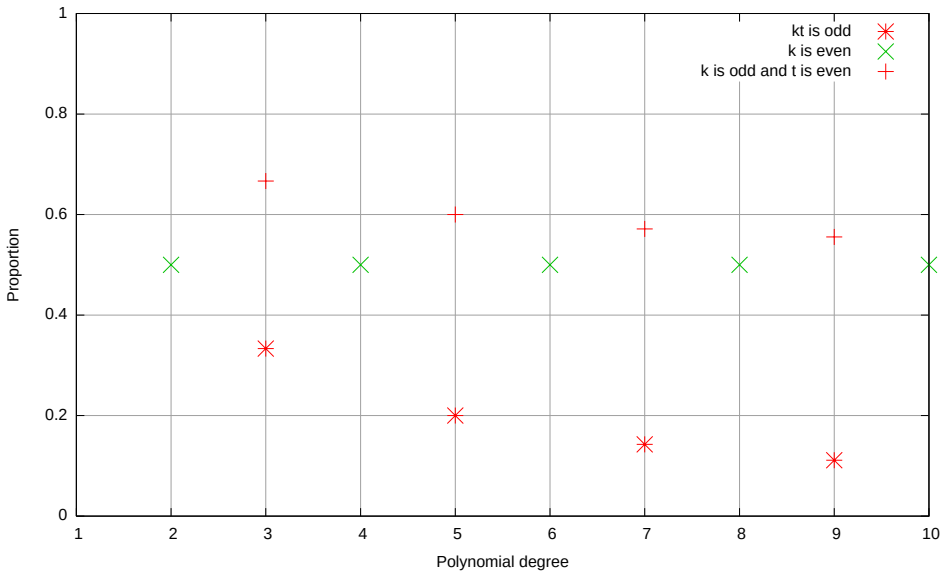

Figure 3.2: Proportion of bits saved in the Compact Subgourp Protocol.

The number of bits needed for representing the polynomials in the Compact Subgroup Protocol is either $(k-1)\log_2 p^t$, $(k-1)/2\log_2 p^t$ or $k/2\log_2 p^t$.

Computation Efficiency

The combination function is used to compute the $priv^{th}$ term in the sequence (s_i). Computation of such a sequence would require initial values. These initial values must generate a sequence of the form

$$s_i = \sum_{j=0}^{k-1} \alpha_j^i.$$

The most direct way to obtain the initial values is to factorise the polynomial, which requires field operations in $\mathbb{F}_{p^{kt}}$. An alternative method to compute the minimal polynomial of $(s_{priv,i})$ is the Berlekamp-Massey Algorithm.

Alice will receive the public key, pub_B, of Bob. From pub_B a subsequence $(s_{pub_B,i})$ of (s_i) is created, where the initial values are arbitrary. Every ${priv_A}^{th}$ term of this sequence will create the shared sequence. This shared sequence is used in the Berlekamp Massey algorithm. The output of the Berlekamp

Massey Algorithm is the shared polynomial, from which the shared secret can be obtained.To generate the shared sequence, $2k \cdot priv_A \leq 2k \cdot \text{ord}(\alpha)$ values of the initial sequence are needed. Since the Berlekamp Massey algorithm provides the result in at most k^2 steps, the combining function would require at most $k^2 + 2 \cdot \text{ord}(\alpha) = O(\text{ord}(\alpha))$ operations[1]. The computational complexity is not feasible. In Section 4.2 and 4.3 an improved sequence computation method is given that will make implementation of the combining function computationally feasible.

3.3 Security

The security of the Compact Subgroup Protocol is based on the Diffie-Hellman protocol. A modification of the Discrete Logarithm Problem is needed to prove the security of the Compact Subgroup Protocol.

Let

$$f(x) = x^k - a_{k-1}x^{k-1} - \cdots - a_0 \in \mathbb{F}_{p^t}[x]$$

be an irreducible polynomial, with $\alpha^{priv} \in \mathbb{F}_{p^{kt}}$ as a root. The problem is to determine *priv* if $a_0, \ldots, a_{k-1}$ and α are given. If it is possible to calculate a solution, the protocol is considered compromised.

In Definition 3.11, *summing functions* are defined. The summing functions give a more general form for writing the coefficients of a polynomial. The security proof is given in terms of summing functions.

Definition 3.11. Let n be a non-negative number, consider the integers $e_0, \ldots, e_{n-1}$ and the elements $\lambda_0, \ldots, \lambda_{n-1} \in \mathbb{F}_{p^t} \setminus \{0\}$. Then the *summing function* $Z : <\alpha> \rightarrow \mathbb{F}_{p^t}$ is given by:

$$Z(\kappa) = \sum_{i=0}^{n-1} \lambda_i \cdot \kappa^{e_i}, \text{ for } \kappa \in <\alpha>.$$

[1]The complexity is same as solving the Discrete Logarithm Problem

The number n is called the *degree* of the summing function and the number $d = \gcd(e_0, e_1, \ldots, e_{n-1}, \text{ord}(\alpha))$ is called the order of the summing function.

The coefficients of the polynomial F can be written as

$$\begin{aligned} Z_i(\alpha) &= \sum_{\substack{H \subset \{0,1,\ldots,n-1\} \\ |H| = n-i}} \left(\prod_{j \in H} \alpha_j \right) \\ &= \sum_{\substack{H \subset \{0,..n-1\} \\ |H| = n-i}} \alpha^{\sum_{j \in H} p^{tj}}, \end{aligned}$$

where $\alpha_i = \alpha^{p^{it}}$. Thus the coefficients of a polynomial can be written in terms of summing functions. With the exception of Z_0 and Z_n the order of the summing functions is 1. Notice that $e_i = \sum_{i \in H} p^{ti}$, $\text{ord}(\alpha) | (p^{tn} - 1)$ and also $\left(\sum_{i \in H} p^{ti}, p^{tn} - 1 \right) = 1$.

Theorem 3.12 (Polynomial Security Proof)**.** [2, Theorem 3.2] *Let Z be a summing function of order d. Also let O be an oracle that on basis of any γ^x and γ^y computes $Z(\gamma^{xy})$. Then there exists a polynomial time algorithm that computes γ^{yxd} on basis of γ^x and γ^y. Thus for $d = 1$ there exists a polynomial time algorithm that solves the Diffie-Hellman problem in $<\gamma>$.*

Proof. Let $V = \gamma^x$ and $W = \gamma^y$ be any elements in $<\gamma>$. Using the oracle O the value $Z\left(\gamma^{x(y+1)}\right)$ can be determined using V and $\gamma^i W$ as input. The following system of equations can be constructed

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ V^{e_1} & V^{e_2} & \cdots & V^{e_n} \\ \vdots & \vdots & \ddots & \vdots \\ V^{(n-1)e_1} & V^{(n-1)e_2} & \cdots & V^{(n-1)e_n} \end{bmatrix} \begin{bmatrix} \lambda_1 \gamma^{xye_1} \\ \lambda_2 \gamma^{xye_2} \\ \vdots \\ \lambda_n \gamma^{xye_n} \end{bmatrix} = \begin{bmatrix} Z(\gamma^{xy}) \\ Z\left(\gamma^{x(y+1)}\right) \\ \vdots \\ Z\left(\gamma^{x(y+n-1)}\right) \end{bmatrix}$$

The above matrix is a Vandermonde matrix.

First consider the case that $V^{e_1}, V^{e_2}, \ldots, V^{e_n}$ are distinct. Then the matrix is invertible and the elements $\gamma^{xye_1}, \gamma^{xye_2}, \ldots, \gamma^{xye_n}$ can be determined by the system of equations. By taking a suitable combination of these elements the element γ^{xyd} can be computed.

If there exist integers i and j such that $V^{e_i} = V^{e_j}$, then the two columns containing V^{e_i} and V^{e_j} are identical and $\gamma^{xye_i} = \gamma^{xye_j}$. By removing duplicate columns and γ values, the matrix is invertible and the values of γ^{xyd} can be computed. □

The above security proof is not directly applicable to the Compact Subgroup protocol. In the protocol a polynomial is sent as the public key, instead of the roots of the polynomial. When a polynomial is sent, the Scipione del Ferro root finding technique can be used to reduce the security problem in polynomial time to the one state in Theorem 3.12. Therefore the Compact Subgroup Protocol is at least as secure as the Diffie-Hellman protocol.

Chapter 4

Examples of the Compact Subgroup Protocol

In Chapter 3 a motivation is given for the use of polynomials and their characteristic sequences in key agreement systems. In this chapter three such existing systems are discussed.

4.1 Doing More with Fewer Bits

In [2] Brouwer et. al. introduced a key distribution system that uses a 6^{th} degree polynomial and which is an example of the Compact Subgroup Protocol. This system is referred to here as *DMFB*. In the literature, this protocol introduced the idea of using a polynomial to reduce the data that needs to be transmitted during the execution of the protocol. It will follow that the 6^{th} degree polynomial used in DMFB is determined by only two coefficients.

4.1.1 Description

The system is defined by a 6^{th} degree polynomial $f(x) \in \mathbb{F}_p[x]$,

$$f(x) = x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + 1$$

with p a prime number. The polynomial has roots $\alpha_0, \ldots, \alpha_5$ of prime order q dividing $\phi_6(p) = p^2 - p + 1$. Thus the Diffie-Hellman group is the cyclic group generated by any root α_i. The private key is an integer $priv \in [2, \text{ord}(\alpha_0))$ and the associated public key is $f_{priv}(x)$.

Since $f(x)$ satisfies the condition of Theorem 3.9 it follows that the leading and constant coefficients of the polynomials f and f_{priv} are 1. Furthermore three of the remaining five coefficients determine the polynomial uniquely, namely $a_5 = a_1^{p^3}$, $a_4 = a_2^{p^3}$ and a_3. However, the coefficient a_3 can be written in terms of a_1 and a_2: Let $\alpha_1, \ldots \alpha_6$ be the roots of $f(x)$. From Theorem 4.1 it follows that the first and second terms in the linear sequence are given by

$$\begin{aligned} s_1 &= \sigma_1(\alpha_1, \ldots, \alpha_6) \\ s_2 &= \alpha_1 s_1 - 2\sigma_2(\alpha_1, \ldots, \alpha_6) \end{aligned}$$

where σ_i is the i^{th} elementary symmetrical polynomial over $\mathbb{F}_p[x_1, \ldots x_6]$. Then by elementary computation it follows that

$$a_3 = -2\sum_{i=0}^{5} \alpha_i - \sum_{i=0}^{5} \alpha_i^2 - 2,$$

from which follows that

$$a_3 = -2 + 2\alpha_1 - \alpha_1^2 + 2\alpha_2.$$

Theorem 4.1 (Newton's Formula)**.** [15, Theorem 1.75] *Let $\sigma_1, \ldots, \sigma_n$ be the elementary symmetric polynomials in $x_1, \ldots, x_n$ over a ring R, and let $s_0 = n \in \mathbb{Z}$ and $s_k = s_k(x_1, \ldots, x_n) = x_1^k + \cdots + x_n^k \in R[x_1, \ldots, x_n]$ for $k \geq 1$. Then the formula*

$$s_k - s_{k-1}\sigma_1 + s_{k-2}\sigma_2 + \cdots + (-1)^{m-1} s_{k-m+1}\sigma_{m-1} + (-1)^m \frac{m}{n} s_{k-m} = 0$$

holds for $k \geq 1$, where $m = \min\{k, n\}$.

Using the Newton's Formula and noting that $s_1 = \sigma_1(\alpha_1, \ldots, \alpha_6) = a_1$

and $\sigma_2(\alpha_1, \ldots, \alpha_6) = a_2$, it follows that

$$\begin{aligned} a_3 &= -2a_1 - s_2 - 2 \\ &= 2a_1 - (s_1\sigma_1(\alpha_1, \ldots, \alpha_6) - 2\sigma_2(\alpha_1, \ldots, \alpha_6)) - 2 \\ &= -2 + 2a_1 - a_1^2 + 2a_2. \end{aligned}$$

Thus $f(x)$ is determined by a_1 and a_2. Consequently, as in the Compact Subgroup Protocol the public keys

$$f_{priv}(x) = \prod_{i=0}^{5} \left(x - \alpha_i^{priv}\right)$$

can be represented by only two of its coefficient, see (3.7).

4.1.2 Computations

Computation in this protocol does not use the sequence representation. All the computations are done in the field $\mathbb{F}_{p^6}$ and the roots of the polynomial must be computed. Thus, the performance of the protocol is not very good and no complexity analysis is performed.

The system parameter is generated by Algorithm 4. The algorithm selects an element $\alpha \in \mathbb{F}_{p^6}^*$ of order q, where q is the order of the Diffie-Hellman group. The system polynomial is the minimal polynomial of α, over the prime field. Only two of the coefficients are used to represent the system polynomial.

Algorithm 4: Parameter generation of DMFB.

Data: n_p and n_q, the required bit lengths of primes p and q respectively.
Output: primes p and q and the coefficient pair (a_1, a_2)

1 Choose $p \in \{n \in \mathbb{Z}_{2^{n_p}} \setminus \mathbb{Z}_{2^{n_p-1}} : n \text{ is prime}\}$;
2 Choose $q \in \{i \in \mathbb{Z}_{2^{n_q}} \setminus \mathbb{Z}_{2^{n_q-1}} : i \text{ divides } \phi_6(p) \text{ and is prime}\}$;
3 Choose $\alpha \in \mathbb{F}_{p^6}$ such that $\text{ord}(\alpha) = q$;
4 $f(x) \leftarrow$ the minimal polynomial of α over $\mathbb{F}_p$;
5 $(a_1, a_2) \leftarrow$ the coefficients of x and x^2 in $f(x)$;

Each user of the system uses his private key, $priv$, to compute the $priv^{th}$ power of the roots of the system polynomial. These new elements form the roots of an irreducible polynomial $f_{priv}(x)$ over $\mathbb{F}_p$ of which only two coefficients are needed to represent the public key. Algorithm 5 computes the public and private keys.

Algorithm 5: Selection of private and public keys in DMFB.

Data: primes p and q, the field $\mathbb{F}_{p^6}$ and coefficient pair (a_1, a_2)
Output: $priv$ and pub the private and public key respectively

1 $f(x) \leftarrow 1 + a_1 x + a_2 x^2$;
2 $f(x) \leftarrow f(x) + (-2 + 2a_1 - a_1^2 + 6a_2)x^3$;
3 $f(x) \leftarrow f(x) + a_2^{p^3} x^4 + a_1^{p^3} x^5 + x^6$;
4 Choose $\alpha \in \mathbb{F}_{p^6}$ such that $f(\alpha) = 0$;
5 $priv \in_R [2, q-1)$.;
6 $f_{priv} \leftarrow$ the minimal polynomial of α^{priv} over $\mathbb{F}_p$;
7 $pub \leftarrow$ the coefficients of x and x^2 in $f_{priv}(x)$;

Computation of the shared secret is similar to the computation of a user's public key, the only difference being the input parameters of the algorithm. Algorithm 6 is used to compute the shared secret.

Algorithm 6: Combining function for DMFB

Data: the field $\mathbb{F}_{p^6}$, a public key pub and the private key $priv$
Output: The shared secret s.

1 $(a_1, a_2) \leftarrow pub$;
2 $f(x) \leftarrow 1 + a_1 x + a_2 x^2$;
3 $f(x) \leftarrow f(x) + (-2 + 2a_1 - a_1^2 + 6a_2)x^3$;
4 $f(x) \leftarrow f(x) + a_2^{p^3} x^4 + a_1^{p^3} x^5 + x^6$;
5 $\alpha \in \mathbb{F}_{p^6}$ such that $f_{priv}(\alpha) = 0$;
6 $g(x) \leftarrow$ the minimal polynomial of α^{priv} over $\mathbb{F}_p$;
7 $s \leftarrow$ the coefficient of x in $g(x)$;

4.1.3 Security

For the security proof of the DMFB protocol, Theorem 3.12 can be applied directly. Is is worthwhile to note that in DMFB there exists a dependence

between the coefficients of the polynomial, that is not given in Theorem 3.9. This indicates that only one of the two coefficients should be used as the shared secret, as is the case in DMFB.

Example 4.2 (DMFB)**.** Let the characteristic of the field be 113, then $\phi_6(113) = 3 \cdot 4219$. Select $q = 4219$ as the order of the Diffie-Hellman group in $\mathbb{F}^*_{113^6}$ and let $\mathbb{F}_{113^6}$ be defined by the irreducible polynomial $x^6 + 65x^5 + 17x^4 + 84x^3 + 25x^2 + 67x + 98 \in \mathbb{F}_{113}[x]$. Implementation of Algorithm 4 gives the system polynomial

$$f(x) = x^6 + 37x^5 + 63x^4 + 41x^3 + 63x^2 + 37x + 1$$

over $\mathbb{F}_{113}$ which is represented by $(37, 63)$.

An implementation of Algorithm 5 is used to compute both the public and private key pairs. Let the private key of Alice be 944. The root obtained by factorising $f(x)$ is $98x^5 + 94x^4 + 90x^3 + 91x^2 + 24x + 111$ and its 944^{th} power is $41x^5 + 11x^4 + 21x^3 + 58x^2 + 6x + 47$ giving a root of the public polynomial $x^6 + 81x^5 + 16x^4 + 72x^3 + 16x^2 + 81x + 1$ that is represented by $(81, 16)$.

Likewise, let 1850 be the private key of Bob. The root of $f(x)$ used by Bob in its computations is $79x^5 + 20x^4 + 23x^3 + 63x^2 + 45x + 43$ and its 1850^{th} power is $50x^5 + 24x^4 + 43x^3 + 39x^2 + 55x + 64$ with minimal polynomial $x^6 + 33x^5 + 78x^4 + 35x^3 + 78x^2 + 33x + 1$, represented by $(33, 78)$.

Algorithm 6 is used to find the shared secret. In the computations of the shared secret, Alice uses $priv_A = 944$ and $pub_B = (33, 78)$ where user Bob uses $priv_B = 1850$ and $pub_A = (81, 16)$ to compute the shared secret 99.

4.2 Cubic Field Extension

In [8] a public key cryptographic system was introduced with a polynomial of degree three where the order of the roots are not a prime number. The system is abbreviated here as CFE. In the literature, the main contribution of the article is the introduction of a linear shift register in the computation

of the shared secret. Sequence representation is used for the computations of the protocol.

4.2.1 Description

The system is defined by an irreducible polynomial

$$f(x) = x^3 - ax^2 + bx - 1, \quad a, b \in \mathbb{F}_p \tag{4.1}$$

with p a prime. The polynomial $f(x)$ has roots α_0, α_1 and α_2 of order $\phi_3(p) = p^2 + p + 1$ and associated linear recurring relation

$$s_k = as_{k-1} - bs_{k-2} + s_{k-3}. \tag{4.2}$$

Since the roots are conjugates, $\alpha_i = \alpha_0^{p^i}$ for $i = 0, 1, 2$. From Theorem 3.3 it follows that the elements of a sequence satisfying (4.2) can be written as

$$s_n = \alpha_0^n + \alpha_1^n + \alpha_2^n,$$

with initial values

$$s_0 = 3,$$

$$\begin{aligned} s_1 &= \alpha_0 + \alpha_1 + \alpha_2 \\ &= a, \end{aligned}$$

$$\begin{aligned} s_2 &= \alpha_0^2 + \alpha_1^2 + \alpha_2^2 \\ &= (\alpha_0 + \alpha_1 + \alpha_2)^2 - 2(\alpha_0\alpha_1 + \alpha_0\alpha_2 + \alpha_1\alpha_2) \\ &= a^2 - 2b. \end{aligned}$$

This sequence is called the *characteristic sequence* of $f(x)$ and its k^{th} term is denoted by $s_k(a, b)$. The *reciprocal sequence* of $(s_k(a, b))$ is the sequence generated by the reciprocal of $f(x)$, namely $f^{-1}(x) = x^3 - bx^2 + ax - 1$ with

associated linear recurring relation

$$s_{k-3} = bs_{k-2} - as_{k-1} + s_k.$$

The k^{th} term of the reciprocal sequence is denoted by $s_{-k}(a, b)$.

As in the Polynomial Representation, the public key is given by the polynomial

$$\begin{aligned} f_{priv}(x) =& \left(x - \alpha_0^{priv}\right)\left(x - \alpha_1^{priv}\right)\left(x - \alpha_2^{priv}\right) \\ =& x^3 - \left(\alpha_0^{priv} + \alpha_1^{priv} + \alpha_2^{priv}\right)x^2 \\ &+ \left(\alpha_0^{priv}\alpha_1^{priv} + \alpha_0^{priv}\alpha_2^{priv} + \alpha_1^{priv}\alpha_2^{priv}\right)x - \alpha_0^{priv}\alpha_1^{priv}\alpha_2^{priv} \\ =& x^3 - s_{priv}(a, b) + \left(\alpha_2^{-priv} + \alpha_1^{-priv} + \alpha_0^{-priv}\right)x - 1 \\ =& x^3 - s_{priv}(a, b)x^2 + s_{-priv}(a, b)x - 1 \end{aligned}$$

where $priv \in [2, \text{ord}(\alpha_0))$ and $(priv, \text{ord}(\alpha_0)) = 1$. The public key $f_{priv}(x)$ is irreducible: Since $(priv, \text{ord}(\alpha_0)) = 1$ it follows that $<\alpha_0> = <\alpha_0^{priv}>$. Thus $f_{priv}(x)$ is irreducible, since $f(x)$ is irreducible.

For the private key, $priv$, the corresponding public key is given by the polynomial $f_{priv}(x)$ which is represented by $pub = (s_{priv}(a, b), s_{-priv}(a, b))$. Using the private keys $priv_A$ and $priv_B$, the shared secret will be the polynomial $f_{priv_A \cdot priv_B}(x)$ represented by $(s_{priv_A \cdot priv_B}(a, b), s_{-priv_A \cdot priv_B}(a, b))$.

Note that the system polynomial $f(x)$ is used to define the sequence that is used to derive the public key of each user, where the public key of a user is used to generate the sequence used to derive the shared secret. Thus the initial values for the sequence used to derive the shared secret is $3, s_{priv}$ and $s_{priv}^2 - 2s_{-priv}$.

4.2.2 Computation

In the computations of the CFE protocol a more efficient method is given, than that of the Compact Subgroup Protocol. It uses a linear shift register and avoids the Berlekamp-Massey algorithm. In the computation of the system parameters elements of the field $\mathbb{F}_{p^3}$ are used but only elements of $\mathbb{F}_p$

are used for computing the shared secret.

The computation of the system parameters is straight forward and is given in Algorithm 7.

Algorithm 7: Computing system parameters for CFE

Data: n_p the required bit length of the prime p.
Output: prime p and system polynomial (a, b).
1 Choose $p \in \{n \in \mathbb{Z}_{2^{n_p}} \setminus \mathbb{Z}_{2^{n_p-1}} : n \text{ is prime}\}$;
2 Choose $\alpha \in \{\beta \in \mathbb{F}_{p^3} : \text{ord}(\beta) = \phi_3(p)\}$;
3 $f(x) \leftarrow$ the minimal polynomial of α over $\mathbb{F}_p$;
4 $(a, b) \leftarrow$ the coefficient of x and x^2 in $f(x)$;

The computation of the shared secret is based on the method of repeated squaring, given in Section 1.3, which is a more direct method of computation than that used in the Compact Subgroup Protocol. The index of the term in the sequence that is required is treated as the exponent in the repeated squaring algorithm. Depending on the current bit of the index, one of three functions is executed to get the next sequence element. The initial values (s_{-1}, s_0, s_1) are computed using

$$s_k = \alpha_0^k + \alpha_1^k + \alpha_2^k \tag{4.3}$$

and are

$$\begin{aligned} s_{-1} &= \alpha_0^{-1} + \alpha_1^{-1} + \alpha_2^{-1} \\ &= \alpha_1\alpha_2 + \alpha_1\alpha_3 + \alpha_0\alpha_2 \\ &= b, \\ s_0 &= 3, \\ s_1 &= \alpha_0 + \alpha_1 + \alpha_2 \\ &= a. \end{aligned}$$

These initial values are used to compute the sequences (s_k) and (s_{-k}) simultaneously, as the relations used in the computations need values from both

sequences. The linear relations for these two sequences are

$$s_k = as_{k-1} - bs_{k-2} + s_{k-3}$$

and

$$s_{k-3} = bs_{k-2} - as_{k-1} + s_k$$

respectively. To avoid the transmission of the initial values, these values are computed from the characteristic polynomial. Therefore, the algorithm that is used to compute the shared secret is much simpler than the algorithm used in the Compact Subgroup Protocol, i.e. avoiding the use of the Berlekamp-Massey algorithm.

Lemma 4.3. [8, Lemma 3] *Let (s_k) be the characteristic sequence of the polynomial $f(x)$, defined by (4.1), and (s_{-k}) its reciprocal sequence. Then for any positive integers n and m, with $n \neq m$,*

$$s_{2n} = s_n^2 - 2s_{-n} \tag{4.4}$$

$$s_n s_m - s_{n-m}s_{-m} = s_{n+m} - s_{n-2m}. \tag{4.5}$$

Proof. Firstly, by using (4.3)

$$s_{2n} = \alpha_0^{2n} + \alpha_1^{2n} + \alpha_2^{2n} \text{ and } s_n^2 = (\alpha_0^n + \alpha_1^n + \alpha_2^n)^2$$

are obtained. By using $\alpha_0\alpha_1\alpha_2 = 1$ it follows that

$$\begin{aligned} s_n^2 &= \alpha_0^{2n} + \alpha_1^{2n} + \alpha_2^{2n} + 2(\alpha_0^n\alpha_1^n + \alpha_1^n\alpha_2^n + \alpha_2^n\alpha_0^n) \\ &= s_{2n} + 2\left(\alpha_2^{-n} + \alpha_2^{-n} + \alpha_0^{-n}\right) \\ &= s_{2n} + 2s_{-n}. \end{aligned}$$

Secondly it follows from $\text{ord}(\alpha) \equiv 0 \pmod{(p^2+p+1)}$ that

$$\begin{aligned} s_n s_m =& \alpha_0^{n+m} + \alpha_1^{n+m} + \alpha_2^{n+m} + \alpha_0^n \alpha_1^m + \alpha_0^n \alpha_2^m + \alpha_1^n \alpha_0^m + \alpha_1^n \alpha_2^m \\ &+ \alpha_2^n \alpha_0^m + \alpha_2^n \alpha_1^m \\ =& s_{n+m} + \alpha_0^n \alpha_1^m + \alpha_0^n \alpha_2^m + \alpha_1^n \alpha_0^m + \alpha_1^n \alpha_2^m + \alpha_2^n \alpha_0^m + \alpha_2^n \alpha_1^m \end{aligned}$$

and

$$\begin{aligned} s_{n-m} s_{-m} =& \alpha_0^{n-2m} + \alpha_1^{n-2m} + \alpha_2^{n-2m} + \alpha_0^{n-m} \alpha_1^{-m} + \alpha_0^{n-m} \alpha_2^{-m} \\ &+ \alpha_1^{n-m} \alpha_0^{-m} + \alpha_1^{n-m} \alpha_2^{-m} + \alpha_2^{n-m} \alpha_0^{-m} + \alpha_2^{n-m} \alpha_1^{-m} \\ =& s_{n-2m} + \alpha_0^{n-m-mp} + \alpha_0^{n-m-mp^2} + \alpha_0^{np-m-mp} \\ &+ \alpha_0^{np-mp-mp^2} + \alpha_0^{np^2-m-mp^2} + \alpha_0^{np^2-mp-mp^2} \\ =& s_{n-2m} + \alpha_0^{n+mp^2} + \alpha_0^{n+mp} + \alpha_0^{np+mp^2} \alpha_0^{np+m} \\ &+ \alpha_0^{np^2+mp} + \alpha_0^{np^2+m} \\ =& s_{n-2m} + \alpha_0^n \alpha_2^m + \alpha_0^n \alpha_1^m + \alpha_1^n \alpha_2^m + \alpha_1^n \alpha_0^m + \alpha_2^n \alpha_1^m + \alpha_2 \alpha_0^m, \end{aligned}$$

thus

$$s_n s_m - s_{n-m} s_{-m} = s_{n+m} - s_{n-2m}.$$

□

The above lemma will be used to construct relations for the computation of the sequence. The index of the required sequence element is written in the form

$$k = \sum_{j=0}^{r} k_j 2^{r-j}, \tag{4.6}$$

and let $t_j = k_j + 2t_{j-1}$ with $t_0 = k_0 \neq 0$. If $k_j \in \{0, 1\}$ then (4.6) is the binary expansion, and if $k_j \in \{-1, 0, 1\}$ then (4.6) is a signed digit representation. The sequence elements s_{t_j} are computed where $t_j = k_j + 2t_{j-1}$. The following

relations

$$\begin{aligned} s_{2t_j-2} &= s_{t_j-1}^2 - 2s_{-(t_j-1)} \\ s_{2t_j-1} &= s_{t_j}s_{t_j-1} - bs_{-t_j} + s_{-(t_j+1)} \\ s_{2t_j} &= s_{t_j}^2 - 2s_{-t_j} \\ s_{2t_j+1} &= s_{t_j}s_{t_j+1} - as_{-t_j} + s_{-(t_j-1)} \\ s_{2t_j+2} &= s_{t_j+1}^2 - 2s_{-(t_j+1)} \end{aligned}$$

are constructed from (4.4) and (4.5) using $n = t_j - 1$; $m = -t_j$ and $n = -1$; $n = t_j$; $m = -t_j$ and $n = 1$; $n = t_j + 1$ respectively. From these relations it is seen that both the sequence (s_k) and its reciprocal sequence are needed. From the above relations three relations are selected depending on the current digit in the expansion of the number k. The relations are:

If $k_j = -1$ **then** $t_j = 2t_{j-1} - 1$**:**

$$\begin{aligned} s_{t_j-1} &= s_{t_{j-1}-1}^2 - 2s_{-(t_{j-1}-1)} \\ s_{t_j} &= s_{t_{j-1}}s_{t_{j-1}-1} - bs_{-t_{j-1}} + s_{-(t_{j-1}+1)} \\ s_{t_j+1} &= s_{t_{j-1}}^2 - 2s_{-t_{j-1}}. \end{aligned}$$

If $k_j = 0$ **then** $t_j = 2t_{j-1}$**:**

$$\begin{aligned} s_{t_j-1} &= s_{t_{j-1}}s_{t_{j-1}-1} - bs_{-t_{j-1}} + s_{-(t_{j-1}+1)} \\ s_{t_j} &= s_{t_{j-1}}^2 - 2s_{-t_{j-1}} \\ s_{t_j+1} &= s_{t_{j-1}}s_{t_{j-1}+1} - as_{-t_{j-1}} + s_{-(t_{j-1}-1)}. \end{aligned}$$

If $k_j = 1$ **then** $t_j = 2t_{j-1} + 1$**:**

$$\begin{aligned} s_{t_j-1} &= s_{t_{j-1}}^2 - 2s_{-t_{j-1}} \\ s_{t_j} &= s_{t_{j-1}}s_{t_{j-1}+1} - as_{-t_{j-1}} + s_{-(t_{j-1}-1)} \\ s_{t_j+1} &= s_{t_{j-1}+1}^2 - 2s_{-(t_{j-1}+1)}. \end{aligned}$$

To be able to compute the next triple from the above relation sets, the

triples (s_{j-1}, s_j, s_{j+1}) and $(s_{-j-1}, s_{-j}, s_{-j+1})$ are needed. The same two triples are needed to compute the reciprocal sequence as can be seen from the next relation sets that are obtained in the same way as the previous set, but by using $-t_j = -k_j - 2t_{j-1}$ instead.

For $k_j = -1$ thus $-t_j = -2t_{j-1} + 1$:

$$\begin{aligned} s_{-t_j-1} &= s^2_{-t_{j-1}} - 2s_{t_{j-1}} \\ s_{-t_j} &= s_{-t_{j-1}} s_{-t_{j-1}+1} - a s_{t_{j-1}} + s_{t_{j-1}+1} \\ s_{-t_j+1} &= s^2_{-t_{j-1}+1} - 2s_{t_{j-1}-1}. \end{aligned}$$

For $k_j = 0$ thus $-t_j = -2t_{j-1}$:

$$\begin{aligned} s_{-t_j-1} &= s_{-t_{j-1}} s_{-t_{j-1}-1} - b s_{t_{j-1}} + s_{t_{j-1}-1} \\ s_{-t_j} &= s^2_{-t_{j-1}} - 2s_{t_{j-1}} \\ s_{-t_j+1} &= s_{-t_{j-1}} s_{-t_{j-1}+1} - a s_{t_{j-1}} + s_{t_{j-1}+1}. \end{aligned}$$

For $k_j = 1$ thus $-t_j = -2t_{j-1} - 1$:

$$\begin{aligned} s_{-t_j-1} &= s^2_{-t_{j-1}-1} - 2s_{t_{j-1}+1} \\ s_{-t_j} &= s_{-t_{j-1}} s_{-t_{j-1}-1} - b s_{t_{j-1}} + s_{t_{j-1}-1} \\ s_{-t_j+1} &= s^2_{-t_{j-1}} - 2s_{t_{j-1}}. \end{aligned}$$

Algorithm 8 gives the combining function for CFE and is also used in the generation of a user's public key. The functions F_{-1}, F_0, and F_1 used in Algorithm 8 are defined by the above relations. The functions F_{k_j} are defined by transforming the two triples $\left(s_{t_j-1}, s_{t_j}, s_{t_j+1}\right)$ and $\left(s_{-t_j-1}, s_{-t_j}, s_{-t_j+1}\right)$ to the two triples $\left(s_{2t_j-1+k_j}, s_{2t_j+k_j}, s_{2t_j+1+k_j}\right)$ and $\left(s_{-2t_j-1-k_j}, s_{-2t_j-k_j}, s_{-2t_j+1-k_j}\right)$ for $k_j \in \{-1, 0, 1\}$.

Lemma 4.4 gives the number of operations in Algorithm 8. To improve the performance of the algorithm, the iterations where $k_i = 0$ need to be reduced. The Hamming weight of k can be decreased by using signed digit representation, see [9]. In this dissertation this is not done.

Algorithm 8: Combining function for CFE

Data: The index $k = (k_{r-1}, \ldots, k_0)_2$ and the polynomial (a, b).
Output: $(s_{-k-1}, s_{-k}, s_{-k+1}, s_{k-1}, s_k, s_{k+1})$

```
M ← (b, 3, a, b, 3, a);
for i = r − 1 to 0 do
    M ← F_{k_i}(M);
end
(s_{−k−1}, s_{−k}, s_{−k+1}, s_{k−1}, s_k, s_{k+1}) ← M;
```

Lemma 4.4. *Let $f(x)$ be a system polynomial for the CFE protocol and (s_k) the associated characteristic sequence. The k^{th} term in the sequence can be computed with $6\log_2 k$ multiplications and $5\log_2 k - H_k$ additions in $\mathbb{F}_p$, where H_k is the Hamming weight of k.*

4.2.3 Security

The security proof of the CFE Protocol is a specific case of Section 3.3. Using a Diffie-Hellman group that does not have a prime order makes an attack using the Chinese Remainder Theorem applicable, see Section 2.3.2. The security of the protocol is therefore based on the largest prime factor of $\phi_3(p)$.

Example 4.5 (Small implementation of CFE)**.** To simplify the example, only the binary expansion of the private key will be used. Let the characteristic of the field be 101. Then $\phi_3(101) = 10303$ which is a prime number. Select the order of the Diffie-Hellman group as 10303. Define the field $\mathbb{F}_{101^3}$ by the irreducible polynomial $x^3 + 23x^2 + 40x + 69$ with root α. An element of order 10303 is $15\alpha^2 + 79\alpha + 29$ and the system polynomial is given by $f(x) = x^3 + 45x^2 + 3x + 100$. This gives $a = -45 = 56$ and $b = 3$, thus the initial values of the sequence are 3, 3 and 56.

Let the private key of Alice be $priv_A = 1882 = 11101011010_2$. The computation of Algorithm 8 is given in Table 4.1.

Iteration	k_i	t_i	s_{t_i-1}	s_{t_i}	s_{t_i+1}	s_{-t_i-1}	s_{-t_i}	s_{-t_i+1}
0	-	0	3	3	56	3	3	56
1	1	1	3	56	100	99	3	3
2	1	3	100	82	5	6	31	99
3	1	7	97	86	13	26	40	90
4	0	14	67	44	79	71	14	9
5	1	29	90	75	39	35	20	7
6	0	58	59	30	95	60	48	19
7	1	117	97	80	17	77	21	22
8	1	235	96	4	34	37	60	79
9	0	470	39	98	87	82	57	5
10	1	941	97	87	32	86	76	23
11	0	1882	15	44	66	9	47	39

Table 4.1: Computing the public key of Alice in CFE

From the table it is seen that $s_{1882} = 44$ and $s_{-1882} = 47$, thus the public key is $pub_A = x^3 + 57x^2 + 47x + 100$.

The private key of user Bob is selected as $priv_B = 7998$ and the associated public key $pub_B = x^3 + 100x^2 + 65x + 100$ is computed in a similar manner as the public key of Alice.

For user Alice to compute the shared secret the private key $priv_A$ and the public key pub_B are used. The values are 65, 3 and 1 are the initial values for the linear sequence and is constructed from $priv_B$. The computed shared secret is $x^3 + 17x^2 + 56x + 100$.

4.3 Extended XTR

Lenstra defined the system XTR in [12] based on a third degree irreducible polynomial over the field $\mathbb{F}_{p^2}$. This idea was extended in [16] to any field $\mathbb{F}_{p^{2m}}$. In this dissertation, the system in [16] is called the Extended XTR (EXTR). The protocol follows the Compact Subgroup Protocol, but like CFE defined in the previous section, a more direct computation method is used to compute the public keys and the associated shared secret. An additional performance improvement is achieved by using an optimal normal basis.

In the next section EXTR is described; in Section 4.3.2 the parameter selection is given. Section 4.3.3 gives a method of computation and the last section provides a security proof.

4.3.1 Description of EXTR

For any positive integer m, the system is defined by an irreducible system polynomial

$$f(x) = x^3 - cx^2 + c^{p^m}x - 1,\ c \in \mathbb{F}_{p^{2m}} \tag{4.7}$$

with $p \nmid 6m$ and $\text{ord}(f) \nmid 6m$ and where the roots $\alpha_0, \alpha_1, \alpha_2$ in $\mathbb{F}_{p^{6m}}$ have prime order dividing $\phi_{6m}(p)$. These constraints are needed for security reasons as from Lemma 2.13 it follows that the smallest field containing the roots of $f(x)$ is $\mathbb{F}_{p^{6m}}$. From the primality of $\text{ord}(\alpha_0)$ it follows that the minimum polynomial of α_0^i is irreducible for $i \in [1, \text{ord}(\alpha_0))$. Furthermore $2m+1$ must be prime and p primitive in $\mathbb{Z}_{2m+1}$ to ensure the existence of an optimal normal basis.

The associated linear recurrence relation is

$$s_{k+3} = cs_{k+2} - c^{p^m}s_{k+1} + s_k,\ k \geq 0. \tag{4.8}$$

The sequence (s_k) satisfying (4.8) and with initial values $s_0 = 3$, $s_1 = c$ and $s_2 = c^2 - 2c^{p^m}$ is called the *characteristic sequence* of $f(x)$. It follows from Theorem 3.3 that the characteristic sequence (s_k) satisfies

$$s_k = \alpha_0^k + \alpha_1^k + \alpha_2^k,\ k \geq 0. \tag{4.9}$$

Since the roots are conjugates, $\alpha_i = \alpha_0^{p^{2mi}}$ for $i = 0, 1, 2$ it follows that

$$s_k = Tr_{p^{6m}/p^{2m}}\left(\alpha_0^k\right) \tag{4.10}$$

where $Tr_{p^{6m}/p^{2m}}$ is the trace of α_0^k, see [15, Definition 2.22].

In Lemma 4.13 it is shown that $\phi_{6m}(p)|(p^{2m} - p^m + 1)$, ensuring that the order of the roots of $f(x)$ divides $p^{2m} - p^m + 1$.

Lemma 4.6. *Let $f(x)$ be a third degree polynomial over $\mathbb{F}_{p^{2m}}[x]$, with roots $\alpha_0, \alpha_1, \alpha_2$. If $f(x)$ is irreducible and the order of the roots divides $p^{2m} - p^m + 1$ then the polynomial is of the form*

$$f(x) = x^3 - cx^2 + c^{p^m}x - 1, \ c \in \mathbb{F}_{p^{2m}}.$$

Proof.

$$\begin{aligned}
f(x) =& (x-\alpha_0)(x-\alpha_1)(x-\alpha_2) \\
=& x^3 - (\alpha_0+\alpha_1+\alpha_2)x^2 + (\alpha_0\alpha_1 + \alpha_0\alpha_2 + \alpha_1\alpha_2)x - \alpha_0\alpha_1\alpha_2 \\
=& x^3 - cx^2 + \left(\alpha_0^{1+p^{2m}} + \alpha_0^{1+p^{4m}} + \alpha_0^{p^{2m}+p^{4m}}\right)x - \alpha_0^{1+p^{2m}+p^{4m}} \\
=& x^3 - cx^2 + \left(\alpha_0^{p^m} + \alpha_0^{1-p^m} + \alpha_0^{p^{2m}-p^m}\right)x - \alpha_0^{1+p^{2m}-p^m} \\
=& x^3 - cx^2 + \left(\alpha_0^{p^m} + \alpha_0^{-p^{2m}} + \alpha_0^{p^{2m}-p^m}\right)x - 1 \\
=& x^3 - cx^2 + \left(\alpha_0 + \alpha_0^{-p^m} + \alpha_0^{p^m-1}\right)^{p^m} x - 1 \\
=& x^3 - cx^2 + \left(\alpha_0 + \alpha_0^{p^{4m}} + \alpha_0^{p^{2m}}\right)^{p^m} x - 1 \\
=& x^3 - cx^2 + (\alpha_0+\alpha_1+\alpha_2)^{p^m} x - 1 \\
=& x^3 - cx^2 + c^{p^m}x - 1
\end{aligned}$$

where

$$c = \alpha_0 + \alpha_1 + \alpha_2$$

and the modulus relations used are

$$\begin{aligned}
p^{2m} &\equiv p^m - 1 \\
p^{4m} &\equiv -p^m \\
1 - p^m &\equiv -p^{2m} \\
-1 &\equiv p^{2m} - p^m,
\end{aligned}$$

all modulo $(p^{2m} - p^m + 1)$. □

Since $\text{ord}(\alpha_0)$ is a prime number, the group $<\alpha_0>$ is not contained in any proper subfield of $\mathbb{F}_{p^{6m}}$. Therefore the public key

$$f_{priv}(x) = (x - \alpha_0^{priv})(x - \alpha_1^{priv})(x - \alpha_2^{priv})$$

defined by (3.3) and associated with a private key $priv \in [2, \text{ord}(\alpha_0))$, is irreducible. By Lemma 4.6

$$f_{priv}(x) = x^3 - c_{priv}x^2 + c_{priv}^{p^m}x - 1$$

where

$$c_{priv} = \alpha_0^{priv} + \alpha_1^{priv} + \alpha_2^{priv}. \tag{4.11}$$

The shared secret for Alice and Bob is now given by $f_{priv_A \cdot priv_B}(x)$. The key agreement is illustrated in Figure 4.1. It is important to note that $pub_A = f_{priv_A}(x)$ and $pub_B = f_{priv_B}(x)$ are the public data that is exchanged in the clear.

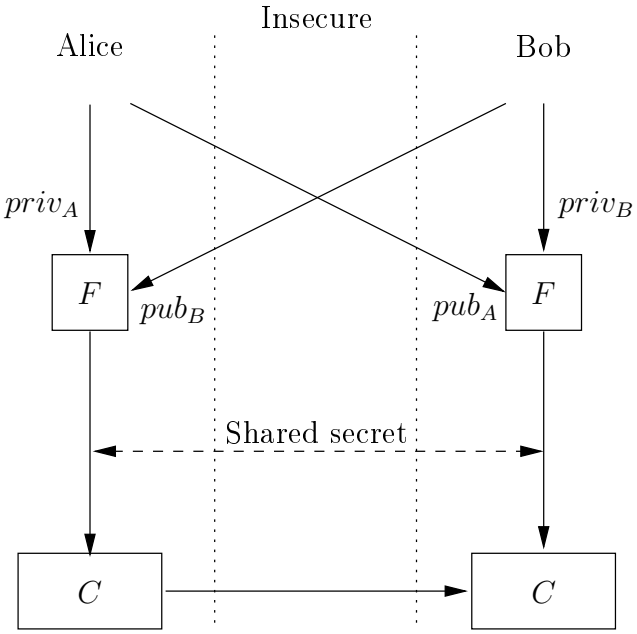

Figure 4.1: Two parties using a public key distribution system.

Each user computes the shared secret independently. It is seen that the computed values are the same by considering

$$\begin{aligned} c_{priv_A \cdot priv_B} &= \alpha_0^{priv_A \cdot priv_B} + \alpha_1^{priv_A \cdot priv_B} + \alpha_2^{priv_A \cdot priv_B} \\ &= \alpha_0^{priv_B \cdot priv_A} + \alpha_1^{priv_B \cdot priv_A} + \alpha_2^{priv_B \cdot priv_A} \\ &= c_{priv_B \cdot priv_A}. \end{aligned}$$

The coefficient $c_{priv_B \cdot priv_A}$ of the shared polynomial can be computed from the characteristic sequence, as seen from (4.8) and (4.11).

4.3.2 Parameter Selection

Parameter selection in EXTR has three parts, namely the selection of an optimal normal basis, the selection of the system polynomial and the computation of $\phi_{6m}(p)$. The optimal normal basis is needed for efficient computations. An irreducible polynomial is needed for the selection of the system polynomial. Lastly, a more efficient formula is given for the computation of $\phi_{6m}(p)$ from which it follows that $\phi_{6m}(p)|(p^{2m} - p^m + 1)$.

Optimal normal basis

Any finite field $\mathbb{F}_q$ can be extended by using an irreducible polynomial of degree n to construct a finite field $\mathbb{F}_{q^n} \cong < \mathbb{F}_q[x]/< f(x) >$. This field $\mathbb{F}_{q^n}$ can be represented by a basis over $\mathbb{F}_q$.

Definition 4.7. [15, Definition 2.32] A basis of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$ of the form $\left\{\alpha, \alpha^q, \ldots, \alpha^{q^{n-1}}\right\}$, consisting of a suitable element $\alpha \in \mathbb{F}_{q^n}$ and its conjugates with respect to $\mathbb{F}_q$, is called a *normal basis* of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$.

Note that if $G = Gal(\mathbb{F}_{q^n} : \mathbb{F}_q)$ and $\alpha \in \mathbb{F}_{q^n}$, then $\{\sigma(\alpha) : \sigma \in G\} = \left\{\alpha, \alpha^q, \alpha^{q^2}, \ldots, \alpha^{q^{n-1}}\right\}$. The element α will always be selected, in such a way that $\{\sigma(\alpha) : \sigma \in G\}$ is a normal basis. Theorem 2.35 of [15] gives the existence of a normal basis for any finite field.

For efficient computations *optimal* normal bases, introduced in [20], will be used and discussed later. Before optimal normal bases are discussed, some

theory of dual bases is needed.

Definition 4.8. [15, Definition 2.30] Let K be a finite field and L a finite extension of K. Then two bases $\{\alpha_1, \ldots, \alpha_m\}$ and $\{\beta_1, \ldots, \beta_m\}$ of L over K are said to be *dual bases*, if for $1 \leq i, j \leq m$

$$Tr_{L/K}(\alpha_i \beta_j) = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j. \end{cases}$$

It is now shown how the trace function can be used to determine the coordinates of an element $A \in \mathbb{F}_{q^n}$ with respect to a normal basis $\{\sigma(\alpha) : \sigma \in G\}$. Let $\alpha, \beta \in \mathbb{F}_{q^n}$ such that $\{\sigma(\alpha) : \sigma \in G\}$ and $\{\sigma(\beta) : \sigma \in G\}$ are dual normal bases of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$. For any $\tau \in G$

$$\begin{aligned} Tr_{\mathbb{F}_{q^n}/\mathbb{F}_q}(A\tau(\beta)) &= Tr_{\mathbb{F}_{q^n}/\mathbb{F}_q}\left(\tau(\beta) \sum_{\sigma \in G} a_\sigma \sigma(\alpha)\right) \\ &= Tr_{\mathbb{F}_{q^n}/\mathbb{F}_q}\left(\sum_{\sigma \in G} a_\sigma \sigma(\alpha)\tau(\beta)\right) \\ &= \sum_{\sigma \in G} a_\sigma Tr_{\mathbb{F}_{q^n}/\mathbb{F}_q}(\sigma(\alpha)\tau(\beta)) \\ &= a_\tau. \end{aligned}$$

Thus, $Tr_{\mathbb{F}_{q^n}/\mathbb{F}_q}(x\tau(\beta))$ defines a linear transformation that determines the τ^{th} coordinate of A. This transformation is uniquely defined by $\tau(\beta)$, see [15, Theorem 2.24].

Next, it is shown that the dual of a normal basis is normal. Let $\{\beta_1, \ldots, \beta_n\}$ be the dual of a normal basis $\{\sigma(\alpha) : \sigma \in G\}$. Then for integers i and j

$$\begin{aligned} Tr_{\mathbb{F}_{q^n}/\mathbb{F}_q}\left(\alpha^{q^i}\beta_j\right) &= Tr_{\mathbb{F}_{q^n}/\mathbb{F}_q}\left(\left[\alpha^{q^i}\beta_j\right]^q\right) \\ &= Tr_{\mathbb{F}_{q^n}/\mathbb{F}_q}\left(\alpha^{q^{i+1}}\beta_j^q\right). \end{aligned}$$

For any linear transformation from $\mathbb{F}_{q^n}$ to $\mathbb{F}_q$ there exists a unique $\beta \in \mathbb{F}_{q^n}$,

such that the linear transformation in x is defined by $Tr(\beta \cdot x)$. Since the two linear transformations $Tr_{\mathbb{F}_{q^n}/\mathbb{F}_q}(\beta_j x)$ and $Tr_{\mathbb{F}_{q^n}/\mathbb{F}_q}\left(\beta_j^q x\right)$ with indeterminate x are identical, it follows that $\beta_j^q = \beta_{j+1}$. Hence $\{\beta_1, \ldots, \beta_n\} = \left\{\beta_1, \beta_1^q \ldots, \beta_1^{q^{n-1}}\right\}$ is a normal basis.

Before the existence of optimal normal bases is proven, the multiplication method of Massey and Omura [17] is discussed. Computations in $\mathbb{F}_{q^n}$ are done with respect to a normal basis $\{\sigma(\alpha) : \sigma \in G\}$ of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$. Let $A, B \in \mathbb{F}_{q^n}$, with

$$A = \sum_{\sigma \in G} a_\sigma \sigma(\alpha) \text{ and } B = \sum_{\sigma \in G} b_\sigma \sigma(\alpha),\ a_\sigma, b_\sigma \in \mathbb{F}_q.$$

Addition is computed component wise. The product of A and B is given by

$$\begin{aligned} AB &= \sum_{\sigma,\gamma \in G} a_\sigma b_\gamma \sigma(\alpha)\gamma(\alpha) \\ &= \sum_{\sigma,\gamma \in G} a_\sigma b_\gamma \sigma\left(\alpha\sigma^{-1}(\gamma(\alpha))\right). \end{aligned}$$

Notation: The automorphism $\sigma_i \in G$ is defined by $\sigma_i : \alpha \mapsto \alpha^{q^i}$. A product $\alpha\sigma(\alpha)$ with $\sigma \in G$ can be written as

$$\alpha\sigma(\alpha) = \sum_{\tau \in G} d_\alpha(\tau, \sigma)\tau(\alpha), \tag{4.12}$$

where $d_\alpha(\cdot, \cdot)$ is an $n \times n$ matrix over $\mathbb{F}_q$ defined by

$$\begin{bmatrix} \alpha\sigma_1(\alpha) \\ \alpha\sigma_2(\alpha) \\ \vdots \\ \alpha\sigma_n(\alpha) \end{bmatrix} = \begin{bmatrix} d(\sigma_1, \sigma_1) & d(\sigma_2, \sigma_1) & \cdots & d(\sigma_n, \sigma_1) \\ \vdots & & \ddots & \\ d(\sigma_1, \sigma_n) & d(\sigma_2, \sigma_n) & \cdots & d(\sigma_n, \sigma_n) \end{bmatrix} \begin{bmatrix} \sigma_1(\alpha) \\ \sigma_2(\alpha) \\ \vdots \\ \sigma_n(\alpha) \end{bmatrix}$$

Since α is a unit, $\{\alpha \cdot \sigma_i(\alpha) : 0 \leq i \leq n-1\}$ is also a basis of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$ and

the matrix d_α is invertible. The product AB is now given by

$$\begin{aligned} AB &= \sum_{\sigma,\gamma\in G} a_\sigma b_\gamma \sigma\left(\sum_{\tau\in G} d_\alpha\big(\tau,\sigma^{-1}\gamma\big)\tau(\alpha)\right) \\ &= \sum_{\sigma,\gamma\in G} a_\sigma b_\gamma \sum_{\tau\in G} d_\alpha\big(\tau,\sigma^{-1}\gamma\big)\sigma(\tau(\alpha)) \\ &= \sum_{\sigma,\gamma\in G} a_\sigma b_\gamma \sum_{\tau\in G} d_\alpha\big(\sigma^{-1}\tau,\sigma^{-1}\gamma\big)\tau(\alpha). \end{aligned}$$

Multiplication can therefore be computed through the use of the matrix d_α. Computational efficiency of multiplication is determined by the number of non-zero entries in the matrix. The minimum number of non-zero elements in d_α is determined next. From

$$\begin{aligned} \sum_{\sigma,\tau\in G} d_\alpha(\tau,\sigma)\tau(\alpha) &= \sum_{\sigma\in G} \alpha\sigma(\alpha) \\ &= \alpha\sum_{\sigma\in G}\sigma(\alpha) \\ &= \alpha Tr_{\mathbb{F}_{q^n}/\mathbb{F}_q}(\alpha) \end{aligned}$$

it follows that

$$\sum_{\sigma\in G} d_\alpha(\tau,\sigma) = \begin{cases} Tr_{\mathbb{F}_{q^n}/\mathbb{F}_q}(\alpha) & \text{if } \tau = 1, \\ 0 & \text{if } \tau \neq 1. \end{cases}$$

Therefore each row and column of the matrix d_α contains at least one non-zero element. From the above summation, it follows that all columns with $\tau \neq 1$ have at least two non-zero values. Thus

$$\#\{(\tau,\sigma)\in(G,G) : d_\alpha(\tau,\sigma)\neq 0\} \geq 2n-1. \tag{4.13}$$

If the number of non-zero elements is $2n-1$, the basis is called an *optimal normal basis*.

The matrix d_α defined in (4.12) can also be defined by the dual basis

$\{\sigma(\beta) : \sigma \in G\}$ of $\{\sigma(\alpha) : \sigma \in G\}$ i.e.

$$\alpha\tau(\beta) = \sum_{\sigma \in G} d_\alpha(\tau, \sigma)\sigma(\beta) \tag{4.14}$$

of all τ in G. It is sufficient to observe that the coefficient of $\alpha\tau(\beta)$ at $\sigma(\beta)$ is given by

$$\begin{aligned} Tr_{\mathbb{F}_{q^n}/\mathbb{F}_q}((\alpha\tau(\beta))\sigma(\alpha)) &= Tr_{\mathbb{F}_{q^n}/\mathbb{F}_q}((\alpha\sigma(\alpha))\tau(\beta)) \\ &= Tr_{\mathbb{F}_{q^n}/\mathbb{F}_q}\left(\sum_{\rho \in G} d_\alpha(\rho, \sigma)\rho(\alpha)\tau(\beta)\right) \\ &= d_\alpha(\tau, \sigma). \end{aligned}$$

In [7] it is proven that only two types of optimal normal bases exist. Fields with characteristic 2 are considered less secure and therefore optimal normal bases of fields with characteristic 2 will not be considered.

Theorem 4.9. *Let $\mathbb{F}_q$ be a finite field of characteristic not equal to 2 and $\mathbb{F}_{q^{n-1}}$ a finite Galois extension of $\mathbb{F}_q$, with Galois group G and let $\alpha \in \mathbb{F}_{q^{n-1}}$. Then $\{\sigma(\alpha)\}_{\sigma \in G}$ is an optimal normal basis if and only if there is a prime n, a primitive n^{th} root of unity ζ in some algebraic extension of $\mathbb{F}_q$ and an element $c \in \mathbb{F}_q^*$ such that the irreducible polynomial of ζ over $\mathbb{F}_q$ has degree $n-1$, $\mathbb{F}_{q^{n-1}} = \mathbb{F}_q(\zeta)$ and $\alpha = c\zeta$.*

Proof. It is first proven that the conditions are sufficient for $\{\sigma(\alpha)\}_{\sigma \in G}$ to be an optimal normal basis. Since n is prime, it follows that the irreducible polynomial of ζ is the n^{th} cyclotomic polynomial $\phi_n(x) = x^{n-1} + \cdots + x + 1$. Also, since ζ is a primitive n^{th} root of unity an integer i exists for each j, such that $\zeta^{q^j} = \zeta^i$, $1 \leq i \leq n-1$. Now let N be a the normal basis of $\mathbb{F}_{p^{n-1}}$, i.e.

$$\begin{aligned} N =& \left\{\zeta, \zeta^q, \ldots, \zeta^{q^{n-2}}\right\} \\ =& \left\{\zeta, \zeta^2, \ldots, \zeta^{n-1}\right\}. \end{aligned}$$

Note that

$$\zeta \cdot \zeta^i = \zeta^{i+1} \in N,\ 1 \leq i < n-1,$$

and

$$\begin{aligned}\zeta \cdot \zeta^{n-1} &= 1 \\ &= -\zeta^{n-1} - \cdots - \zeta^2 - \zeta.\end{aligned}$$

There are $2n-1$ non-zero elements in the multiplication matrix d_ζ and thus N is an optimal normal basis. Note N is an optimal normal basis if and only if cN is an optimal normal basis.

Now it is proven that the conditions are necessary. Assume that $\{\sigma(\alpha)\}_{\sigma \in G}$ is an optimal normal basis of L over K. From the argument leading to (4.13) it follows that

(1) For each $\tau \in G$, $\tau \neq 1$ only two elements σ exist in G, such that $d_\alpha(\tau, \sigma)$ is non-zero.

(2) Only one element σ exists in G such that $d_\alpha(\sigma_n, \sigma)$ is non-zero and its value is $Tr(\alpha)$.

Thus, there exists a $\mu \in G$ such that $\alpha\beta = Tr_{\mathbb{F}_{q^{n-1}}/\mathbb{F}_q}(\alpha) \cdot \mu(\beta)$, see (4.14). Since $c\alpha$ with $c \in \mathbb{F}_q$ generates an optimal normal basis, without loss of generality, it can be assumed that $Tr_{\mathbb{F}_{q^{n-1}}/\mathbb{F}_q}(\alpha) = -1$. Thus $\alpha\beta = -1\mu(\beta)$. Also

$$\begin{aligned}Tr_{\mathbb{F}_{q^{n-1}}/\mathbb{F}_q}(\alpha) Tr_{\mathbb{F}_{q^{n-1}}/\mathbb{F}_q}(\beta) &= \sum_{\sigma,\tau \in G} \sigma(\alpha)\tau(\beta) \\ &= \sum_{\sigma,\rho \in G} \sigma(\alpha\rho(\beta)) \\ &= \sum_{\rho \in G} Tr_{\mathbb{F}_{q^{n-1}}/\mathbb{F}_q}(\alpha\rho(\beta)) \\ &= 1,\end{aligned}$$

from which it follows that without loss of generality it can be assumed that

$Tr_{\mathbb{F}_{q^{n-1}}/\mathbb{F}_q}(\beta) = -1$.

The case $\mu = 1$ results in no extension field being generated and the case $\mu^n = 1$ for $n > 2$ forces the characteristic of the field to be 2 which is not considered.

Consider the case $\mu^2 = 1$. Then $\alpha = -\mu(\beta)/\beta$ and $\mu(\alpha) = -\mu^2(\beta)/\mu(\beta) = -\beta/\mu(\beta) = 1/\alpha$. Thus

$$\begin{aligned}\alpha\mu(\alpha) &= 1 \\ &= -Tr_{\mathbb{F}_{q^{n-1}}/\mathbb{F}_q}(\alpha) \\ &= \sum_{\sigma\in G} -\sigma(\alpha).\end{aligned}$$

Thus $d(\sigma, \mu) = -1$ for all $\sigma \in G$. For each row where $\sigma \neq 1$, two non-zero entries and a $\sigma^* \neq \mu$ exist in G such that

$$\alpha\sigma(\beta) = \sigma^*(\beta) - \mu(\beta).$$

The mapping $F : G \setminus \{1\} \to G \setminus \{\mu\}$ is defined such that $\sigma \mapsto \sigma^*$ as in the above relation. For any σ and τ not equal to each other it follows that $\alpha\sigma(\beta) \neq \alpha\tau(\beta)$ and thus $F(\sigma) \neq F(\tau)$. This proves that F is injective and since the range and domain of F are finite and of the same cardinality, the function is a bijection. Determining the coefficients $\tau(\alpha)$ of $\alpha\sigma^*(\alpha)$,

$$\begin{aligned}\alpha\tau(\beta) =& \tau^*(\beta) - \mu(\beta) \\ \alpha\tau(\beta)\sigma^*(\alpha) =& \tau^*(\beta)\sigma^*(\alpha) - \mu(\beta)\sigma^*(\alpha) \\ Tr_{\mathbb{F}_{q^{n-1}}/\mathbb{F}_q}([\alpha\sigma^*(\alpha)]\tau(\beta)) =& Tr_{\mathbb{F}_{q^{n-1}}/\mathbb{F}_q}(\tau^*(\beta)\sigma^*(\alpha)) \\ & - Tr_{\mathbb{F}_{q^{n-1}}/\mathbb{F}_q}(\mu(\beta)\sigma^*(\alpha)) \\ =& Tr_{\mathbb{F}_{q^{n-1}}/\mathbb{F}_q}(\tau^*(\beta)\sigma^*(\alpha)) \\ =& \begin{cases} 1 & \text{if } \tau = \sigma \\ 0 & \text{if } \tau \neq \sigma. \end{cases}\end{aligned}$$

Now

$$\alpha\sigma^*(\alpha) = \sigma(\alpha) \text{ for } \sigma^* \neq \mu \text{ and} \tag{4.15}$$

$$\alpha\mu(\alpha) = 1. \tag{4.16}$$

The set $\{1\} \cup \{\sigma(\alpha) : \sigma \in G\}$ is closed under multiplication by α and under the action of G and thus it is a group of order n. Since $\alpha^n = 1$ and $\alpha \neq 1$ it follows that α is a zero of $(x^n - 1)/(x - 1) = x^{n-1} + \ldots + x + 1$. From the fact that the degree of α over K is $n - 1$ and that α is a root of $x^{n-1} + \ldots x + 1$ it follows that $x^{n-1} + \ldots x + 1$ is irreducible. Note that the n^{th} cyclotomic polynomial $x^{n-1} + \ldots + x + 1$ is irreducible if and only if n is a prime number. □

In EXTR it is necessary to find an optimal normal basis for the field $\mathbb{F}_{p^{2m}}$. Thus, according to the above theorem, $2m + 1$ must be prime and p must be primitive in $\mathbb{Z}_{2m+1}$.

Computation of the system polynomial

An irreducible polynomial of order dividing $\phi_{6m}(p)$ is needed. This can be done by considering each irreducible polynomial of degree 3, compute its order and test if the required condition is satisfied. A more efficient method is given for the selection of a required polynomial.

Lemma 4.10. [13, Lemma 2.3.2 (iv) and (vi)] *Let*

$$f(x) = x^3 + cx^2 - c^{p^m}x - 1 \in \mathbb{F}_{p^{2m}}[x]$$

have roots $\alpha_0, \alpha_1, \alpha_2$. *Then for* $j = 0, 1, 2$,

(1) $f(\alpha_j^{-p^m}) = 0$.

(2) $f(x)$ *is irreducible over* $\mathbb{F}_{p^{2m}}$ *if and only if* $\text{ord}(\alpha_j)|(p^{2m} - p^m + 1)$ *and* $\text{ord}(\alpha_j) > 3$.

Proof. (1) From $f(\alpha_j) = \alpha_j^3 - c\alpha_j^2 + c^{p^m}\alpha_j - 1 = 0$ it follows that $\alpha_j \neq 0$. Since $c^{p^{2m}} = c$ and $\alpha_j \neq 0$ it follows that

$$\begin{aligned} 0 &= f(\alpha_j)^{p^m} \\ &= \alpha_j^{3p^m} - c^{p^m}\alpha_j^{2p^m} + c^{p^{2m}}\alpha_j^{p^m} - 1 \\ &= -\alpha_j^{3p^m}\left(\alpha_j^{-3p^m} - c\alpha_j^{-2p^m} + c^{p^m}\alpha_j^{-p^m} - 1\right) \\ &= -\alpha_j^{3p^m} \cdot f(\alpha_j^{-p^m}), \end{aligned}$$

i.e. $f(\alpha_i^{-p^m}) = 0$.

(2) It is first proven that $\text{ord}(\alpha_j)|(p^{2m} - p^m + 1)$ and $\text{ord}(\alpha_j) > 3$ are necessary conditions for $f(x)$ to be irreducible. From (1) it follows that either $\alpha_j = \alpha_j^{-p^m}$ or that $\alpha_j = \alpha_{j+1 \pmod 3}^{-p^m}$ for $j = 0, 1, 2$ or without loss of generality that $\alpha_0 = \alpha_0^{-p^m}$, $\alpha_1 = \alpha_2^{-p^m}$ and $\alpha_2 = \alpha_1^{-p^m}$.

In the first case all α_j's have order dividing $p^m + 1$ and are thus in $\mathbb{F}_{p^{2m}}$.

In the second case it follows from $\alpha_0\alpha_1\alpha_2 = 1$ that

$$\begin{aligned} 1 &= \alpha_0\alpha_2^{-p^m}\alpha_0^{-p^m} \\ &= \alpha_0\alpha_0^{p^{2m}}\alpha_0^{-p^m} \\ &= \alpha_0^{p^{2m}-p^m+1} \end{aligned}$$

so that α_0, α_1 and α_2 have order dividing $p^{2m} - p^m + 1$.

In the last case α_0 has order dividing $p^m + 1$, $\alpha_1 = \alpha_2^{-p^m} = \alpha_1^{p^{2m}}$ so that both α_1 and α_2 have order dividing $p^{2m} - 1$ and again all roots are in $\mathbb{F}_{p^{2m}}$.

Assume that $f(x)$ is an irreducible polynomial. Then $\alpha_j \notin \mathbb{F}_{p^{2m}}$ and therefore $\text{ord}(\alpha_j)|(p^{2m} - p^m + 1)$ for $j = 0, 1, 2$.

Suppose that $\text{ord}(\alpha_j) \leq 3$. Since $p^{2m} - p^m + 1$ is odd, $\text{ord}(\alpha_j) \in \{1, 3\}$,

i.e. $\text{ord}(\alpha_j)|3$. Since

$$\begin{aligned} 3 &= \big(p^{2m} - p^m + 1\big) - (p^m - 2)(p^m + 1) \\ &= \big(p^{2m} - p^m + 1\big) + \big[\big(p^{2m} - p^m + 1\big) - \big(p^{2m} - 1\big)\big](p^m + 1) \\ &= \big(p^{2m} - p^m + 1\big)(p^m + 2) - \big(p^{2m} - 1\big)(p^m + 1), \end{aligned}$$

$\text{ord}(\alpha_j)|(p^{2m} - 1)$, contradicting the fact that $\alpha_j \notin \mathbb{F}_{p^{2m}}$. Therefore $\text{ord}(\alpha_j) > 3$ for $j = 0, 1, 2$.

Conversely, assume that $\text{ord}(\alpha_j)|(p^{2m} - p^m + 1)$ and $\text{ord}(\alpha_j) > 3$ for $j = 0, 1, 2$. Suppose that $\alpha_j \in \mathbb{F}_{p^{2m}}$, i.e. $\text{ord}(\alpha_j)|(p^{2m} - 1)$. Then it follows from $p^{2m} - p^m + 1 = (p^{2m} - 1) - (p^m - 2)$ that $\text{ord}(\alpha_j)|(p^m - 2)$ and from $p^{2m} - p^m + 1 = (p^m - 2)(p^m + 1) + 3$ that $\text{ord}(\alpha_j)|3$, contradicting $\text{ord}(\alpha_j) > 3$. Therefore α_0, α_1 and α_2 are not in $\mathbb{F}_{p^{2m}}$ and $f(x)$ is irreducible over $\mathbb{F}_{p^{2m}}$.

□

The next lemma shows that it is quite efficient to find an irreducible polynomial of the required form by using a random search.

Lemma 4.11. [13, Lemma 3.2.1] *For a randomly selected $c \in \mathbb{F}_{p^{2m}}$ the probability that*

$$f(x) = x^3 - cx^2 + c^{p^m}x - 1 \in \mathbb{F}_{p^{2m}}[x]$$

is irreducible, is approximately one third for large enough m and p.

Proof. From Lemma 4.10 it is seen that a polynomial of the required form is irreducible if and only if the order of the roots divide $p^{2m} - p^m + 1$ and are greater than three. Let $\beta \in \mathbb{F}^*_{p^{6m}}$ such that $\text{ord}(\beta) = p^{2m} - p^m + 1$. The existence of β follows from the fact that $\phi_{6m}(p)|(p^{6m} - 1)$. For any $\alpha \in <\beta>$ with $\text{ord}(\alpha) > 3$ the irreducible polynomial with roots $\alpha, \alpha^{p^{2m}}$ and $\alpha^{p^{4m}}$ is of the form (4.7), see Lemma 4.6. Since $\text{ord}(\beta)$ is odd, the only possible orders of α smaller than 4 is 1 or 3. Now the number of elements $\alpha \in <\beta>$ with

$\text{ord}(\alpha) > 3$ is

$$\begin{aligned}&\left(p^{2m} - p^m + 1\right) - a' \text{ with } a' \in \{1, 3\}\\ &= p^{2m} - p^m - a \text{ with } a \in \{0, 2\}.\end{aligned}$$

Since the roots of an irreducible polynomial are conjugates, there are $(p^{2m} - p^m - a)/3$ different irreducible polynomials $f(x)$. Thus the probability that $f(x)$ is irreducible is

$$\begin{aligned}\frac{(p^{2m} - p^m - a)/3}{p^{2m} - 1} &= \frac{1}{3} \cdot \frac{p^{2m} - p^m - a}{p^{2m} - 1}\\ &= \frac{1}{3}\left(\frac{p^{2m} - 1}{p^{2m} - 1} - \frac{p^m + 1 - a}{p^{2m} - 1}\right)\\ &= \frac{1}{3}\left(1 - \frac{1}{p^m - 1} - \frac{a}{p^{2m} - 1}\right)\\ &\approx \frac{1}{3}.\end{aligned}$$

□

The following lemma gives a condition for the irreducibility of a polynomial in terms of the characteristic sequence of the polynomial. The use of a characteristic sequence gives an efficient test for irreducibility.

Lemma 4.12. [13, Lemma 2.3.4 part (iii)] *Let $f(x) = x^3 + cx^2 - c^{p^m}x - 1 \in \mathbb{F}_{p^{2m}}[x]$ be a polynomial with roots $\alpha_0, \alpha_1, \alpha_2$ and let (s_k) be the characteristic sequence of $f(x)$. Then $f(x)$ is irreducible over $\mathbb{F}_{p^{2m}}$ if and only if $s_{p^m+1} \notin \mathbb{F}_{p^m}$.*

Proof. If $f(x)$ is reducible then all α_j are in $\mathbb{F}_{p^{2m}}$. If $f(x)$ is reducible it follows that $\alpha_j^{(p^m+1)p^m} = \alpha_j^{p^{2m}-1}\alpha_j^{p^m+1} = \alpha_j^{p^m+1}$ and $\alpha_j^{p^m+1} \in \mathbb{F}_{p^m}$ for $j = 0, 1, 2$ so that $s_{p^m+1} \in \mathbb{F}_{p^m}$, since $s_{p^m+1} = \alpha_0^{p^m+1} + \alpha_1^{p^m+1} + \alpha_2^{p^m+1}$.

Conversely, if $s_{p^m+1} \in \mathbb{F}_{p^m}$, then $s_{p^m+1}^{p^m} = s_{p^m+1}$ and $f_{s_{p^m+1}}(x) = x^3 - s_{p^m+1}x^2 + s_{p^m+1}x - 1$, see (3.3). Thus $f_{s_{p^m+1}}(1) = 0$. Because the roots of $f_{s_{p^m+1}}(x)$ are the $(p^m+1)^{th}$ powers of the roots of $f(x)$, it follows that $f(x)$ has roots of order dividing $p^m + 1$, i.e. an element of $\mathbb{F}_{p^{2m}}$. Thus $f(x)$ is reducible over $\mathbb{F}_{p^{2m}}$. □

The EXTR system requires an irreducible polynomial of prime order q and of the form (4.7). Algorithm 9 randomly selects such a polynomial.

Algorithm 9: Computation of $f(x)$

Data: Primes p and q, a positive integer m, $q \nmid 6m$ and $q|\phi_{6m}(p)$
Output: $f(x) = x^3 - cx^2 + c^{p^m}x - 1 \in \mathbb{F}_{p^{2m}}[x]$, irreducible and of order q

1 Pick $a \in \mathbb{F}_{p^{2m}} \setminus \mathbb{F}_{p^m}$ at random;
2 $g(x) \leftarrow x^3 - ax^2 + a^{p^m}x - 1$;
3 $(s_k) \leftarrow$ characteristic sequence of $g(x)$;
4 If $s_{p^m+1} \in \mathbb{F}_{p^m}$, restart;
5 If $s_{(p^{2m}-p^m+1)/q} = 3$, restart;
6 $c \leftarrow s_{(p^{2m}-p^m+1)/q}$;
7 $f(x) \leftarrow x^3 - cx^2 + c^{p^m}x - 1$;

The justification of Algorithm 9 is given next. Let $H = p^{2m} - p^m + 1$ and from the property that $\phi_{6m}(p)|H$ it follows that $q|H$ and let n be the largest integer such that $q^n|H$.

Lines 1 to 4 are used to obtain a random irreducible polynomial $g(x)$. From Lemma 4.12 if follows that $s_{p^m+1} \in \mathbb{F}_{p^m}$ if and only if the polynomial is irreducible. This irreducible polynomial is referred to as $g(x)$ with roots α_0, α_1 and α_2, say. The probability that the selected a will result in an irreducible polynomial is approximately one third, see Lemma 4.11.

Line 5 determines whether the order of $g(x)$ is divisible by q^n. It is shown that if

$$\begin{aligned} s_{H/q} &= \alpha_0^{H/q} + \alpha_1^{H/q} + \alpha_2^{H/q} \\ &\neq 3 \end{aligned}$$

then $\text{ord}\left(\alpha_0^{H/q}\right) = q$. Suppose that $\text{ord}\left(\alpha_0^{H/q}\right) \neq q$ then $\alpha_0^{H/q} = 1$ and therefore $s_{H/q} = 3$ is a contradiction. Note that since $g(x)$ is irreducible, the orders of all its roots are the same.

Lines 6 and 7 give the irreducible output polynomial $f(x)$ of order q. It follows from Lemma 4.10 that if a root of $f(x)$ has order q the polynomial $f(x)$ is irreducible. From line 6 it follows that $c_{H/q} = \alpha_0^{H/q} + \alpha_1^{H/q} + \alpha_2^{H/q}$,

defining $\alpha_0^{H/q}, \alpha_1^{H/q}$ and $\alpha_2^{H/q}$ as the roots of $f(x)$ where the order of the roots is q. Thus the order of $f(x)$ is q.

For the remainder of this section, the probability that the algorithm will terminate is discussed. Both lines 4 and 5 determine when the algorithm terminates. Line 4 is covered by Lemma 4.11 and line 5 is discussed next.

Let C_{q^n} and C_{H/q^n} be cyclic groups of order q^n and H/q^n respectively. Then the group $G = C_{q^n} \times C_{H/q^n}$ is a cyclic group of order H. An element β in G has the form (β_1, β_2), where $\beta_1 \in C_{q^n}$ and $\beta_2 \in C_{H/q^n}$. Let β be a root of $g(x)$. An element of order q is constructed if the H/q^{th} power of β is not 1, that is if $q^n | \text{ord}(\beta)$. The number of elements in G that satisfy this constraint is

$$\begin{aligned}(|C_{q^n}| - |C_{q^{n-1}}|) \cdot |C_{H/q^n}| &= \left(\frac{H}{q^n}\right)(q^n - q^{n-1}) \\ &= H - H/q.\end{aligned}$$

From the proof of Lemma 4.11 it follows that the number of elements in G that are possible roots of $g(x)$ is $p^{2m} - p^m - d$ with $d \in \{0, 2\}$. Thus the probability that $q^n | \text{ord}(g)$ is

$$\begin{aligned}\frac{H - H/q}{p^{2m} - p^m - b} &= \frac{(p^{2m} - p^m - 1) - (p^{2m} - p^m - 1)/q}{p^{2m} - p^m - b} \\ &\approx \frac{(p^{2m} - p^m - 1) - (p^{2m} - p^m - 1)/q}{p^{2m} - p^m - 1} \\ &= 1 - 1/q.\end{aligned}$$

For $q \approx 2^{160}$, the influence of line 5 can be ignored.[1]

Computation of $\phi_{6m}(p)$

The computation of the cyclotomic polynomial $\phi_{6m}(x)$ is important for the implementation of EXTR. It is an advantage to find an easy computable formula for $\phi_{6m}(p)$ for the implementation. From this formulation it will also follow that $\phi_{6m}(p) | (p^{2m} - p^m + 1)$.

[1] The recommended subgroup size needed for security by Lenstra in [14].

Three different cases are considered.

(1) m is prime and $m \neq 2, 3$,

(2) m has the form $2^j 3^i$ where i and j are non-negative integers and

(3) m has the form $2^j 3^i m'$ where i, j and m' are non-negative integers with $m' > 1$.

In which follows the formula

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

is repeatedly used [15, Theorem 2.45].

Let m be a prime number not equal to 2 or 3.

$$\begin{aligned}
\phi_{6m}(x) &= \frac{x^{6m}-1}{\prod_{\substack{d<6m \\ d|6m}} \phi_d(x)} \\
&= \frac{x^{6m}-1}{[\phi_1(x)\phi_3(x)\phi_m(x)\phi_{3m}(x)]\phi_2(x)\phi_6(x)\phi_{2m}(x)} \\
&= \frac{x^{6m}-1}{(x^{3m}-1)\left(\dfrac{\phi_1(x)\phi_2(x)\phi_m(x)\phi_{2m}(x)}{\phi_1(x)\phi_m(x)}\right)\phi_6(x)} \\
&= \frac{(x^{3m}-1)(x^{3m}+1)[\phi_1(x)\phi_m(x)]}{(x^{3m}-1)(x^{2m}-1)\phi_6(x)} \\
&= \frac{(x^{3m}+1)(x^m-1)}{\phi_6(x)(x^m-1)(x^m+1)} \\
&= \frac{(x^{2m}-x^m+1)(x^m-1)}{\phi_6(x)(x^m-1)} \\
&= \frac{x^{2m}-x^m+1}{x^2-x+1}.
\end{aligned}$$

In the following two cases, set notation is used to determine distinct sets of which the union is $\{d \in \mathbb{Z}^+ : d|6m,\ d \neq 6m\}$.

Let m be of the form $m = 2^j 3^i$, where i and j are non-negative. Now

$$\begin{aligned}&\left\{d \in \mathbb{Z}^+ : d|2^{j+1}3^{i+1},\ d \neq 2^{j+1}3^{i+1}\right\}\\ =&\left\{d : d|2^j 3^i\right\} \cup \left\{2^{j+1}d : d|3^i\right\} \cup \left\{3^{i+1}d : d|2^j\right\},\end{aligned}$$

where the last two sets in the union can be written as

$$\begin{aligned}\left\{2^{j+1}d : d|3^i\right\} &= \left\{d : d|2^{j+1}3^i,\ 2^{j+1}|d\right\}\\ &= \left\{d : d|2^{j+1}3^i\right\} \setminus \left\{d : d|2^j 3^i\right\}\end{aligned}$$

and

$$\begin{aligned}\left\{3^{i+1}d : d|2^j\right\} &= \left\{d : d|2^j 3^{i+1},\ 3^{i+1}|d\right\}\\ &= \left\{d : d|2^j 3^{i+1}\right\} \setminus \left\{d : d|2^j 3^i\right\}.\end{aligned}$$

This partition is used to compute the cyclotomic polynomial.

$$\begin{aligned}\phi_{6m}(x) &= \frac{x^{6m}-1}{\prod_{\substack{d|2^{j+1}3^{i+1}\\ d\neq 2^{j+1}3^{i+1}}} \phi_d(x)}\\ &= \frac{x^{6m}-1}{\prod_{d|2^j 3^i} \phi_d(x)} \frac{\prod_{d|2^j 3^i} \phi_d(x)}{\prod_{d|2^{j+1}3^i} \phi_d(x)} \frac{\prod_{d|2^j 3^i} \phi_d(x)}{\prod_{d|2^j 3^{i+1}} \phi_d(x)}\\ &= \frac{(x^{6m}-1)(x^m-1)(x^m-1)}{(x^m-1)(x^{2m}-1)(x^{3m}-1)}\\ &= \frac{(x^{3m}+1)(x^m-1)}{(x^{2m}-1)}\\ &= \frac{x^{3m}+1}{x^m+1}\\ &= x^{2m}-x^m+1.\end{aligned}$$

Lastly, let m be of the form $m = 2^j 3^i m'$ where i, j and m' are non-negative integers and $m' > 1$. The same technique as above is used to partition the

divisors into different sets. Now

$$\begin{aligned}&\left\{d \in \mathbb{Z}^* : d|2^{j+1}3^{i+1}m', d \neq 2^{j+1}3^{i+1}m'\right\}\\ =&\left\{d : d|2^j3^im'\right\} \cup \left\{2^{j+1}d : d|3^im'\right\} \cup \left\{3^{i+1}d : d|2^jm'\right\}\\ &\cup \left\{2^{j+1}3^{i+1}d : d|m', d \neq m'\right\}\end{aligned}$$

where the second and third sets can be written as

$$\begin{aligned}\left\{2^{j+1}d :\ d|3^im'\right\} &= \left\{d :\ d|2^{j+1}3^im',\ 2^{j+1}|d\right\}\\ &= \left\{d :\ d|2^{j+1}3^im'\right\} \setminus \left\{d :\ d|2^j3^im'\right\}\end{aligned}$$

and

$$\begin{aligned}\left\{3^{i+1}d :\ d|2^jm'\right\} &= \left\{d :\ d|3^{i+1}2^jm',\ 3^{i+1}|d\right\}\\ &= \left\{d :\ d|3^{i+1}2^jm'\right\} \setminus \left\{d :\ d|2^j3^im'\right\}.\end{aligned}$$

This partition is used to compute the cyclotomic polynomial.

$$\begin{aligned}\phi_{6m}(x) &= \phi_{2^{j+1}3^{i+1}m'}(x)\\ &= \frac{x^{6m}-1}{\prod_{\substack{d|2^{j+1}3^{i+1}m'\\ d\neq 2^{j+1}3^{i+1}m'}} \phi_d(x)}\\ &= \frac{x^{6m}-1}{\prod_{d|2^j3^im'} \phi_d(x) \prod_{\substack{d|m'\\ d\neq m'}} \phi_{2^{j+1}3^{i+1}m'}(x)} \frac{\prod_{d|2^j3^im'} \phi_d(x)}{x^{2m}-1} \frac{\prod_{d|2^j3^im'} \phi_d(x)}{x^{3m}-1}\\ &= \frac{(x^{6m}-1)(x^m-1)(x^m-1)}{(x^m-1)(x^{2m}-1)(x^{3m}-1)} \left(\frac{1}{\prod_{\substack{d|m'\\ d\neq m'}} \phi_{2^{j+1}3^{i+1}m'}(x)}\right)\\ &= \frac{x^{3m}+1}{x^m+1} \left(\frac{1}{\prod_{\substack{d|m'\\ d\neq m'}} \phi_{2^{j+1}3^{i+1}m'}(x)}\right)\\ &= \frac{x^{2m}-x^m+1}{\prod_{\substack{d|m'\\ d\neq m'}} \phi_{2^{j+1}3^{i+1}m'}(x)}.\end{aligned}$$

From these results the next lemma is obtained.

Lemma 4.13. *The $6m^{th}$ cyclotomic polynomial $\phi_{6m}(x)$ over the field $\mathbb{F}_p$ divides the polynomial $x^{2m} - x^m + 1$.*

4.3.3 Computations

Attention is given to computation in a finite field defined by an optimal normal basis and to the computation of a specific element in a linear shift register. An optimal normal basis is used for the construction of the field $\mathbb{F}_{p^{2m}}$ over which the EXTR polynomials are defined. The sequence computations are specific to the shift register used in the EXTR protocol.

Optimal Normal basis Computations

The next lemma gives the number of multiplications in $\mathbb{F}_p$ needed to compute an operation in $\mathbb{F}_{p^{2m}}$. The number of additions needed is also given. Addition is normally not added in the complexity analysis, but the addition count is used in the construction of the computational models in the next chapter.

Lemma 4.14. [16, Lemma 5] *Let p and $2m+1$ be prime numbers, where p is a primitive element in $\mathbb{Z}^*_{2m+1}$. Let $x, y, z \in \mathbb{F}_{p^{2m}}$.*

(1) Computing $x + y$ takes $2m$ additions in $\mathbb{F}_p$.

(2) Computing x^p is for free.

(3) Computing xy takes $4m^2$ multiplications and $4m^2 - 2m$ additions in $\mathbb{F}_p$.

(4) Computing $xz - yz^{p^m}$ takes $4m^2$ multiplications and $8m^2 - 4m$ additions in $\mathbb{F}_p$.

Proof. Let $x, y, z, \alpha \in \mathbb{F}_{p^{2m}}$ such that $\{\sigma(\alpha) : \sigma \in G\}$ is an optimal normal

basis with

$$x = \sum_{i=1}^{2m} x_i \sigma_i(\alpha)$$
$$y = \sum_{i=1}^{2m} y_i \sigma_i(\alpha)$$
$$z = \sum_{i=1}^{2m} z_i \sigma_i(\alpha)$$

and $\alpha \in G$ the automorphism which is the index to the all -1 row of the matrix d_α.

(1) An element in $\mathbb{F}_{p^{2m}}$ can be represented by a $2m$ dimensional vector over $\mathbb{F}_p$, from which the result follows.

(2) Represent the element x with a normal basis. Then

$$\begin{aligned} x^p &= \left(\sum_{i=1}^{2m} x_i \sigma_i(\alpha) \right)^p \\ &= \sum_{i=1}^{2m} x_i \sigma_i(\alpha)^p \\ &= \sum_{i=1}^{2m} x_i \sigma_{i+1}(\alpha). \end{aligned}$$

Note that x^p is just a cyclic right shift of the coefficients of x. Also note that $\sigma_{2m+1} = \sigma_1$.

(3) Multiplication of x and y is done with an optimal normal basis. The number of multiplications needed is $4m^2$ in $\mathbb{F}_p$. This is seen from the formula of multiplication given in Section 4.3.2,

$$\begin{aligned} xy &= \sum_{\sigma,\gamma \in G} x_\sigma y_\gamma \sum_{\tau \in G} d_\alpha(\sigma^{-1}\tau, \sigma^{-1}\gamma)\tau(\alpha) \\ &= \sum_{\tau \in G} \left(\sum_{\sigma,\gamma \in G} x_\sigma y_\gamma d_\alpha(\sigma^{-1}\tau, \sigma^{-1}\gamma) \right) \tau(\alpha). \end{aligned}$$

The number of additions needed is determined by fixing the automorphism τ in the above equation and determining when $d_\alpha(\sigma^{-1}\tau, \sigma^{-1}\gamma)$ is non-zero. Let κ index the row consisting of only minus ones, i.e. $\sigma^{-1} = \kappa\gamma^{-1}$. Any other row must contain only a single 1, since the basis is optimal normal. Thus only $2m$ additions are needed to compute the coefficient for a fixed τ. Since the same summation is done for each coefficient, the total number of additions is $2m(2m-1) = 4m^2 - 2m$.

(4) Note that

$$z^{p^m} = \sum_{i=1}^{m} z_i \sigma_{i+m}(\alpha) + \sum_{i=m+1}^{2m} z_i \sigma_{i-m}(\alpha).$$

Now

$$\begin{aligned} xz - yz^{p^m} =& \sum_{i=1}^{2m}\sum_{j=1}^{2m} x_{\sigma_i} z_{\sigma_j} \sum_{\tau\in G} d_\alpha(\sigma_{-i}\tau, \sigma_{-i}\sigma_j)\tau(\alpha) \\ &- \sum_{i=1}^{2m}\sum_{j=1}^{m} y_{\sigma_i} z_{\sigma_j} \sum_{\tau\in G} d_\alpha(\sigma_{-i}\tau, \sigma_{-i}\sigma_{j+m})\tau(\alpha) \\ =& \tau(\alpha)\sum_{\tau\in G}\{\left(\sum_{j=1}^{m} z_{\sigma_j} \sum_{i=1}^{2m} (x_{\sigma_i} d_\alpha(\sigma_{-i}\tau, \sigma_{-i}\sigma_j) - y_{\sigma_i} d_\alpha(\sigma_{-i}\tau, \sigma_{-i}\sigma_{j+m}))\right) \\ &+ \left(\sum_{j=m+1}^{2m} z_{\sigma_j} \sum_{i=1}^{2m} (x_{\sigma_i} d_\alpha(\sigma_{-i}\tau, \sigma_{-i}\sigma_j) - y_{\sigma_i} d_\alpha(\sigma_{-i}\tau, \sigma_{-i}\sigma_{j-m}))\right)\}. \end{aligned}$$

Since the products $\sigma_i(\alpha)\sigma_j(\alpha)$ are 'free', a total of $4m^2$ multiplications in $\mathbb{F}_p$ is needed. The number of additions is determined by the number of non-zero entries in the matrix d_α. Now consider the pairs

$$(d_\alpha(\sigma_{-i}\tau, \sigma_{-i}\sigma_j), d_\alpha(\sigma_{-i}\tau, \sigma_{-i}\sigma_{j-m}))$$

for $i, j = 1, 2, \ldots, 2m$ and τ a fixed automorphism. First consider when the pairs have -1 as the first coordinate. This only happens when $\mathrm{ord}(\sigma_{j-i}) = 2$ which is the case if $2m|2(j-i)$, i.e. $m|(j-i)$ and thus $m|(j-i-m)$ and $\sigma_{j-i} = \sigma_{j-i-m}$. Thus a coordinate is -1 if and only if

both coordinates are -1. This happens $2m$ times. The only other possible cases are $(1,0)$ and $(0,1)$, which happen only $2(m-1)$ times. Thus the number of additions needed to compute the coefficient of τ is $4m-2$. The total number of additions is now given by $2m(4m-2) = 8m^2 - 4m$.

□

Sequence computations

The characteristic sequence (s_i) of $f(x)$ will be used to compute the polynomial $f_n(x)$ for positive integers n. The private key will determine the subsequences (s_{in}) with characteristic polynomial

$$f_n(x) = x^3 - s_n x^2 + s_n^{p^m} x - 1,$$

as the associated public key. Attention is now given to the computation of a specific term in the characteristic sequence of the polynomial $f(x) = x^3 - cx^2 + c^{p^m} x - 1 \in \mathbb{F}_{p^{2m}}[x]$. This computation is needed in Algorithm 9 and in the computation of the shared secret.

The computation of the sequence terms is done in a similar manner as in CFE. The *reciprocal sequence* of (s_k) is generated by the reciprocal of $f(x)$, namely $f^{-1}(x) = x^3 - c^{p^m} x^2 + cx - 1$, with associated linear recurring relation

$$s_k = c^{p^m} s_{k+1} - c s_{k-2} + s_{k+3} \text{ for } k \leq -1$$

and with the same initial values as the characteristic sequence of $f(x)$. It follows that the terms of the reciprocal sequence of $f(x)$ also satisfy the relation (4.9), namely $s_k = \alpha_0^k + \alpha_1^k + \alpha_2^k$, $k \leq -1$.

The characteristic and reciprocal sequences of $f(x)$ are therefore the subsequences $(s_k)_{k\geq 0}$ and $(s_k)_{k\leq -1}$ of the sequence $(s_k)_{-\infty}^{\infty}$, with initial values $s_0 = 3$, $s_1 = c$ and $s_2 = c^2 - 2c^{p^m}$. No distinction will be made between the first two sequences as both subsequences satisfy the same linear relation

$$s_{k+3} = c s_{k+2} - c^{p^m} c_{k+1} - c_k \text{ for } k \in \mathbb{Z}.$$

The following lemmas provide relations that are used in the computation of the sequence terms.

Lemma 4.15. [13, Lemma 2.3.2 part (v)] *The terms of the characteristic and the reciprocal sequences* (s_k) *generated by*

$$f(x) = x^3 - cx^2 + c^{p^m}x - 1 \in \mathbb{F}_{p^{2m}}[x]$$

satisfy the property

$$s_{-k} = s_{p^m k} = s_k^{p^m},\ k \geq 0.$$

Proof. Let α_0, α_1 and α_2 be the roots of the polynomial $f(x)$. Then from Lemma 4.10 it follows that $f(\alpha_i^{-p^m}) = 0$ for all i and therefore the exponent $-p^m$ maps a root of the polynomial to a different root, i.e. in set notation $\left\{\alpha_0^{-p^m}, \alpha_1^{-p^m}, \alpha_2^{-p^m}\right\} = \{\alpha_0, \alpha_1, \alpha_2\}$. Now from (4.9)

$$\begin{aligned} s_{-k} &= \alpha_0^{-k} + \alpha_1^{-k} + \alpha_2^{-k} \\ &= \alpha_0^{p^m k} + \alpha_1^{p^m k} + \alpha_2^{p^m k} \\ &= s_{p^m k} \\ &= s_k^{p^m}. \end{aligned}$$

□

It follows from the above lemma that in EXTR the reciprocal sequence is not needed in the computation of s_k, as is the case in CFE. In the sequel the sequence $(s_k)_{-\infty}^{\infty}$ will be referred to as the characteristic sequence of $f(x)$.

Lemma 4.16. [13, Lemma 2.3.4] *The relation*

$$s_{u+v} = s_u s_v - s_v^{p^m} s_{u-v} + s_{u-2v},\ u, v \in \mathbb{Z}$$

holds for the characteristic sequence (s_k) *of the polynomial*

$$x^3 - cx^2 + c^{p^m}x - 1 \in \mathbb{F}_{p^{2m}}[x].$$

Through substitution it follows that

(1) $s_{2n} = s_n^2 - 2s_n^{p^m}$

(2) $s_{2n+1} = s_n s_{n+1} - cs_n^{p^m} + s_{n-1}^{p^m}$

(3) $s_{2n-1} = s_n s_{n-1} - c^{p^m} s_n^{p^m} + s_{n+1}^{p^m}$.

Proof. The relation

$$s_{u+v} = s_u s_v - s_v^{p^m} s_{u-v} + s_{u-2v}$$

is obtained by the following computations:

$$\begin{aligned}
s_{u+v} =& \alpha_0^{u+v} + \alpha_1^{u+v} + \alpha_2^{u+v},\\
s_u s_v =& (\alpha_0^u + \alpha_1^u + \alpha_2^u)(\alpha_0^v + \alpha_1^v + \alpha_2^v)\\
=& \alpha_0^{u+v} + \alpha_0^u\alpha_1^v + \alpha_0^u\alpha_2^v + \alpha_1^u\alpha_0^v + \alpha_1^{u+v} + \alpha_1^u\alpha_2^v + \alpha_2^u\alpha_1^v\\
&+ \alpha_2^u\alpha_1^v + \alpha_2^{u+v}\\
=& s_{u+v} + \alpha_0^u\alpha_1^v + \alpha_0^u\alpha_2^v + \alpha_1^u\alpha_0^v + \alpha_1^u\alpha_2^v + \alpha_2^u\alpha_1^v + \alpha_2^u\alpha_1^v,\\
s_v^{p^m} s_{u-v} =& s_{-v}s_{u-v} \text{ (Lemma 4.15)}\\
=& \left(\alpha_0^{-v} + \alpha_1^{-v} + \alpha_2^{-v}\right)\left(\alpha_0^{u-v} + \alpha_1^{u-v} + \alpha_2^{u-v}\right)\\
=& \alpha_0^{u-2v} + \alpha_0^{-v}\alpha_1^{u-v} + \alpha_0^{-v}\alpha_2^{u-v} + \alpha_1^{-v}\alpha_0^{u-v} + \alpha_1^{u-2v} + \alpha_1^{-v}\alpha_2^{u-v}\\
&+ \alpha_2^{-v}\alpha_0^{u-v} + \alpha_2^{-v}\alpha_1^{u-v} + \alpha_2^{u-2v}\\
=& s_{u-2v} + \alpha_0^{-v}\alpha_1^{u-v} + \alpha_0^{-v}\alpha_2^{u-v} + \alpha_1^{-v}\alpha_0^{u-v} + \alpha_1^{-v}\alpha_2^{u-v}\\
&+ \alpha_2^{-v}\alpha_0^{u-v} + \alpha_2^{-v}\alpha_1^{u-v}\\
=& s_{u-2v} + \alpha_2^v\alpha_1^u + \alpha_1^v\alpha_2^u + \alpha_2^v\alpha_0^u + \alpha_0^v\alpha_2^u + \alpha_1^v\alpha_0^u + \alpha_0^v\alpha_1^u.
\end{aligned}$$

In the last step, the relation $\alpha_0\alpha_1\alpha_2 = 1$ is used. Now

$$\begin{aligned} s_u s_v - s_v^{p^m} s_{u-v} + s_{u-2v} =& s_{u+v} + \alpha_2^v\alpha_1^u + \alpha_1^v\alpha_2^u + \alpha_2^v\alpha_0^u \\ &+ \alpha_0^v\alpha_2^u + \alpha_1^v\alpha_0^u + \alpha_0^v\alpha_1^u \\ &- (\alpha_2^v\alpha_1^u + \alpha_1^v\alpha_2^u + \alpha_2^v\alpha_0^u + \alpha_0^v\alpha_2^u + \alpha_1^v\alpha_0^u + \alpha_0^v\alpha_1^u) \\ =& s_{u+v}. \end{aligned}$$

Using the above relation, Lemma 4.15 and substitution, the required numbered relations are obtained by using the fact that $s_1 = c$.

(1) The substitutions $v = n$ and $u = n$ give

$$\begin{aligned} s_{2n} &= s_n^2 - s_0 s_n^{p^m} + s_{n-2n} \\ &= s_n^2 - 3s_n^{p^m} + s_{-n} \\ &= s_n^2 - 3s_n^{p^m} + s_n^{p^m} \\ &= s_n^2 - 2s_n^{p^m}. \end{aligned}$$

(2) The substitutions $v = n$ and $u = n + 1$ give

$$\begin{aligned} s_{2n+1} &= s_n s_{n+1} - s_n^{p^m} s_1 + s_{n+1-2n} \\ &= s_n s_{n+1} - c s_n^{p^m} + s_{1-n} \\ &= s_n s_{n+1} - c s_n^{p^m} + s_{n-1}^{p^m}. \end{aligned}$$

(3) The substitutions $v = n$ and $u = n - 1$ give

$$\begin{aligned} s_{2n-1} &= s_n s_{n-1} - s_n^{p^m} s_{-1} + s_{n-1-2n} \\ &= s_n s_{n-1} - s_1^{p^m} s_n^{p^m} + s_{-n-1} \\ &= s_n s_{n-1} - c^{p^m} s_n^{p^m} + s_{n+1}^{p^m}. \end{aligned}$$

□

The term s_n in the characteristic sequence is computed as discussed in

Section 3.2. The index n is written in its binary form

$$n = \sum_{j=0}^{r} n_j 2^{r-j}. \tag{4.17}$$

The index can be computed by using the linear relation $t_j = n_j + 2t_{j-1}$, where $j \geq 0$ and $t_0 = n_0 \neq 0$. The value n is given by t_r.

Two functions F_{n_j} with $n_j \in \{0, 1\}$ are needed in the computation of s_n. These functions are defined, using Lemma 4.16, such that they map $\left(s_{t_j-1}, s_{t_j}, s_{t_j+1}\right)$ to $\left(s_{t_{j+1}-1}, s_{t_{j+1}}, s_{t_{j+1}+1}\right)$. The functions F_{n_j} are defined as follows:

If $n_{j+1} = 0$ **then** $t_{j+1} = 2t_j$**:**

$$\begin{aligned} s_{t_{j+1}-1} &= s_{t_j} s_{t_j-1} - c^{p^m} s_{t_j}^{p^m} + s_{t_j+1}^{p^m} \\ s_{t_{j+1}} &= s_{t_j}^2 - 2s_{t_j}^{p^m} \\ s_{t_{j+1}+1} &= s_{t_j} s_{t_j+1} - c s_{t_j}^{p^m} + s_{t_j-1}^{p^m}. \end{aligned}$$

If $n_{j+1} = 1$ **then** $t_{j+1} = 2t_j + 1$**:**

$$\begin{aligned} s_{t_{j+1}-1} &= s_{t_j}^2 - 2s_{t_j}^{p^m} \\ s_{t_{j+1}} &= s_{t_j} s_{t_j+1} - c s_{t_j}^{p^m} + s_{t_j-1}^{p^m} \\ s_{t_{j+1}+1} &= s_{t_j+1}^2 - 2s_{t_j+1}^{p^m}. \end{aligned}$$

Note that the above functions can be computed using less multiplications if the form $xz - yz^{p^m}$ is used.

These functions are used in Algorithm 10 to compute the n^{th} term in the characteristic sequence.

Algorithm 10 is based on the repeated squaring algorithm, where F_0 corresponds to squaring while F_1 corresponds to squaring and multiplication. The value of M starts with 3 as its center element and the index of this element is 0. In the 'for loop' the binary expansion of n is scanned. The bit scanned determines whether F_0 or F_1 is executed. The index of the center element of M will thus satisfy the relation $t_j = n_j + 2t_{j-1}$. After the execution

Algorithm 10: Computation of (s_{n-1}, s_n, s_{n+1})

Data: The index $n = (n_r, n_{r-1}, \ldots, n_0)_2$ and c defining the characteristic sequence (s_n)

Output: (s_{n-1}, s_n, s_{n+1})

$M \leftarrow \left(c^{p^m}, 3, c\right)$;
for $i = 0$ *to* r **do**
    $M \leftarrow F_{n_{r-i}}(M)$;
end
$(s_{n-1}, s_n, s_{n+1}) \leftarrow M$;

of the 'for loop' the value of t_r is n as required.

Lemma 4.17. *Let $f(x)$ be a system polynomial, as defined in (4.7) and (s_k) the associated characteristic sequence. The n^{th} term in the sequence can be computed with*

$$16m^2 \log_2 n + 4m^2 H_n$$

multiplications and

$$\left(72m^2 - 36m + 7\right) \log_2 n - \left(8m^2 - 4m + 1\right) H_n$$

additions in $\mathbb{F}_p$, where H_n is the Hamming weight of $\lceil n \rceil$.

Proof. In Lemma 4.14 the number of operations in $\mathbb{F}_p$ needed to compute the functions F_{n_i} in $\mathbb{F}_{p^{2m}}$ is given. For the function F_0 the number of multiplications in $\mathbb{F}_p$ is $16m^2$ and the number of additions in $\mathbb{F}_p$ is $32m^2 - 16m + 3$. For the function F_1 the number of multiplications in $\mathbb{F}_p$ is $20m^2$ and the number of additions in $\mathbb{F}_p$ is $40m^2 - 20m + 4$. Therefore the total number of multiplications in $\mathbb{F}_p$ is

$$\begin{aligned} &H_n\left(20m^2\right) + \left(\log_2 n - H_n\right)\left(16m^2\right) \\ &= 16m^2 \log_2 n + 4m^2 H_n \end{aligned}$$

and the total number of additions in $\mathbb{F}_p$ is

$$\begin{aligned} &H_n\big(40m^2 - 20m + 4\big) + (\log_2 n - H_n)\big(32m^2 - 16m + 3\big) \\ &= \big(72m^2 - 36m + 7\big)\log_2 n - \big(8m^2 - 4m + 1\big)H_n. \end{aligned}$$

□

Example 4.18 (EXTR)**.** In this example a characteristic of 101 is used and $m = 1$. Thus the original XTR system is given here. Computations of the shift register are computed in $\mathbb{F}_{p^2} \cong \mathbb{F}_p[x]/<g(x)>$, where $g(x) = x^2 + x + 1$. The basis used for $\mathbb{F}_{p^2}$ is an optimal normal basis given by $\{\alpha, \alpha^2\}$ where α is a root of $g(x)$.

The system polynomial is selected as

$$f(x) = x^3 - \big(21\alpha + 7\alpha^2\big)x^2 + \big(7\alpha + 21\alpha^2\big)x - 1$$

and is of order 103. Let the private keys of Alice and Bob be $priv_A = 51$ and $priv_B = 3$. The associated public keys are $pub_A = 60\alpha + 52\alpha^2$ and $pub_B = 77\alpha + 64\alpha^2$, as computed by Algorithm 10. For Alice to compute the shared secret, pub_B and $priv_A$ are provided as input for Algorithm 10. The triples computed for each iteration of the 'for loop' in Algorithm 10 are given in Table 4.2 below. From the table it is seen that the shared secret is $17\alpha + 79\alpha^2$.

Iteration	k_i	t_i	s_{t_i-1}	s_{t_i}	s_{t_i+1}
1	–	0	$64\alpha + 77\alpha^2$	$98\alpha + 98\alpha^2$	$77\alpha + 64\alpha^2$
2	1	1	$98\alpha + 98\alpha^2$	$77\alpha + 64\alpha^2$	$71\alpha + 60\alpha^2$
3	1	2	$71\alpha + 60\alpha^2$	$73\alpha + 84\alpha^2$	$10\alpha + 15\alpha^2$
4	0	5	$28\alpha + 7\alpha^2$	$78\alpha + 90\alpha^2$	$54\alpha + 62\alpha^2$
5	0	11	$7\alpha + 82\alpha^2$	$41\alpha + 69\alpha^2$	$39\alpha + 87\alpha^2$
6	1	22	$76\alpha + 82\alpha^2$	$29\alpha + 32\alpha^2$	$3\alpha + 10\alpha^2$
7	1	45	$13\alpha + 38\alpha^2$	$17\alpha + 79\alpha^2$	$20\alpha + 44\alpha^2$

Table 4.2: Computing the public key of Alice in EXTR

4.3.4 Security

In Section 1.2 three necessary properties of a public key distribution system are given. These properties translate into three problems very similar to those in Section 2.2.

Definition 4.19 (EXTR Discrete Logarithm Problem)**.** Let $G \subset \mathbb{F}_{p^{6m}}^*$ be a finite cyclic group generated by α. The problem of computing from $\beta \in G$ a number a such that $Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}(\alpha^a) = Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}(\beta)$ is called the *EXTR Discrete Logarithm Problem*. Notation: $a = EXTRDL_\alpha(\beta)$.

Definition 4.20 (EXTR Diffie-Hellman Problem)**.** Let $G \subset \mathbb{F}_{p^{6m}}^*$ be a finite cyclic group generated by α. The problem of computing $Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}\left(\alpha^{ab}\right)$ from $Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}(\alpha^a)$ and $Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}\left(\alpha^b\right)$ is called the *EXTR Diffie-Hellman Problem*. Notation:

$$Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}\left(\alpha^{ab}\right) = EXTRDH\left(Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}(\alpha^a), Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}\left(\alpha^b\right)\right)$$

Definition 4.21 (EXTR Diffie-Hellman Decision Problem)**.** Let $G \subset \mathbb{F}_{p^{6m}}^*$ be a cyclic group with generator α. Let α^a,α^b,α^c be chosen independently and randomly in G according to the uniform distribution. Given the triples

$$\left(Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}(\alpha^a), Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}\left(\alpha^b\right), Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}\left(\alpha^{ab}\right)\right)$$

and

$$\left(Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}(\alpha^a), Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}\left(\alpha^b\right), Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}(\alpha^c)\right)$$

in random order, the *EXTR Diffie-Hellman Decision Problem* is to decide, with probability greater than 1/2, which of the triples is the correct EXTR Diffie-Hellman triple. Notation:

$$EXTRDHD\left(Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}(\alpha^a), Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}\left(\alpha^b\right), Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}(\alpha^c)\right)$$

The main difference between the above problems and those defined in Section 2.2, is the incorporation of the trace function. The occurrence of the

trace function is a direct result from the use of polynomials to represent the public key, see Section 3.1.

Since the trace is the sum of conjugates the solution to the first problem is not unique. In fact the problem has three solutions. The additional solutions are a direct consequence of the reduction in the key space. Now it follows that

$$XTRDL(\alpha^a) = XTRDL\left(\alpha^{ap^{2m}}\right)$$

and

$$\begin{aligned} &EXTRDH\left(Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}(\alpha^a), Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}\left(\alpha^b\right)\right) \\ =&EXTRDH\left(Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}\left(\alpha^{ap^{2m}}\right), Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}\left(\alpha^{bp^{4m}}\right)\right). \end{aligned}$$

The fact that the solution is not unique is of no concern and the reduction in the key space is so small that it can be ignored. The theorem given below gives the assurance that there exists no security risk due to the different solutions. Two problems A and B are (a, b)-equivalent, where a and b are positive integers, if A can be solved by invoking a instances of B and if B can be solved by invoking b instances of A.

Theorem 4.22. [13, Theorem 5.2.1] *Let α be a root of order q of an EXTR system polynomial. Then the following equivalences hold:*

(1) The EXTRDL problem is $(1, 1)$-equivalent to the DL problem in $<\alpha>$.

(2) The EXTRDH problem is $(1, 2)$-equivalent to the DH problem in $<\alpha>$.

(3) The EXTRDHD problem is $(3, 2)$-equivalent to the DHD problem in $<\alpha>$.

Proof. Let $a, d, b \in \mathbb{F}_{p^{2m}}$ and y, x, z, w be positive integers and $r(a)$ be any root of the EXTR system polynomial defined by a, i.e.

$$x^3 - ax^2 + a^{p^m}x - 1.$$

Note $r(a)$ can be one of three roots and can be computed in polynomial time with the Scipione del Ferro method.

(1) To compute $DL(\alpha^y)$, let $x = EXTRDL\left(Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}(\alpha^y)\right)$, then $DL(\alpha^y) = x \cdot p^{2mj} \pmod{q}$ for either $j = 0$, $j = 1$ or $j = 2$. Conversely $EXTRDL(a) = DL_\alpha(r(a))$.

(2) To compute $DH(\alpha^x, \alpha^y)$, first compute

$$d_i = EXTRDH\left(Tr_{\mathbb{F}_{p^{6m}}/p^{2m}}\left(\alpha^x \cdot \alpha^i\right), Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}(\alpha^y)\right)$$

for $i = 0, 1$, i.e. $d_i = Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}\left(\alpha^{(x+i)yp^{2mj}}\right)$ for some $j = 0, 1, 2$. Therefore

$$\begin{aligned} r(d_i) &\in \left\{\left(\alpha^{(x+i)y}\right)^{p^{2mj}} : j = 0, 1, 2\right\} \\ &= \left\{\left(DH(\alpha^x, \alpha^y) \cdot \alpha^{yi}\right)^{p^{2mj}} : j = 0, 1, 2\right\}. \end{aligned}$$

To determine $DH(\alpha^x, \alpha^y)$ find two roots $r(d_0)$ and $r(d_1)$ such that $r(d_0) = r(d_1)\alpha^y$ then $DH(\alpha^x, \alpha^y) = r(d_0)$. Conversely

$$EXTRDH(\alpha^x, \alpha^y) = Tr_{p^{6m}/p^{2m}}(DH(\alpha^x, \alpha^y)). \tag{4.18}$$

(3) To solve $DHD(\alpha^x, \alpha^y, \alpha^w)$. Let z be such that

$$\begin{aligned} XTRDH\left(Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}(\alpha^x), Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}(\alpha^y)\right) &= Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}(\alpha^{xy}) \\ &= Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}(\alpha^z) \end{aligned}$$

i.e. there exists an integer i such that $xy = zp^{2mi} \pmod{q}$. Then for some value of $j = 0, 1, 2$

$$\begin{aligned} XTRDH\left(Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}\left(\alpha^{(x+1)}\right), Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}(\alpha^y)\right) &= Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}(\alpha^{xy}\alpha^y) \\ &= Tr_{\mathbb{F}_{p^{6m}}/\mathbb{F}_{p^{2m}}}\left(\alpha^z \alpha^{yp^{2mj}}\right). \end{aligned}$$

Thus $z = xyp^{2mj} \pmod{q}$. The $DHD(\alpha^x, \alpha^y, \alpha^w)$ problem can be solved

by two invocations of $XTRDHD$ by determining whether $w = z$ with $j = 0$. Conversely $XTRDHD(a, b, d)$ can be solved by determining if one of the three roots $r(a)$ satisfies $DHD\left(r(a), r(b), r(d)^{p^{2mj}}\right)$ for some $j = 0, 1, 2$.

□

From this proof it appears that Diffie-Hellman is at least twice as secure as EXTR. This does not mean that the key length of EXTR must be double the length of the keys of Diffie-Hellman, but rather that EXTR could be broken with half the effort it would take to break the Diffie-Hellman Protocol.

Chapter 5

Protocol Implementations

In this chapter it is determined how well the complexity of an algorithm, considering only finite field operations, can predict the execution time of that algorithm. Linear regression [5] is used to determine how good these predictions are. The object is to verify that the theoretical models for the finite field operations, the combining function and the generation of the system parameters in the protocols are a good fit for the empirical data. Using SAS we fit regression curves to simulated data obtained from measuring the execution time of the implemented algorithms. During the linear regression it is assumed that the errors made during measurement are normally distributed and that the models for the finite field operations do not contain any error term.

Only the CFE and XTR protocols are considered in this chapter. This is due to the type of computations that are used in the protocols. The main distinction between DMFB and the other two protocols is that DMFB uses factorisation in the computation of the system parameters and the combining function. Since this dissertation focuses on the use of linear shift registers in the computations the DMFB protocol is not considered further in this chapter. The measurements are available in `./data`, the implementations are available in `./src` and the output of SAS is given in `./stats`, on the attached CD.

The values predicted by the fitted models, the measured data as well as

the standardised error are represented by graphs. The standardised error is computed by

$$\frac{f-d}{RMS},$$

where f is the value predicted by the model, d the measured data and RMS the root mean square. The error $f-d$ is scaled by the RMS, so that the standardised error is follow a standard normal distribution. For the distribution the propability of values between 3 and -3 is 0.9973; hence for a good linear regression almost all the standardised errors should be between these limits.

Another method which is used to determine how good a model is, is to consider a second (test) data set. With the second data set the error must also normally distributed. The standardised error for the second data set is computed by

$$\frac{f-d}{RMS}$$

where f is the fitted model evaluated on the second data set, d is the current value of the second data set and RMS is obtained from the SAS analysis on the first data set. When the standardised error does not give randomly[1] distributed values between 3 and -3 for both data sets, the model is rejected.

5.1 Time Measurement and Data Collection

The measurement of execution time is performed with the assembly instruction 'rdtsc', that gives the number of CPU clock cycles that have passed since start up. By calling the instruction 'rdtsc' twice, the number of clock cycles that have elapsed is given. This method was selected as the other methods known to the author did not provide a time resolution small enough. The time measuremens are performed on an Intel Pentium 4, 3.0 GHz machine running linux and the protocols are compiled with the standard settings of a

[1] Randomness is determine by visual inspection.

g++ compiler. The measurements are used to determine whether the models for execution time are valid in this environment.

To avoid that the factorisation needed in the protocol overshadows the field operations, all factorisation was performed by Mathematica before protocol execution. The numbers that are factored are $\phi_3(p)$ and $\phi_6(p)$ where p is a prime. The primes used in the computations are congruent to 2 (mod 3), and only one prime for each bit length. Considering only primes that are congruent to 2 (mod 3) is a requirement of the XTR protocol.

Two data sets are created during the measurements; the first data set is used to create models, and the second is used to determine how good the models predict another data set. In the first data set two measurements are performed for each prime. The second data set is created by randomly selecting data from a data set where a large number of measurements are performed for each prime. For the protocols the parameters k is randomly selected for each measurement.

5.2 Analysis of Finite Field Operations

The operations used in the protocol that are of interest during complexity analysis are the field operations. The analysis is done only for the fields $\mathbb{F}_p$ and $\mathbb{F}_{p^3}$ in the current section. The complexity for multiplication in the field $\mathbb{F}_{p^n}$ is denoted by M_{p^n} and for addition the complexity is denoted by A_{p^n}. Note the in this section it is only tested that the complexities of addition and multiplication are reasonable by using linear regression.

One other field operation that is also used is the power function. The power function is implemented by the repeated squaring algorithm[2]. The complexity to compute the kth power in the field $\mathbb{F}_{p^n}$ is

$$P_{p^n}(k) = (H_k + \log_2 k)M_{p^n}, \tag{5.1}$$

where H_k is the Hamming weight of k.

[2]An implementation of the repeated squaring method is given in Algorithm 1.

5.2.1 Complexity of Field operations

As noted, the field operations of interest are addition and multiplication in the fields $\mathbb{F}_p$ and $\mathbb{F}_{p^3}$. The prime field's operations are implemented by a big integer library, GMP v4.2.1, and for the computation in the field $\mathbb{F}_{p^3}$ a polynomial basis over $\mathbb{F}_p$ is used.

From [23] it follows that the complexity for addition in $\mathbb{F}_p$ and $\mathbb{F}_{p^3}$ are linear and multiplication is quadratic.

5.2.2 Statistical Analysis

The models for the field operations are computed by SAS and are given in Table 5.1.

	Model	Goodness of fit (R^2)
A_p	$2172.60959 + 1.86646x$	0.1476
M_p	$2169.29748 + 15.53596x + 0.01766x^2$	0.9818
A_{p^3}	$47462 + 22.43205x$	0.3239
M_{p^3}	$492530 + 105.83087x + 0.85509x^2$	0.9168

Table 5.1: SAS output of the field operation analysis, where $x = log_2 p$

The fitted model and measured data for addition in $\mathbb{F}_p$ are given in Figure 5.1. As expected the measured data roughly form two lines. The lower line is where only normal addition is performed and the upper line is where normal addition and subtraction for the modulo operation are performed. Fitting only one line to the data results in a low R^2 value. However any higher order polynomial gives a lower R^2 value.

The errors made by the model with respect to both data sets are given in Figure 5.2. The errors with the second data set are similar to the errors of data set one and this similarity provides confidence in the model.

Figure 5.3 gives the fitted model and the measured data for addition in $\mathbb{F}_{p^3}$. The measured data is mostly linear as expected. The main difference between the behaviour of $\mathbb{F}_p$ and $\mathbb{F}_{p^3}$ in the addition is the "double linearity" that appears in the addition in $\mathbb{F}_p$ does not appear with the addition in the

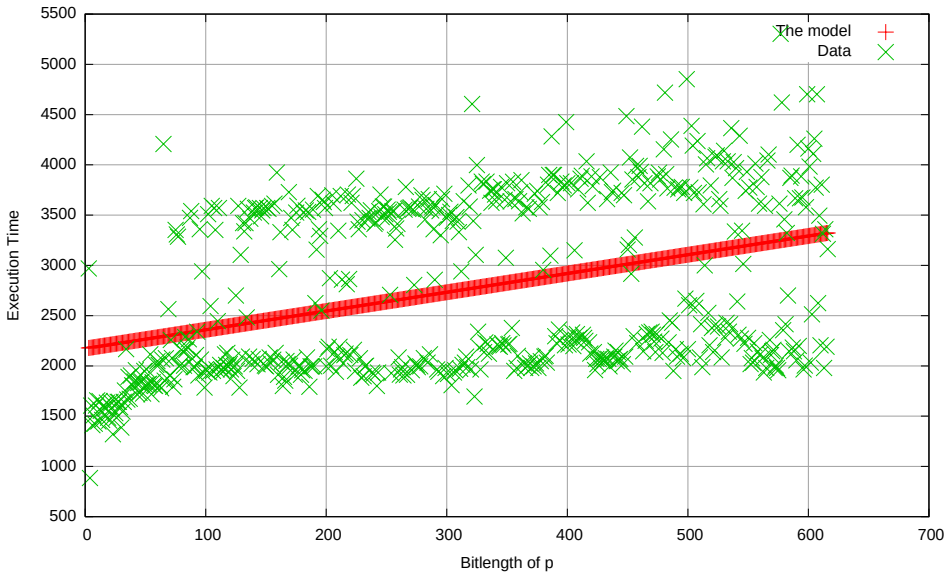

Figure 5.1: Field Addition in the Field $\mathbb{F}_p$.

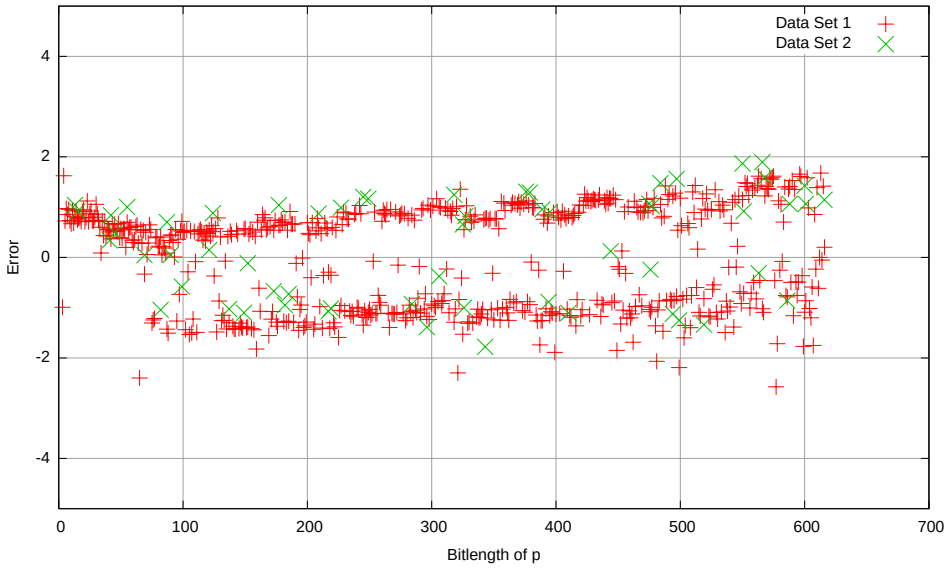

Figure 5.2: Field Addition error in the Field $\mathbb{F}_p$.

field $\mathbb{F}_{p^3}$. This is attributed to the fact that not one but three coefficient in $\mathbb{F}_p$ are added for one addition in $\mathbb{F}_{p^3}$, creating an average between the two lines for addition in $\mathbb{F}_{p^3}$. From the graphs is clear that the overhead is significant for the values considered.

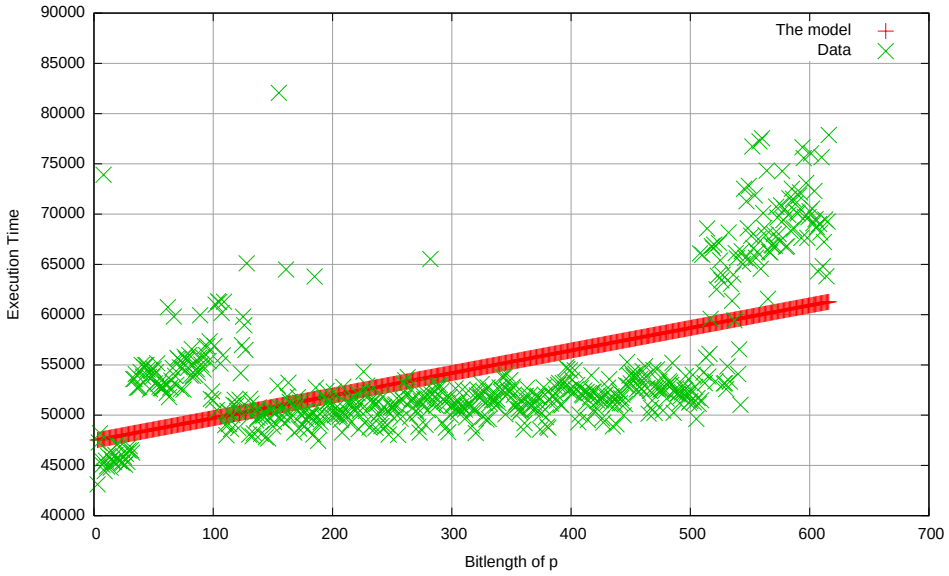

Figure 5.3: Field Addition in the Extension Field $\mathbb{F}_{p^3}$.

Figure 5.4 gives the error made by the model, with both data sets. The errors of both data sets are similar and acceptably small. The "double linearity" in the data of addition in $\mathbb{F}_p$ do not appear in $\mathbb{F}_{p^3}$, since three elements in $\mathbb{F}_p$ are added for addition in $\mathbb{F}_{p^3}$.

Figures 5.5 and 5.6 give the models and measures data for multiplication in $\mathbb{F}_p$ and $\mathbb{F}_{p^3}$ respectively. The data for $\mathbb{F}_p$ is more linear because better optimisations are used in the library GMP than that used for the implementation of $\mathbb{F}_{p^3}$. The jumps in the data are due to the word length of the CPU.

Figure 5.7 and 5.8 give the error of the model for the two data sets. In both figures piecewise lines are created. The piecewise nature of the lines is due to the word length of the CPU. By considering addition and multiplication in the same field it is seen that big jump in the addition graphs does have an influence in the multiplication graphs. This is seen in Figures 5.2; 5.7 and Figures 5.3; 5.8.

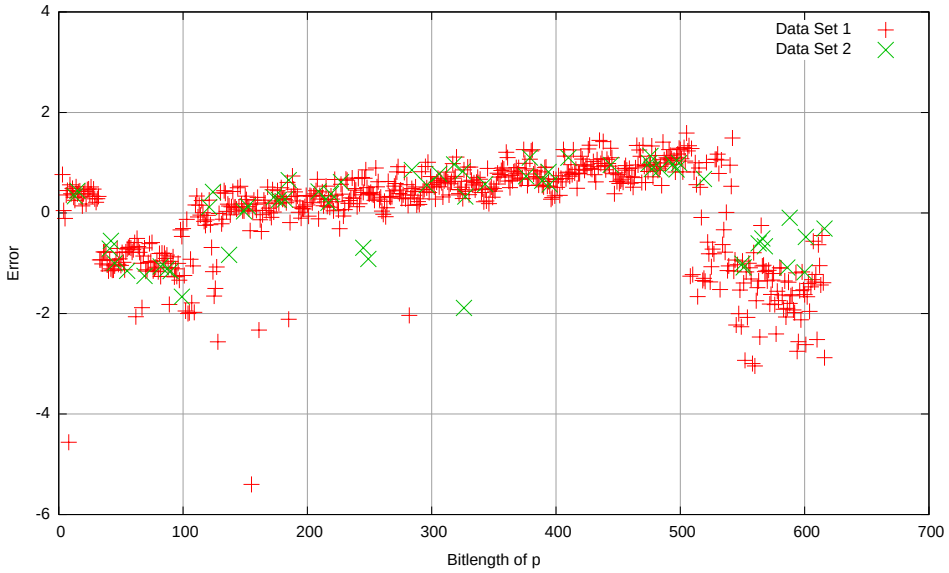

Figure 5.4: Field Addition error in the Extension Field $\mathbb{F}_{p^3}$.

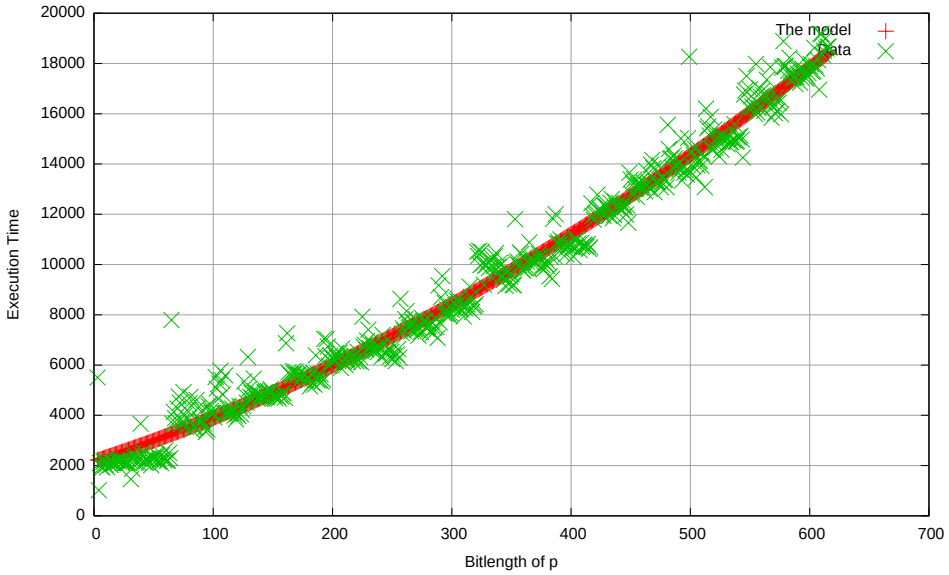

Figure 5.5: Field Multiplication in the Field $\mathbb{F}_p$.

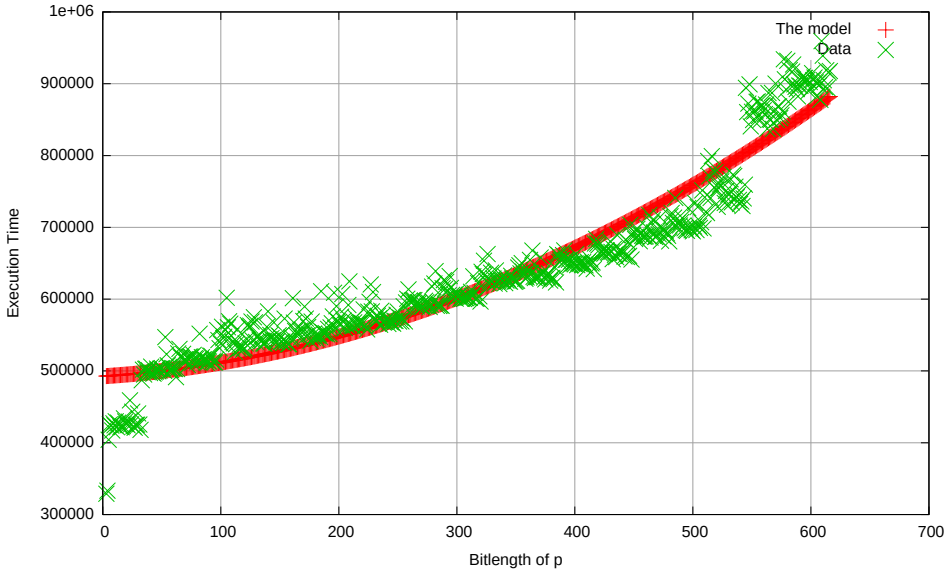

Figure 5.6: Field Multiplication in the Field $\mathbb{F}_{p^3}$.

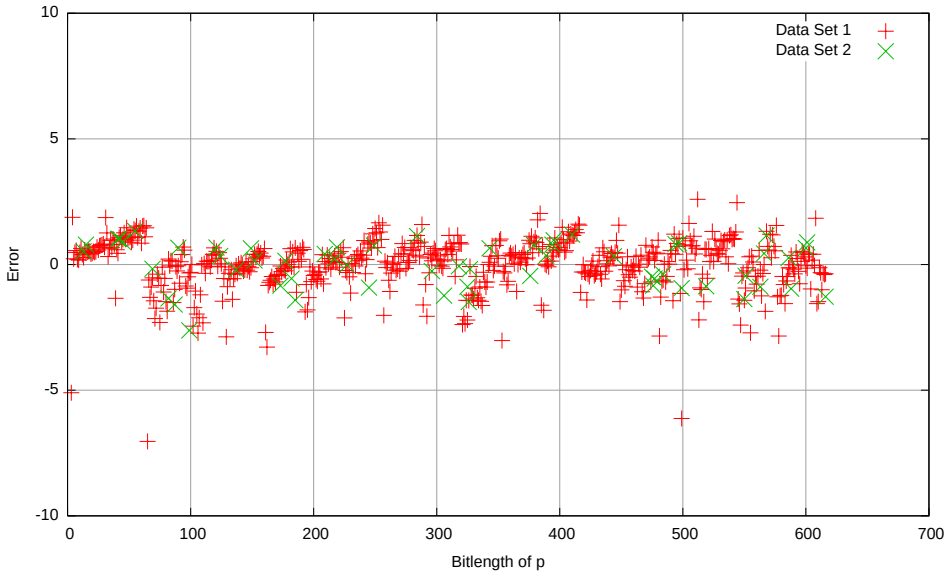

Figure 5.7: Field Multiplication error in the Field $\mathbb{F}_p$.

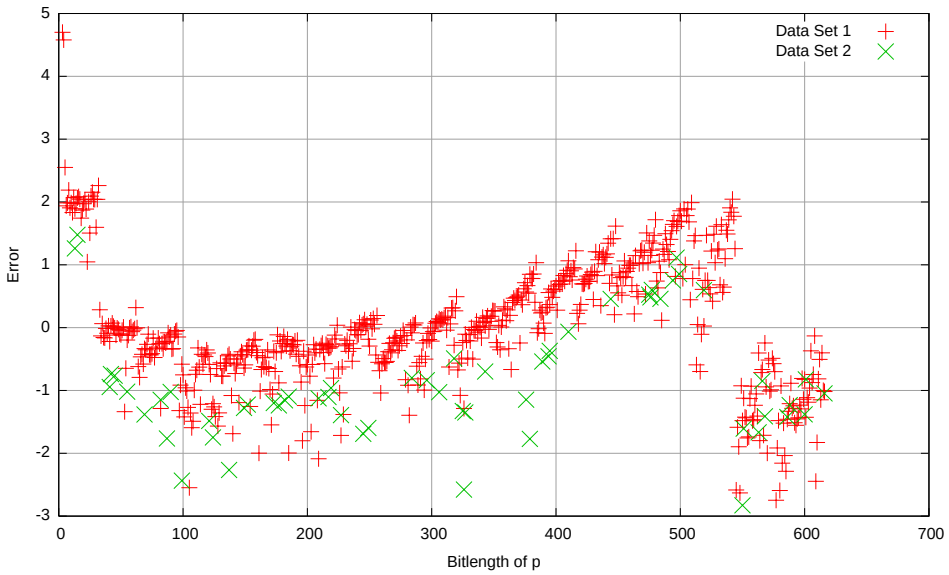

Figure 5.8: Field Multiplication error in the Field $\mathbb{F}_{p^3}$.

5.3 Modeling the CFE protocol

The description of the CFE protocol in [8] does not provide enough information for an implementation. The missing details are mainly in the generation of the system parameters. This was done as the selection of a polynomial of given order was outside the scope of the article. Thus the complexity for the generation of the system parameters given here is more complex than those in the original article. For the combining function the only difference between the analysis done here and that in [8], is the inclusion of the number of additions needed.

5.3.1 Complexity of CFE

System Parameter Generation

Algorithm 11 is used to compute the system parameters, given[3] the characteristic of the field. The algorithm is divided into two parts namely the generation of the field $\mathbb{F}_{p^3}$ and the selection of an irreducible polynomial

[3]The input is different from Algorithm 7, the difference is created for Algorithm 11 to follow the implementation more exactly.

over $\mathbb{F}_p$ of degree 3 with order $\phi_3(p)$. The selected polynomial is the system polynomial for the protocol.

In the construction of $\mathbb{F}_{p^3}$ a random search, lines 2 to 6, is performed for an irreducible polynomial $g(x) \in \mathbb{F}_p[x]$, of degree 3 and order $q = \phi_3(p)$. To determine whether the polynomial $g(x)$ is irreducible, it is tested whether $\gcd(g(x), x^p - x)$ is equal to $g(x)$. Since the Extended Euclidean Algorithm is used, it follows from the following theorem that the complexity to compute the *gcd* is

$$4(6p + dp)(M_p + A_p)$$

where d is some constant.

Theorem 5.1. [23, Theorem 3.11] *The traditional Extended Euclidean Algorithm for polynomials $f, g \in \mathbb{F}_p[x]$ with $deg(f) = n \geq deg(g) = m$ can be performed with:*

(1) at most $m+1$ inversions and $2nm + O(n)$ additions and multiplications in $\mathbb{F}_p$ if only the quotients q_i and remainders r_i are needed.

(2) at most $m+1$ inversions and $6nm + O(n)$ additions and multiplications in $\mathbb{F}_p$ for computing all results.

Corollary 5.2. [15, Corollary 3.21] *The number $N_q(n)$ of monic irreducible polynomials in $\mathbb{F}_q[x]$ of degree n is given by*

$$N_q(n) = \frac{1}{n} \sum_{d|n} \mu(d) q^{n/d},$$

where μ is the Moebius function.

Now it follows from Corollary 5.2 that the expected number of times the Extended Euclidean Algorithm is executed is

$$\begin{aligned} \frac{p^3 - 1}{N_p(3)} &= 3\frac{p^3 - 1}{p^3 - p} \\ &= 3\frac{p^2 + p + 1}{p^2 + p}. \end{aligned}$$

Thus the expected number of operation required to find the field $\mathbb{F}_{p^3}$ is

$$12\left(\frac{(p^2+p+1)(6p+dp)}{p^2+p}\right)(M_p+A_p).$$

Algorithm 11: Compute system parameters for CFE

Data: The prime p.
Output: The system polynomial defined by (a, b).

1 $\phi_3(p) \leftarrow p*p+p+1$;
2 **repeat**
3 $a \in_R \mathbb{F}_p$;
4 $b \in_R \mathbb{F}_p$;
5 $g(x) \leftarrow x^3+ax^2+bx-1$;
6 **until** $gcd\{g(x), x^p-x\} \neq 1$;
7 **while** *true* **do**
8 $f(x) \in_R \{l \in \mathbb{F}_p[x] : deg\{l\} \leq 2\}$;
9 $f(x) \leftarrow f(x)^{(p^3-1)/\phi_3(p)} \pmod{g(x)}$;
10 $found \leftarrow true$;
11 **if** $f(x) \neq 1$ **then**
12 **forall** $e \in \{n \in \mathbb{Z} : n|\phi_3(p),\ and\ n\ is\ prime\}$ **do**
13 **if** $f(x)^{\phi_3(p)/e} \pmod{g(x)} = 1$ **then**
14 $found \leftarrow false$;
15 **break**;
16 **end**
17 **end**
18 **if** $found$ **then**
19 **break**;
20 **end**
21 **end**
22 **end**
23 $r_0 \leftarrow f(x)$;
24 $r_1 \leftarrow r_0^p$;
25 $r_2 \leftarrow r_1^p$;
26 $a \leftarrow -(r_0+r_1+r_2)$;
27 $b \leftarrow r_0 \cdot r_1 + r_0 \cdot r_2 + r_1 \cdot r_2$;

Next a polynomial of order q is created in lines 7 to 22. The method used, is to select an element α of order q in $\mathbb{F}_{p^3}$ and use this element α as a root of

the required polynomial. The algorithm selects an element of order $\beta \in \mathbb{F}_{p^3}$ and tests whether $\beta \in <\alpha>$. It follows from (5.1) that the complexity to perfom the test is

$$P_{p^3}\big((p^3-1)/q\big) = \big(H_{(p^3-1)/q} + \log_2\big((p^3-1)/q\big)\big)M_{p^3}.$$

For an element $\beta \in \mathbb{F}_{p^3}^*$ the order is q if and only if for all prime factors l of q, the element $\beta^{q/l}$ is not the identity element. Thus to test if an element has order q the expected complexity is

$$\frac{C_{primes}}{2}P_{p^3}(C_{len}) = \frac{C_{primes}}{2}(H_{C_{len}} + \log_2(C_{len}))M_{p^3}$$

where C_{primes} is the number of primes dividing q and C_{len} is the average length of the primes dividing q. For a random element $\beta \in \mathbb{F}_{p^3}$, the probability that $\beta^{(p^3-1)/q}$ is equal to a specific element is $1/q$. Since there are $\phi(q)$ generators in $<\alpha>$, the probability that such a generator is selected by β is $\phi(q)/q$. The expected number of selections needed to find a generator is $(p^3-1)\phi(q)/q$. Thus an element of order q can be found in an expected number

$$\frac{\phi(q)(p^3-1)}{q}\left(H_{(p^3-1)/q} + \log_2\big((p^3-1)/q\big) + \frac{C_{primes}}{2}(H_{C_{len}} + \log_2(C_{len}))\right)M_{p^3}$$

of operations.

In lines 23 to 27, the system polynomial's root is given $f(x)$ and the system polynomial is computed.

The complexity to compute the system parameters is given by

$$\begin{aligned}
T_s(p,q) =& 12\left(\frac{(p^2+p+1)(6p+dp)}{p^2+p}\right)(M_p + A_p) + \\
&+ \frac{\phi(q)(p^3-1)}{q}\left(H_{(p^3-1)/q} + \log_2\left(\frac{p^3-1}{q}\right)\right)M_{p^3} \\
&+ \frac{\phi(q)(p^3-1)}{q}\left(\frac{C_{primes}}{2}(H_{C_{len}} + \log_2(C_{len}))\right)M_{p^3} \\
&+ 2P_{p^3} + 3M_{p^3} + 4A_{p^3} \\
=& 12\left(\frac{(p^3+p^2+p)}{p^2+p}(6+d)\right)(M_p + A_p) + \\
&+ \frac{\phi(q)(p^3-1)}{q}\left(H_{(p^3-1)/q} + \log_2\left(\frac{p^3-1}{q}\right)\right)M_{p^3} \\
&+ \frac{\phi(q)(p^3-1)}{q}\left(\frac{C_{primes}}{2}(H_{C_{len}} + \log_2(C_{len}))\right)M_{p^3} \\
&+ (2H_p + 2\log_2 p + 3)M_{p^3} + 4A_{p^3}
\end{aligned}$$

The Combining Function

The combining function is implemented by an algorithm similar to the repeated squaring algorithm. In stead of using multiplication and squaring, two functions F_0 and F_1 are defined. These functions are determined by the characteristic polynomials of the linear shift register. The complexity of the combining function is directly obtained from Lemma 4.4 and is

$$T_c(p,k) = 6(\log_2 k)M_p + (5\log_2 k - H_k)A_p$$

where k is the index of the term in the sequence to be computed.

5.3.2 Statistical Analysis

Linear regression is used in SAS to determine the models for T_s and T_c with data set one. For both models the fitted model and the measured data are plotted. To determine the accuracy of the model's prediction the error made on the second data set is also given.

Combining function

The model for the combining function and the associated data is plotted in Figure 5.9. The jumps in the measured data are a result of the word length of the CPU.

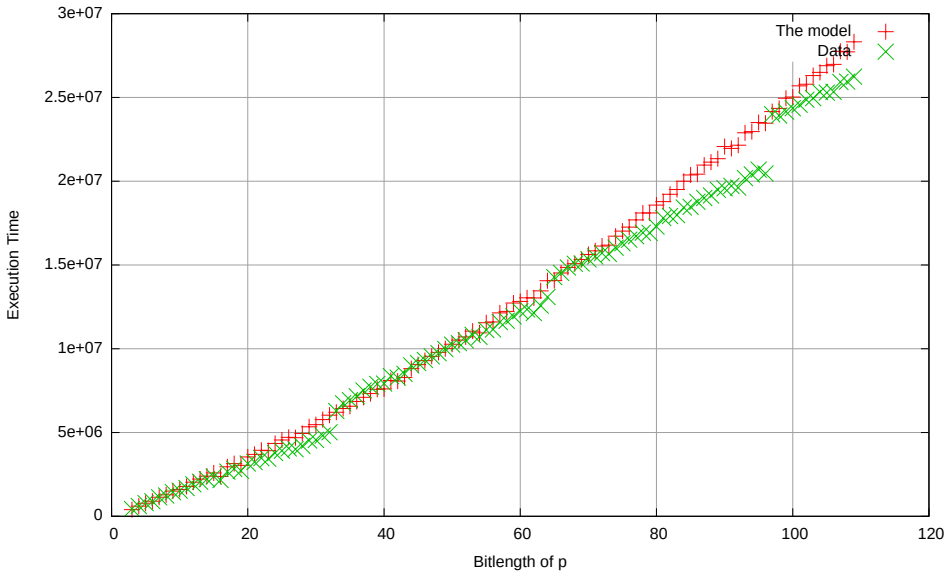

Figure 5.9: Model for the combining function of CFE.

In Figure 5.10 the standarised errors are plotted. Clearly the model does not predict the second data set very well. The only input to the model T_c is the Hamming weight of the field's characteristic. The Hamming weight is investigated in Figure 5.11. It is seen that the average Hamming weight does not deviate much from the expected Hamming weight for the two data sets. This indicates that some unknown dependency is not considered in the model.

System Parameter Generation

The model for the generation of the system parameters and the associated data are plotted in Figure 5.12 and in Figure 5.13 the error is plotted. The error is between -3 and 3 and therefore the model predicts the data well.

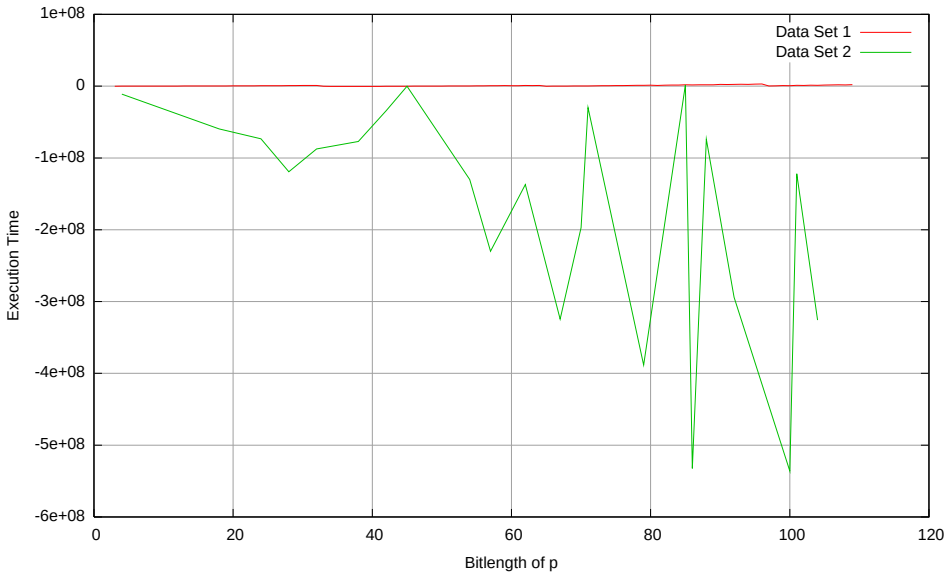

Figure 5.10: The error of CFE combining function's model.

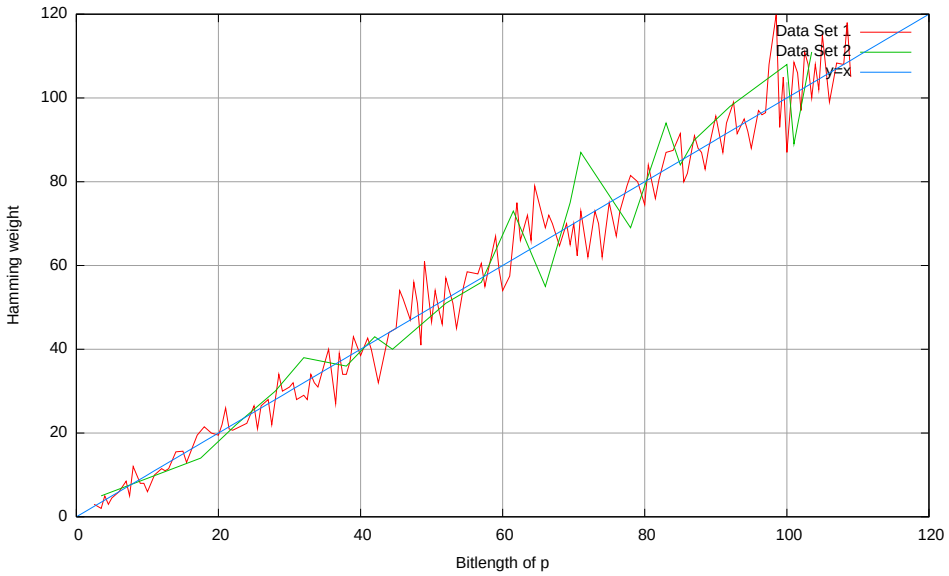

Figure 5.11: The average Hamming weight of the index of the sequence element in CFE.

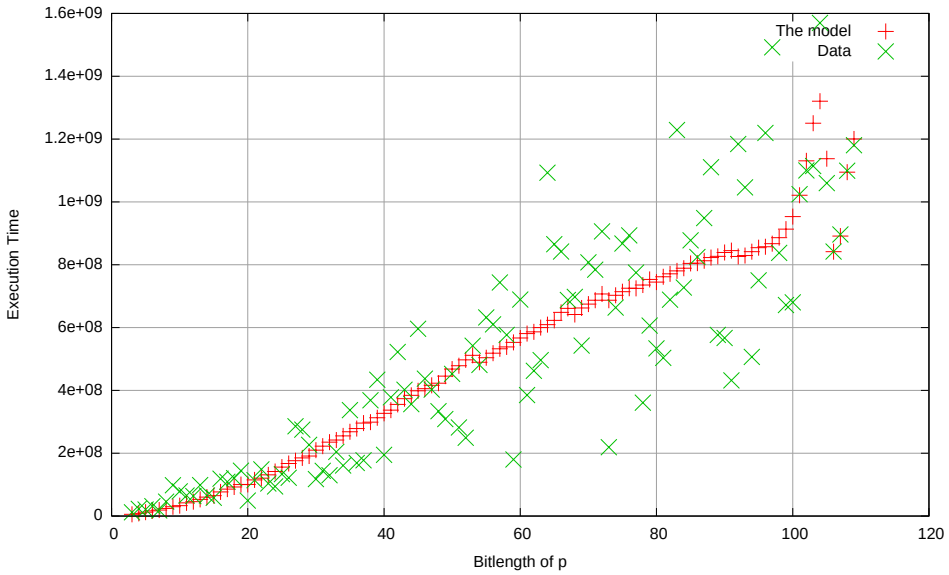

Figure 5.12: Model for system parameter generation in CFE.

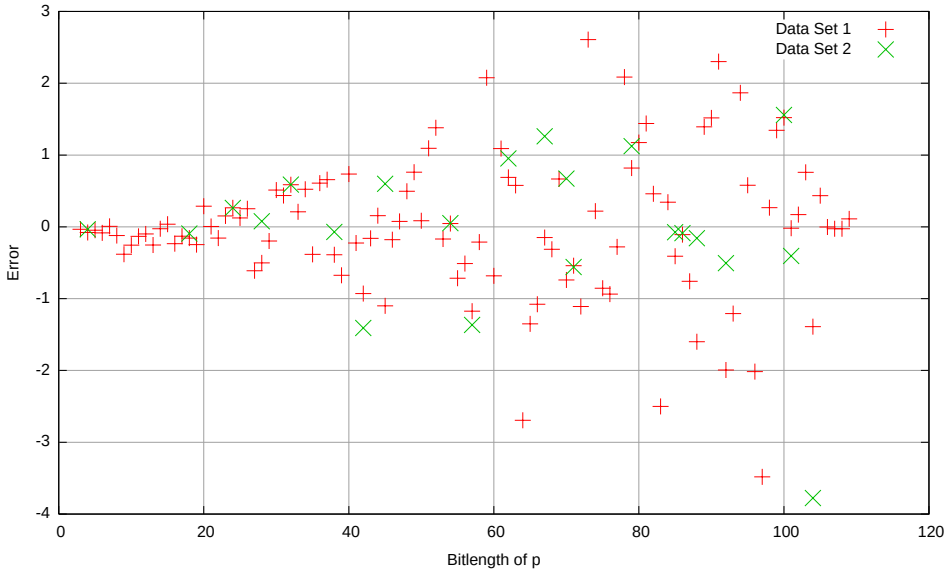

Figure 5.13: The error for CFE system parameter generation model.

5.4 Modeling the EXTR protocol

The models that predict the execution time and the complexity for the generation of the system parameters T_s and the combining function T_c are given in this section. Since only the case where $m = 1$ is implemented the models and complexity are given specifically for $m = 1$, thus the original XTR in [13] is implemented. The performance of XTR is better than that of EXTR, due to the use of a Karatsuba-like approach for the multiplication, [10].

5.4.1 Complexity of XTR

The orginal description in [13] is sufficient for the implementation of XTR, so that no more theory is needed as was the case for CFE. Despite the complex description of the protocol the complexity is less than that of CFE. This is largely due to the removal of the probabilistic nature of the system parameter generation algorithm.

All operations in XTR are performed in $\mathbb{F}_{p^2}$ using an optimal normal basis over $\mathbb{F}_p$. The extension field is created by the polynomial x^2-x-1 with a root α of x^2-x-1 and the basis is $\{\alpha, \alpha^2\}$. Multiplication in $\mathbb{F}_{p^2}$ is now performed by a Karatsuba-like approach resulting in one fewer multiplication in $\mathbb{F}_p$. The Karatsuba-like approach is a trade-off between addition and multiplication, where the number of additions needed is increased. Let $x, y \in \mathbb{F}_{p^2}$ such that $x = x_0\alpha + x_1\alpha^2$ and $y = y_0\alpha + y_1\alpha^2$ where $x_0, x_1, y_0, y_1 \in \mathbb{F}_p$. To compute xy let $a = x_0y_0$, $b = x_1y_1$, $c = (y_0 + y_1)(x_0 + x_1)$ and $d = c - a - b$ then $xy = (b - d)\alpha + (a - d)\alpha^2$. Another optimization from [13] is the following. Let $z \in \mathbb{F}_{p^2}$ such that $z = z_0\alpha + z_1\alpha^2$ where $z_0, z_1 \in \mathbb{F}_p$ then

$$\begin{aligned} & xz - yz^p \\ =&(z_0(y_0 - x_1 - y_1) + z_1(x_1 - x_0 + y_1))\alpha \\ &+ (z_0(x_0 - x_1 + y_0) + z_1(y_1 - x_0 - y_0))\alpha^2. \end{aligned}$$

System Parameter Generation

Algorithm 12 is used to compute the system parameters given the prime order of the Diffie-Hellman group. The algorithm performs a random search for a polynomial with the objective that the polynomial is irreducible and the $(p^2-p+1)/q^{th}$ power of a root of the polynomial is not equal to 1. Thus the time needed to compute the system parameters is

$$\begin{aligned}
T_s(p,q) =& 3\big(T_c(p,p+1)+P_p(p-1)+T_c\big(p,p^2-p+1\big)\big)\\
=&\big(12\log_2(p+1)-4H_{(p+1)}\big)M_p+\big(34\log_2(p+1)-2H_{(p+1)}\big)A_p\\
&+(H_{p-1}+\log_2(p-1))M_{p^2}\\
&+\big(12\log_2\big(\big(p^2-p+1\big)/q\big)-4H_{((p^2-p+1)/q)}\big)M_p\\
&+\big(34\log_2\big(p^2-p+1\big)-2H_{(p^2-p+1)}\big)A_p.\\
=&\big(3H_{p-1}+3\log_2(p-1)-4H_{((p^2-p+1)/q)}\big)M_p\\
&+\big(6H_{p-1}+6\log_2(p-1)-2H_{((p^2-p+1)/q)}\big)A_p\\
&+\big(12\log_2(p+1)-4H_{(p+1)}+12\log_2\big(\big(p^2-p+1\big)/q\big)\big)M_p\\
&+\big(34\log_2(p+1)-2H_{(p+1)}+34\log_2\big(\big(p^2-p+1\big)/q\big)\big)A_p.
\end{aligned}$$

Algorithm 12: Compute the system parameters for XTR an implemented

Data: The prime p and a list of prime divisors of p^3-p+1.
Output: c.

```
q = max{iℤ : i|(p^3 − p + 1), i a prime number};
while true do
    c ∈_R F_{p^2};
    compute s_{p+1} with characteristic polynomial x^p − cx^2 + c^p x − 1;
    if s_{p+1} ∈ F_p then
        compute s_{(p^2−p+1)/q} with characteristic polynomial
        x^p − cx^2 + c^p x − 1;
        if s_{(p^2−p+1)/q} ≠ 3 then
            break;
        end
    end
end
```

Combining function

Algorithm 10 is used to compute a specific term s_k in a sequence which is the output of the combining function for XTR. The execution time of the algorithm is dependent on the Hamming weight of k. If the Hamming weight is 0 then the expression $xz - yz^p$ in the algorithm is computed twice resulting in three additions and two multiplications in $\mathbb{F}_{p^2}$ resulting in a complexity of

$$2(4M_p + 10A_p) + 3(2A_p) + 2(3M_p + 4A_p) = 14M_p + 34A_p.$$

If the Hamming weight is 1 then $xz - yz^p$ is computed once, resulting in three additions and four multiplications in $\mathbb{F}_{p^2}$ and giving a complextity of

$$(4M_p + 10A_p) + 3(2A_p) + 4(3M_p + 4A_p) = 16M_p + 32A_p.$$

Thus the complexity of computing the k^{th} term in the sequence is

$$\begin{aligned} T_c(p,k) =& (\log_2 k - H_k)(12M_p + 34A_p) + H_k(16M_p + 32A_p) \\ =& (12\log_2 k - 4H_k)M_p + (34\log_2 k - 2H_k)A_p. \end{aligned}$$

5.4.2 Statistical analysis

Linear regression is used in SAS to determine the models for the complexities T_s and T_c with data set one. For both models the fitted model and the measured data are plotted.

System Parameter Generation

The model for the the generation of system parameters and the associated data are plotted in Figure 5.14. To determine how well the model fits the second data set, the error made with the original data set and a bigger data set of plotted in Figure 5.15. Both curves lie mostly between -3 end 3, validating that the model fits the data well. Also both curves are similar, giving an indication that the model predicts the second data set well enough.

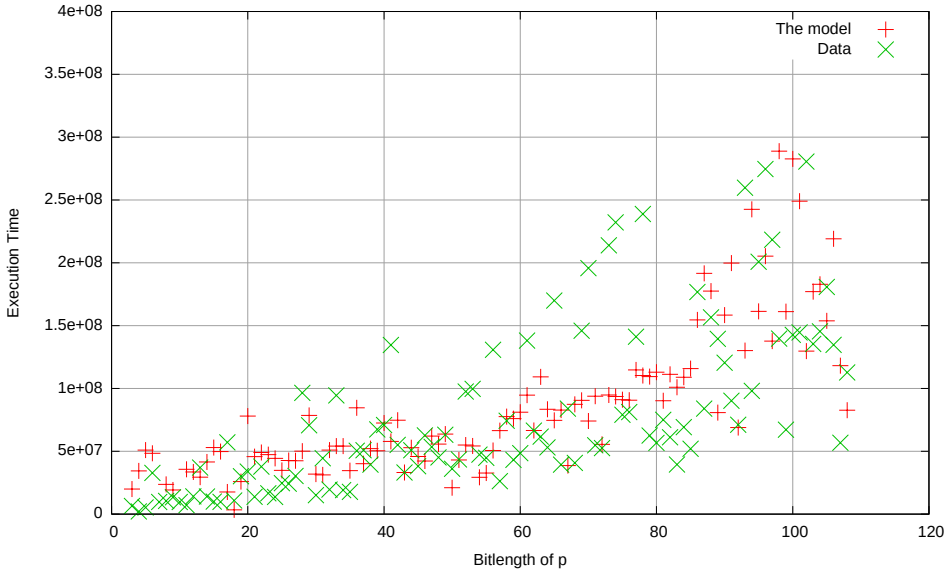

Figure 5.14: Model for generation of the system parameters in XTR.

Combining function

The fitted model for the combining function and the associated data are plotted in Figure 5.16. To determine how well the model fits the second data set, the error made with both data sets are plotted in Figure 5.17. It is immediately seen that the prediction on the second data set is wrong. By plotting the model T_c with the second data set it is seen why the error is so large, see Figure 5.18. The model prediction is much smaller than the measured data. The only inputs of the model are p and the Hamming weight of the exponent k. In Figure 5.19 $\log_2 k/2$ versus H_k for both data sets and the line $y = x$ are plotted. It is expected that $\log_2 k/2 \approx H_k$ but from the graph it is seen that it is not the case and thus the model does not fit the data set very well. The surprising fact is that the average Hamming weight of both data sets are similar and it is expected that the model should fit both data sets equally well.

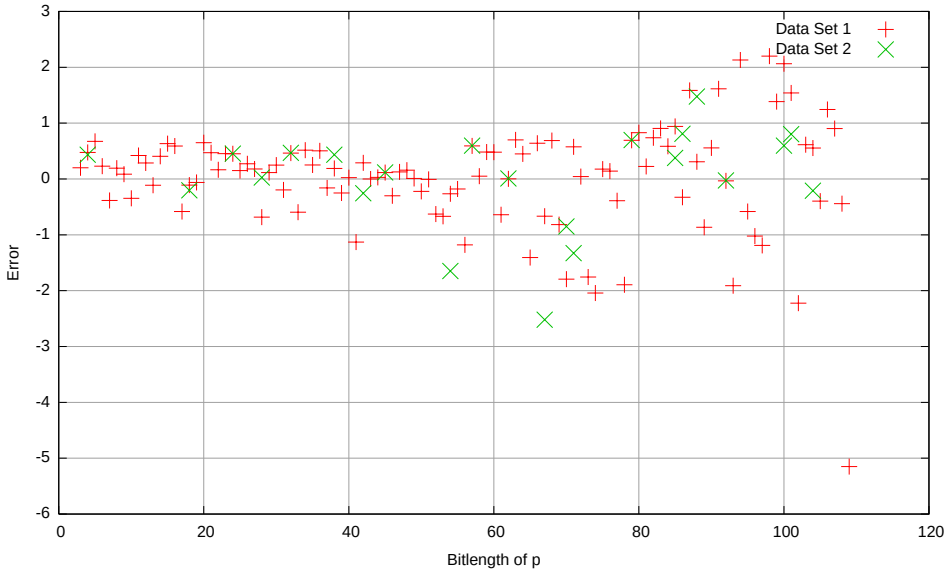

Figure 5.15: The error of XTR system parameter generation's model.

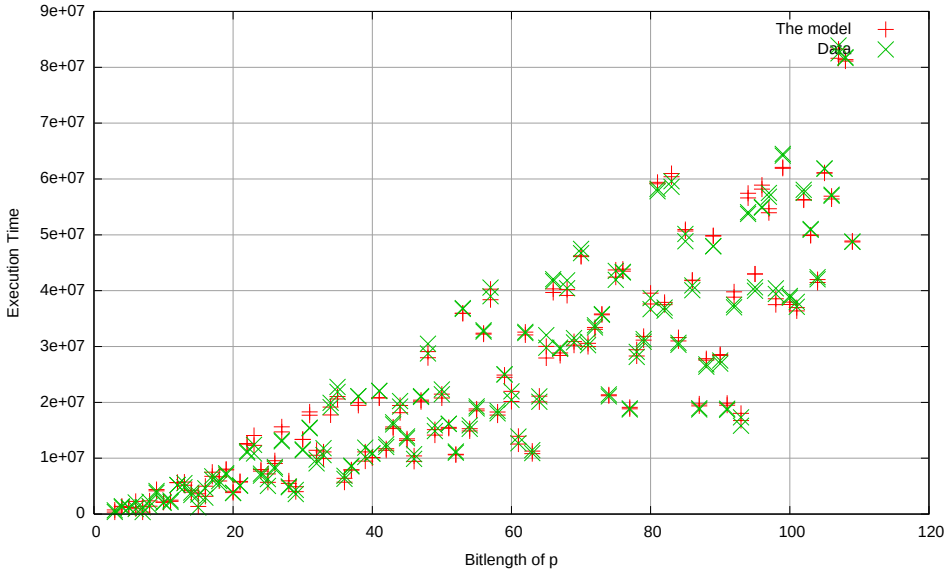

Figure 5.16: Model of XTR combining function.

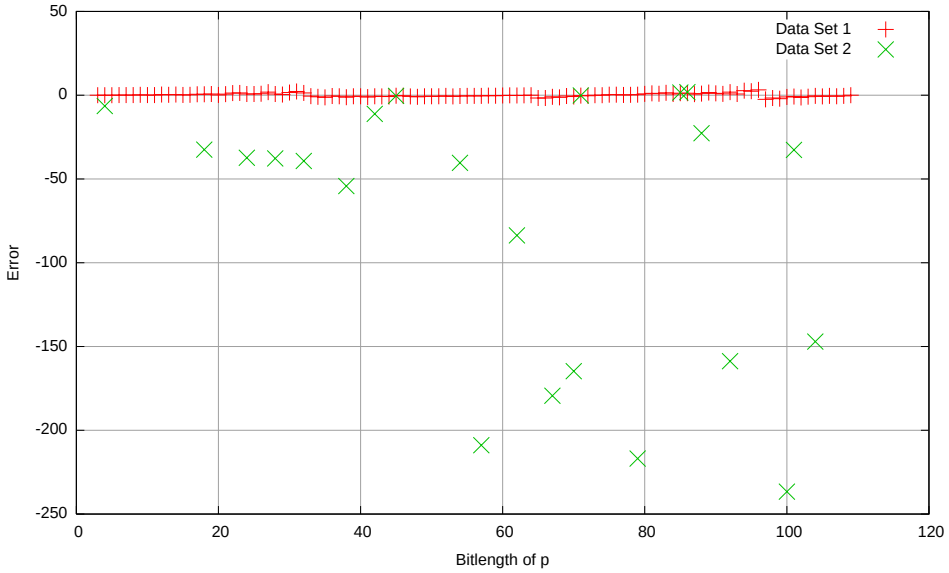

Figure 5.17: The error of XTR combining function's model.

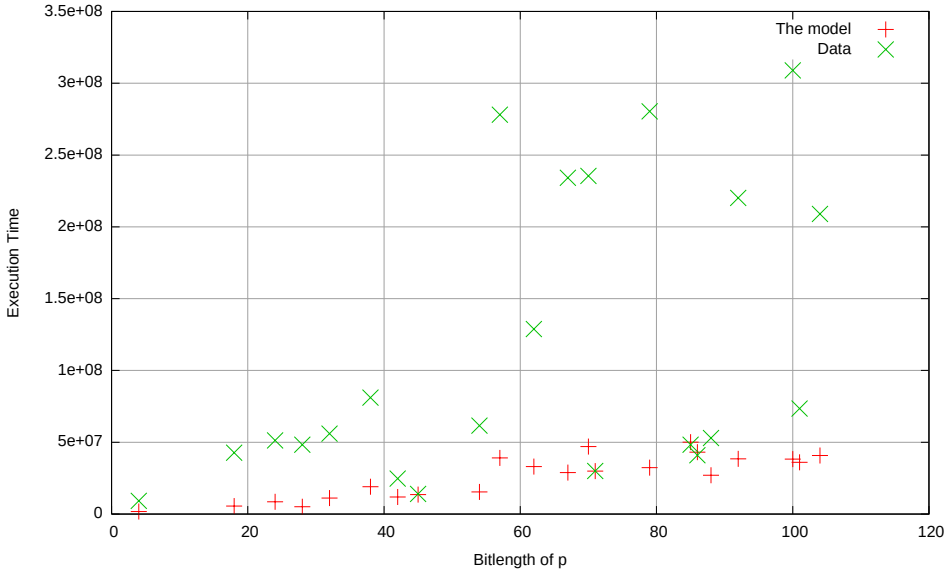

Figure 5.18: The XTR Combining function's model with second data set.

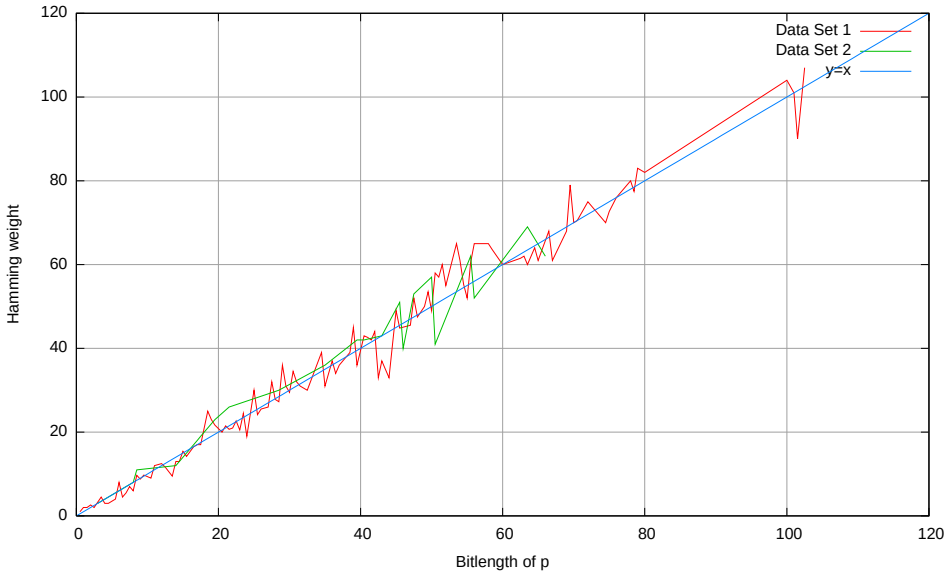

Figure 5.19: The Hamming weight of the index of the sequence element in data set 1 and 2.

Bibliography

[1] Wieb Bosma, James Hutton, and Eric R Verheul. Looking beyond XTR. In *ASIACRYPT '02: Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security*, pages 46–63, London, UK, 2002. Springer-Verlag.

[2] Andries E. Brouwer, Ruud Pellikaan, and Eric R Verheul. Doing more with fewer bits. In *ASIACRYPT '99: Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security*, pages 321–332, London, UK, 1999. Springer-Verlag.

[3] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[4] C. Ding, D. Pei, and A. Salomaa. *Chinese Remainder Theorem*. World Scientific, 1996.

[5] Norman R. Draper and Harry Smith. *Applied Regression Analysis, Includes disk (Wiley Series in Probability and Statistics)*. Wiley-Interscience, April 1998.

[6] Jean-Pierre Escofier. *Galois Theory*. Springer Verlag, 1997.

[7] Shuhong Gao and Hendri W. Lenstra Jr. Optimal normal bases. *Designs, Codes and Cryptography*, 2:315–323, 1992.

[8] Guang Gong and Lein Harn. Public-key cryptosystems based on cubic finite field extension. *IEEE Trans. Inform. Theory*, 45(7):2601–2605, 1999.

[9] Marc Joye and Sung-Ming Yen. Optimal lift-to-right binary signed-digit recoding. *IEEE Transactions on Computers*, 49(7):740–748, 2000.

[10] A. Karatsuba and Yu Ofman. Multiplication of many-digital numbers by automatic computers. *Translation in Physics-Doklady*, 145:293–294, 1962.

[11] Arjen K Lenstra. Using cyclotomic polynomials to construct efficient discrete logarithm cryptosystems over finite fields. *ACIP'97*, LNCS 1270:127–138, 1997.

[12] Arjen K Lenstra and Eric R Verheul. An overview of the XTR public key system. *Proceedings of the Public Key Cryptography and Computation Number Theory Conference*, pages 1–29, 2000.

[13] Arjen K Lenstra and Eric R Verheul. The XTR public key system. In *CRYPTO '00: Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, pages 1–19, London, UK, 2000. Springer-Verlag.

[14] Arjen K Lenstra and Eric R Verheul. Selecting cryptographic key sizes. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 14(4):255–293, 2001.

[15] Rudolf Lidl and Harald Neiderreiter. *Introduction to finite fields and their application*. Cambridge University Press, 1994.

[16] Seongan Lim, Seungjoo Kim, Ikkwon Yie, Jaemoon Kim, and Hongsub Lee. XTR Extended to $\mathbb{F}_{p^{6m}}$. In *SAC '01: Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography*, pages 301–312, London, UK, 2001. Springer-Verlag.

[17] J.L. Massey and J.K. Omura. Computational method and apparatus for finite field arithmetic. US Patent No. 4587627.

[18] Ueli M. Maurer and Stefan Wolf. The Diffie-Hellman Protocol. *Design Codes and Cryptography*, 19(2-3):147–171, 2000.

[19] Kevin S. McCurley. The Discrete Logarithm Problem. *Proceedings of Symposia in Applied Mathematics*, 42:49–74, 1990.

[20] Alfred J. Menezes. *Applications of Finite Fields*. Kluwer, 1993.

[21] The International PGP Home Page. www.pgpi.org.

[22] Victor Shoup. Lower Bounds for Discrete Logarithms and Related Problems. *Lecture Notes in Computer Science*, 1233:256–266, 1997.

[23] Joachim von zur Gathen and Jurgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, second edition, 2003.

[24] Sam S. Wagstaff and Samuel S. Wagstaff. *Cryptanalysis of Number Theoretic Ciphers*. CRC Press, Inc., Boca Raton, FL, USA, 2002.