ON THE TRACE BASED PUBLIC KEY CRYPTOSYSTEMS OVER FINITE FIELDS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MUHAMMAD ASHRAF

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
CRYPTOGRAPHY

AUGUST 2013

Approval of the thesis:

ON THE TRACE BASED PUBLIC KEY CRYPTOSYSTEMS OVER FINITE FIELDS

submitted by **MUHAMMAD ASHRAF** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Department of Cryptography, Middle East Technical University** by,

Prof. Dr. Bülent Karasözen
Director, Graduate School of **Applied Mathematics** _____

Prof. Dr. Ferruh Özbudak
Head of Department, **Cryptography** _____

Prof. Dr. Ersan Akyıldız
Supervisor, **Department of Mathematics, METU** _____

Dr. Barış Bülent Kırlar
Co-supervisor, **Department of Mathematics, SDU** _____

Examining Committee Members:

Associate Prof. Dr. Ali Doğanaksoy
Department of Mathematics, METU _____

Prof. Dr. Ersan Akyıldız
Department of Mathematics, METU _____

Prof. Dr. Ferruh Özbudak
Department of Mathematics, METU _____

Assistant. Prof. Dr. Ömer Küçüksakallı
Department of Mathematics, METU _____

Assistant Prof. Dr. Sedat Akleylek
Department of Computer Engineering, OMU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: MUHAMMAD ASHRAF

Signature :

ABSTRACT

ON THE TRACE BASED PUBLIC KEY CRYPTOSYSTEMS OVER FINITE FIELDS

Ashraf, Muhammad

Ph.D., Department of Cryptography

Supervisor : Prof. Dr. Ersan Akyıldız

Co-Supervisor : Dr. Barış Bülent Kırlar

August 2013, 104 pages

In this thesis, the trace based Public Key Cryptosystems (PKC) are explored from theoretical and implementation point of view. We will introduce cryptographic protocols for the ones they are not discussed yet. We introduce improved trace based exponentiation algorithm for fifth degree recursive relation.

The Discrete Log Problem (DLP), that is computing x, given $y = \alpha^x$ and $< \alpha >= G \subset \mathbb{F}_q^*$, based Public Key Cryptosystems (PKC) are being studied since late 1970's. Such development of PKC was possible because of the trapdoor function $f : \mathbb{Z}_\ell \to G =< \alpha >\subset \mathbb{F}_q^*$, $f(m) = \alpha^m$, is a group homomorphism. Due to this fact, we have Diffie Hellman (DH) type key exchange, ElGamal type message encryption, and Nyberg Rueppel type digital signature protocols. The cryptosystems based on the trapdoor $f(m) = \alpha^m$ are well understood and complete. However, there is another trapdoor function $f : \mathbb{Z}_\ell \to G$, $f(m) \to Tr(\alpha^m)$, where $G =< \alpha >\subset \mathbb{F}_{q^k}^*$, $k \geq 2$, which needs more attention from cryptographic protocols point of view. There are some works for a more efficient algorithm to compute $f(m) = Tr(\alpha^m)$ and not wondering about the protocols. There are also some works dealing with an efficient algorithm to compute $Tr(\alpha^m)$ as well as discussing the cryptographic protocols. In this thesis these works are studied along with introduction of some protocols which are not discussed earlier and trace based exponentiation for fifth degree recursive relation is improved.

Keywords: Public Key Cryptosystems, Discrete Logarithm Problem, Extension Fields, Trace based Exponentiation, Digital Signature Algorithm

ÖZ

SONLU CISIMLER ÜZERINDE IZ TABANLI AÇIK ANAHTARLI KRIPTOSISTEMLER ÜZERINE

Ashraf, Muhammad

Doktora, Kriptografi

Tez Yöneticisi : Prof. Dr. Ersan Akyıldız

Ortak Tez Yöneticisi : Dr. Barış Bülent Kırlar

Ağustos, 2013, 104 sayfa

Bu tezde, iz tabanli Açik Anahtarli Kriptosistemler (AAK), teorik ve uygulama bakis açilarindan incelenmistir. Henüz üzerinde çok durulmamis olanlar için de kriptografik protokoller tanitilmistir. Besinci dereceden özyinelemeli bagintilar için, iyilestirilmis, iz tabanli üs alma yöntemi tanitacagiz. Ayrik Logaritma Problemi (yani verilen $y = \alpha^x$ ve $< \alpha >= G \subset \mathbb{F}_q^*$ degerleri için x'i hesaplama problemi) tabanli Açik Anahtarli Kriptosistemler 1970'lerden beri çalisilmaktadir. Bu AAK çalismalarini mümkün kilan arka kapili fonksiyon $f : \mathbb{Z}_\ell \to G =< \alpha >\subset \mathbb{F}_q^*$, $f(m) = \alpha^m$'in bir grup homomorfizmasi olmasi olmustur. Bunun sayesinde, Diffie-Hellman (DH) tipi anahtar degisim, ElGamal tipi mesaj sifreleme ve Nyberg-Rueppel tipi sayisal imza protokolleri mevcuttur. Arka kapili $f(m) = \alpha^m$ fonksiyonu üzerine kurulu kriptosistemler iyi anlasilmis ve eksiksizdir. Buna ragmen, $f : \mathbb{Z}_\ell \to G$, $f(m) \to Tr(\alpha^m)$, $G =< \alpha >\subset \mathbb{F}_{q^k}^*$, $k \geq 2$ seklinde, kriptografik bakis açisina göre daha fazla önemsenmesi gereken baska bir arka kapili fonksiyon daha vardir. Literatürde, $f(m) = Tr(\alpha^m)$'i hesaplamak için etkili algoritmalar üzerine çalismalar vardir ancak bunlar protokolleri önemsememektedir. Ayrica, $Tr(\alpha^m)$'i etkili bir sekilde hesaplamak için ugrasan ve protokolleri de gözönüne alan çalismalar da mevcuttur. Bu tezde, bu çalismalarla birlikte önceden üzerinde durulmamis bazi protokoller de çalisilmistir. Ve Besinci dereceden özyinelemeli bagintilar için iz tabanli üs alma yöntemi iyilestirilmistir.

Anahtar Kelimeler: Açik Anahtarli Kriptosistemler, Ayrik Logaritma Problemi, Cisim Genislemeleri, Iz tabanli üs alma, Sayisal Imza Algoritmasi

To my family how have been a source of encouragement and to the memory of our parents.

ACKNOWLEDGMENTS

First of all, I would like to express my sincere gratitude to my advisor Prof. Dr. Ersan Akyıldız for the dedicated and continuous support of my Ph.D study and research, for his patience, motivation, enthusiasm, and immense knowledge. His prudent guidance was source of help during all hurdles in my research and writing of this dissertation. I was lucky to get guidance from such a wonderful Ph.D advisor.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Dr. Ferruh Özbudak, Assoc. Prof. Dr. Ali Doğanaksoy, Assist. Prof. Dr. Ömer Kuçuksakallı, and Assist. Prof. Dr. Sedat Akleylek, for their encouragement, insightful comments, and guidance.

My sincere thanks also goes to my co-supervisor Dr. Barış Bülent Kırlar, Assistant. Prof. Dr. Koray Karabina, Dr. Oğuz Yayla, and Dr. Çağdaş Çalık for their wonderful guidance and encouragement.

I thank my fellow labmates in Cryptology Laboratory, IAM, METU, Dr. Mansoor Ahmed Khan, Halil Kemal, and Fuad Hamidli for their congenial company during research.

Last but not the least, I would like to thank my family for their patience, encouragement and prayers for the success, throughout my life.

TABLE OF CONTENTS

ABSTRACT vii

ÖZ ix

ACKNOWLEDGMENTS xiii

TABLE OF CONTENTS xv

LIST OF FIGURES xxi

LIST OF TABLES xxiii

CHAPTERS

1 INTRODUCTION 1

2 PUBLIC KEY CRYPTOSYSTEMS BASED OVER DISCRETE LOGARITHM PROBLEM 3

2.1 Finite Fields 3

2.2 Discrete Logarithm Problem (DLP) 4

2.3 Attacks on DLP 5

2.4 Finite Fields Based PKC 5

2.4.1 DLP Based PKC over G 6

2.4.2 Diffie-Hellman Key Exchange 6

2.4.3 ElGamal Encryption Scheme 7

2.4.4 Nyberg-Rueppel Digital Signature Algorithm . . 7

2.5 Elliptic Curve Based PKC 8

2.5.1 EC Based Diffie-Hellman Key Exchange 10
2.5.2 EC Based ElGamal Public Key Encryption . . . 10
2.5.3 EC Based ElGamal Digital Signature 11
2.6 Non-Singular Circulant Matrices Based PKC 11
2.6.1 Squaring Circulant $k \times k$ Matrix $D : q = 2^r, r \in \mathbb{Z}$ 12
2.6.2 Circulant Matrices Based Diffie-Hellman Type Key Exchange 12
2.6.3 Circulant Matrices Based ElGamal Type Encryption 13
2.7 Torus Based PKC 13
2.8 Trace Based PKC 16
3 ALTERNATE MODELS OF ELLIPTIC CURVES 17
3.1 Introduction 17
3.2 Weierstrass Curves 19
3.3 Normal Form of Elliptic Curves 19
3.3.1 Edwards Curves 19
3.3.2 Twisted Edwards Curves 22
3.3.3 Binary Edwards Curves 25
3.4 Jacobi Curves 25
3.4.1 Jacobi Intersections 26
3.4.2 Twisted Jacobi Intersections 27
3.4.3 Jacobi Quartics 28
3.5 Hessian Curves 30
3.6 Huff Model of Elliptic Curves 33

3.6.1 Huff Curves 34
3.6.2 Binary Huff Curves 37
3.7 Comparison and Conclusions 39
4 MESSAGE TRANSMISSION AND NYBERG RUEPPEL TYPE DIGITAL SIGNATURE FOR GH-PKC 41
4.1 Introduction 41
4.2 GH-Public Key Cryptosystem 43
4.2.1 Preliminaries 43
4.2.2 Motivation to Use LFSR Based Construction . . 45
4.2.3 GH-PKD Scheme 47
4.2.4 GH-RSA-Type Encryption Scheme 47
4.3 Improved Exponentiation Algorithms 48
4.3.1 Improved GH Double Exponentiation 48
4.3.2 Improved GH Single Exponentiation 49
4.4 Proposed Encryption Scheme 50
4.5 GH-Nyberg-Rueppel-Type Digital Signature Algorithm . . 52
4.5.1 Algorithm 4 (GH-NR-DSA Based on GH-ElGamal Type Encryption Scheme) 53
4.5.2 Algorithm 5 (GH-NR-DSA Based on Generic Symmetric Encryption) 53
4.6 Conclusion 55
5 TRACE BASED PUBLIC KEY CRYPTOSYSTEMS 57
5.1 Introduction 57
5.2 Discrete Log Based PKC over Finite Fields 58

5.2.1 DLP Based PKC over G 58

5.3 Trace Based Public Key Cryptosystems over Finite Fields 59

5.3.1 Relation of $f_i(m)$ with LFSR 59

5.3.2 Main Challenges for Trace Based Cryptography . 60

5.3.3 Motivations to use Trace Based Public Key Cryptography 61

5.3.4 Algorithm to compute mixed term $s_{u+v} = Tr(\alpha^u \alpha^v)$, given $Tr(\alpha), u, S_v = (s_{v-k+1} \cdots s_v)$, $s_j = Tr(\alpha^j)$ for $j \in \mathbb{Z}$, and v-unknown. 63

5.3.5 LUC-PKC : Case $k = 2$, p-arbitrary, $q = p^r$, for any positive integer r 64

5.3.5.1 LUC-DH Type Key Exchange 65

5.3.5.2 LUC-ElGamal Encryption Scheme . . 66

5.3.5.3 LUC-Nyberg Rueppel type Digital Signature Algorithm Based on LUC-ElGamal Encryption 66

5.3.6 GH-PKC : Case $k = 3$, p-arbitrary, $q = p^u$, $u \geq 1$ 68

5.3.7 GH-PKC : Special Case $k = 3$, p-arbitrary, $q = p^2$ 69

5.3.7.1 GH-DH Type Key Exchange : Special Case $k = 3$, p-arbitrary, $q = p^2$. 69

5.3.7.2 Modified GH-ElGamal Type Encryption (MXTR-ElGamal): Special Case $k = 3$, p-arbitrary, $q = p^2$ 70

5.3.7.3 GH-Nyberg-Rueppel Type Signature : Special Case $k = 3$, p-arbitrary, $q = p^2$ 71

5.3.7.4 Lenstra and Verheul-Nyberg-Rueppel Type Signature Algorithm 71

5.3.8 Simultaneous Double Exponentiation for Cubic Extensions 72

5.3.9 GG Public Key Cryptosystem 74

5.3.9.1 GG PKC : Case $k = 5$, p-arbitrary, $q = p^u$ 74

5.3.9.2 GG PKC : Special Case, $k = 5$, p-arbitrary, $q = p^2$ 76

5.3.9.3 GG-DH Type Key Exchange : Special Case 77

5.3.9.4 GG-ElGamal Type Encryption Scheme : Special Case 78

5.3.9.5 GG-NR Type Digital Signature Algorithm : Special Case 79

5.3.10 Speeding Up GG-PKC 80

5.3.11 Koray Karabina PKC : Case $k = 4$, $p = 2$, $q = p^{2u+1}, u \geq 1$ 84

5.3.11.1 KK-DH Type Key Exchange : Case $k = 4$, $p = 2$, $q = p^{2u+1}, u \geq 1$ 86

5.3.11.2 KK-ElGamal Type Encryption Scheme : Case $k = 4$, $p = 2$, $q = p^{2u+1}, u \geq 1$ 87

5.3.11.3 KK-Nyberg Rueppel Type Digital Signature Algorithm Based on Generic Symmetric encryption : Case $k = 4$, $p = 2$, $q = p^{2u+1}, u \geq 1$ 87

5.3.11.4 KK-Nyberg Rueppel Type Digital Signature Algorithm Based on KK-ElGamal type encryption : Case $k = 4$, $p = 2$, $q = p^{2u+1}, u \geq 1$ 89

5.3.12 KK-PKC : Case $k = 6$, $p = 3$, $q = p^u, u \geq 1$, $u = odd, t = \sqrt{3q}$ 90

5.3.12.1 KK-DH Type Key Exchange : Case $k = 6$, $p = 3$, $q = p^u$, $u = odd$ 92

5.4 Conclusion 92

6 SUMMARY 95

REFERENCES 97

CURRICULUM VITAE 103

LIST OF FIGURES

Figure 2.1 Addition and doubling over $\mathbb{F}_q$ 8

Figure 3.1 Addition and doubling over $\mathbb{R}$ for $d < 0$ 20

Figure 3.2 Addition and doubling over $\mathbb{R}$ for $d < 1$ 21

Figure 3.3 Addition and doubling over $\mathbb{R}$ for $0 < d < 1$ 21

Figure 3.4 Addition and doubling over $\mathbb{R}$ 28

Figure 3.5 Addition and doubling over $\mathbb{R}$ 31

Figure 3.6 Addition over $\mathbb{R}$ 34

LIST OF TABLES

Table 3.1 Cost of Arithmetic on Alternate Models of Elliptic Curves . . . 39

Table 4.1 Exponentiation Comparison 46

Table 4.2 Computations and rules for double exponentiation 48

Table 4.3 Computational cost for each rules 49

Table 4.4 Comparison of Single Exponentiations 50

Table 4.5 Comparison of proposed encryption scheme with the similar ones 52

Table 4.6 Comparison of GH-NR-DSA with the similar DSAs 55

Table 5.1 Trace Based Compression Factors for Cryptographic Protocols . 62

Table 5.2 Exponentiation Comparison over $\mathbb{F}_{q^2}$ 62

Table 5.3 Exponentiation Comparison over $\mathbb{F}_{q^3}$ 62

Table 5.4 Exponentiation Comparison over $\mathbb{F}_{q^5}$ 62

Table 5.5 LUC-NR-DSA Analysis 67

Table 5.6 Comparison of GH ElGamal encryption scheme with the similar ones 71

Table 5.7 Stam and Lenstra's Rules for Double Exponentiation 72

Table 5.8 Comparison of GH-NR-DSA with the similar DSAs 74

Table 5.9 GG-ElGamal type Encryption scheme Analysis 79

Table 5.10 GG-NR-DSA Analysis 80

Table 5.11 Substitution values for u, v and derived formulas 81

Table 5.12 The Rules for Double Exponentiation for $\mathbb{F}_{q^5}, q = p^2$ 82

Table 5.13 Analysis of Rules for Double Exponentiation for $\mathbb{F}_{q^5}, q = p^2$ . . . 82

Table 5.14 The Rules for Factor-4 Double Exponentiation 86

Table 5.15 Analysis of KK-NR-DSA based on Generic Symmetric Encryption 88

Table 5.16 Analysis of KK-NR-DSA based on KK-ElGamal Encryption . . 90

CHAPTER 1

INTRODUCTION

The Discrete Log Problem (DLP), that is computing x, given $y = \alpha^x$ and $< \alpha >= G \subset \mathbb{F}_q^*$, based Public Key Cryptosystems (PKC) are being studied since late 1970's. Such development of PKC was possible because of the trapdoor function $f : \mathbb{Z}_\ell \to G =< \alpha >\subset \mathbb{F}_q^*$, $f(m) = \alpha^m$ is a group homomorphism. Due to this fact we have; Diffie Hellman (DH) type key exchange, ElGamal type message encryption, and Nyberg Rueppel type digital signature protocols. The cryptosystems based on the trapdoor $f(m) = \alpha^m$ are well understood and complete. However, there is another trapdoor function $f : \mathbb{Z}_\ell \to G$, $f(m) \to Tr(\alpha^m)$, where $G =< \alpha >\subset \mathbb{F}_{q^k}^*$, $k \geq 2$, Tr is the trace function from $\mathbb{F}_{q^k}/\mathbb{F}_q$, which needs more attention of the researchers from cryptographic protocols point of view. In this case although f is a computable, but it is not clear how to produce protocols such as Diffie Hellman type key exchange, ElGamal type message encryption, and Nyberg Rueppel type digital signature algorithm, in general. It would be better, of course if we can find more efficient algorithm than repeated squaring and trace to compute $f(m) = Tr(\alpha^m)$ together with these protocols. In the literature we see some works for a more efficient algorithm to compute $f(m) = Tr(\alpha^m)$ and not wondering about the protocols. We also see some works dealing with an efficient algorithm to compute $Tr(\alpha^m)$ as well as discussing the cryptographic protocols. These works are presented by Smith, Lennon and Skinner (LUC-PKC) in 1994 [71, 72], L.Horn and G.Gong, (GH-PKC) in 1998 [30, 31], A.K.Lenstra and E.R.Verheul (XTR-PKC) in 2000 [47, 48, 16] and K.Giuliani, G.Gong (GG-PKC) in 2003 [27] and Koray Karabina (KK-PKC) in 2009 [41, 42].

In the literature there are also so called torus based cryptosystems introduced by K.Rubin and A.Silverberg in 2003 [64]. These systems depend on the parametric representation of the group $\mathbb{T}_k(\mathbb{F}_q) \cong G_{q,k} =< \alpha >\subset \mathbb{F}_{q^k}^*$ and $\mathbb{T}_k(\mathbb{F}_q)$ rather than the efficient algorithm to compute $Tr(\alpha^m)$, where $\mathbb{T}_k(\mathbb{F}_q)$ is torus consisting of the elements in $\mathbb{F}_{q^k}$ of norm 1 over every subfield.

The aim of this thesis is to study the so called trace based cryptography studied in the literature and introduce cryptographic protocols for the ones they are not discussed in the literature. As a result of this study we introduced GH-ElGamal Encryption scheme, GH-NR-DSA, modified XTR-ElGamal encryption scheme, GG-ElGamal Encryption scheme, GG-NR-DSA and KK-ElGamal type encryption schemes are added to the literature. The semantic security of encryption

schemes is ensured for the trace based cryptosystems. The key exchange security depends upon the solving Discrete Logarithm Problem over finite field extensions where as semantic security of encryption schemes depends upon splitting $z \in \mathbb{F}_q$ into two numbers $(X, Y) \in \mathbb{F}_q^2$ such that $z = X + Y$ where $s_e \mapsto X$ and $s_{-e} \mapsto Y$. Moreover, $e \in \mathbb{Z}_\ell$ and $\mathbb{Z}_e ll$ is chosen large enough so that the brute force attack becomes infeasible. The number of such (X, Y) are equal to $\frac{q-1}{2}$. Now we introduce the general terminologies in cryptography.

The cryptography has made its way into all sphere of modern life. The main applications include but not limited to Internet, banking sector, e-commerce, security agencies, and so on. In fact, as Internet and computers are becoming necessity of modern life so is cryptography receiving attention. Therefore, cryptography has an important role in present sciences and may become highly technical and integral part of future sciences. The cryptography is divided into two branches symmetric key cryptography and asymmetric key cryptography (public key cryptography).

(i) ***Symmetric Key Cryptography***: The sender and recipient use same key for both encryption and decryption.

(ii) ***Asymmetric Key Cryptography***: The sender and recipient use different keys to agree on a shared secret.

Although symmetric key cryptography is generally used for large portion of data but still asymmetric key is used to securely agree on same symmetric key. Another advantage of the public key cryptography is its ability to provide confidentiality, integrity and authenticity simultaneously through encryption and digital signatures. Its importance may be measured from the fact that in many countries digital signatures are considered legal. The PKC with all its advantages has inherited expensive arithmetic operations. Which apparently lure away its usefulness. Therefore a large portion of research in PKC is dedicated to speeding up arithmetic operations, without compromising its security.

CHAPTER 2

PUBLIC KEY CRYPTOSYSTEMS BASED OVER DISCRETE LOGARITHM PROBLEM

Since the introduction of hardness of intractability of discrete logarithm by Diffie and Hellman in late 1970 [22] there have been many efforts to improve efficiency of cryptosystems based upon this intractability. In literature there are efforts to discover new one-way functions such as integer factoring, knapsack, lattices, braid groups, coding theory and non-commutative groups. Despite all these systems, the modern PKC are based on hardness of either integer factorization [63] or Discrete Logarithm Problem [22]. In this chapter, we are briefly dealing with DLP, its variants, attacks on DLP and various PKC based over DLP. Before mentioning these subjects, we first define finite fields.

2.1 Finite Fields

Let $q = p^r$, where p is a prime and $r \in \mathbb{Z}^+$. For every such prime and r there exists a unique (up to isomorphism) finite field with q elements and denoted by $\mathbb{F}_q$, which can be constructed by the following methods:

(i) For every prime p, a finite field $\mathbb{F}_p \cong \mathbb{Z}_p = \{0, 1, \cdots, p-1\}$ can constructed by integers modulo p. The operations such as addition and multiplication can be carried out usually and then result can be reduced by p. However, the inversion is carried out by using extended Euclidean algorithm.

(ii) Let p be a prime and $r \in \mathbb{Z}^+$ then there exists an irreducible monic polynomial $f(x)$ over $\mathbb{F}_p$ of degree r such that the simple algebraic extension $\mathbb{F}_p[x]/f(x)$ of $\mathbb{F}_p$ can be identified by $\mathbb{F}_p(\beta) \subset \bar{\mathbb{F}}$, where $\bar{\mathbb{F}}$ is an algebraic extension and β is a root of $f(x)$. The $\mathbb{F}_p(\beta) \cong \mathbb{F}_{p^r}$ has exactly p^r elements uniquely represented by $\sum_{i=0}^{r-1} c_i \beta^i$, $c_i \in \mathbb{F}_p$. The arithmetic operations are carried out using polynomial modulo $f(x)$

Definition 2.1 (Trace)**.** Let $\mathbb{F}_{q^k}$ be kth degree extension of $\mathbb{F}_q$ and $\alpha \in \mathbb{F}_{q^k}$, then

trace of α over $\mathbb{F}_q$ is given as:

$$Tr : \mathbb{F}_{q^k} \to \mathbb{F}_q,$$
$$Tr(\alpha) = \sum_{i=0} k - 1\alpha^{q^i}.$$

(i) $Tr(\alpha + \beta) = Tr(\alpha) + Tr(\beta)$ where $\alpha, \beta \in \mathbb{F}_{q^k}$.

(ii) $Tr(u\alpha) = uTr(\alpha)$, and $Tr(u) = ku$ for $u \in \mathbb{F}_q$.

Notations: Let G be a subgroup of a multiplicative finite field $\mathbb{F}_q^*$ of $q-1$ elements, α be a generator of G and order of α that is $\text{ord}(\alpha) = \ell | (q-1)$ such that $G =< \alpha > \subset \mathbb{F}_q^*$, where q is a prime power. Let $\mathbb{F}_{q^k}$ be a finite field extension of $\mathbb{F}_q$ of degree $k > 1 \in \mathbb{Z}^+$ and $Tr : \mathbb{F}_{q^k} \longrightarrow \mathbb{F}_q$. Let $Tr(\alpha) = s_1$ and $Tr(\alpha^e) = s_e$.

2.2 Discrete Logarithm Problem (DLP)

The DLP is defined as, computing x, given $y = \alpha^x$ and $G =< \alpha > \subset \mathbb{F}_q^*$. Note that computing y, given x and α is straight forward and well defined exponentiation i.e; $\alpha^x = \overbrace{\alpha\alpha, \cdots, \alpha}^{x}$, but computing x, given $y = \alpha^x$ is hard, hence the one-wayness of discrete logarithm. The development of Public Key Cryptosystems based on this one-wayness was possible because of the trapdoor function, $f : \mathbb{Z}_\ell \to G =< \alpha > \subset \mathbb{F}_q^*$, $f(m) = \alpha^m$ is a group homomorphism. Along with DLP we also take care of the following problems for PKC over finite fields.

Definition 2.2 (Computational Diffie Hellman Problem (CDHP))**.** : The CDHP is defined as, given α, α^m, α^n determine α^{mn}, for $m, n \in \mathbb{Z}$.

Definition 2.3 (Decisional Diffie Hellman Problem (DDHP))**.** : The DDHP is defined as, given α, α^m, α^n, α^{mn} and α^r such that $\alpha^r \neq \alpha^{mn}$, for random $r \in \mathbb{Z}$, determine α^{mn} or α^r is the solution to the DHP with α^m and α^n.

Based on DLP, and DHP following are three basic cryptographic protocols which will be focused in this thesis:

(i) Diffie-Hellman type key agreement,

(ii) ElGamal type encryption scheme,

(iii) Nyberg-Rueppel type Digital Signature scheme.

Let $Tr(\alpha) = s_1$ and $Tr(\alpha^m) = s_m$, where Tr is trace function from $\mathbb{F}_{q^k}/\mathbb{F}_q$. Then,

Definition 2.4 (k-Trace-DLP)**.** Given s_1 and s_e, the problem of finding $e > 1 \in \mathbb{Z}$ with is called the kth-Trace-Based-Discrete Logarithm Problem (k-Trace-DLP) or alternatively kth-order LFSR-Based Discrete Logarithm Problem (k-LFSR-DLP), where LFSR means Linear Feedback Shift Register.

Definition 2.5 (k-Trace-DHP)**.** Given s_1, s_e and s_r, the problem of determining s_{er} is called the kth-Trace-Based-Diffie-Hellman Problem (k-Trace-DHP) or alternatively kth-order LFSR-Based Diffie-Hellman Problem (k-LFSR-DHP).

Definition 2.6 (k-Trace-DDHP)**.** Given s_1, s_e, s_r, s_{er} and s_c, where c is a randomly chosen integer, the problem of determining the solution of k-Trace-DHP whether s_{er} or s_c is called the kth-Trace-Based-Decisional Diffie-Hellman Problem(k-Trace-DDHP) or alternatively kth-order LFSR-Based Decisional Diffie-Hellman Problem (3-LFSR-DDHP).

2.3 Attacks on DLP

To solve DLP one has the option to compute all the values of α^i for $1 < i < G$ and compare with given value $y = \alpha^x$, but this is inefficient and becomes computationally infeasible with growing values of G. In literature we see there are algorithms which can solve DLP efficiently. Therefore, one should be careful in selecting G to be large enough to avoid exploitation by these algorithms. Following are the main algorithms which can solve DLP efficiently in sub-exponential time:

(i) ***Baby-Step Jiant-Step:*** This algorithm is deterministic and is introduced by Shanks [67]. It solves the DLP with time complexity $O(\sqrt{\ell})$.

(ii) ***Pollard's ρ and λ Method:*** These are probabilistic methods. The ρ method can solve DLP with complexity of $O(\sqrt{\ell})$ [61, 62]. The Pollard's λ method can solve DLP with complexity of $O(\sqrt{\gamma})$, where γ is the upper bound of exponent and mostly smaller than ℓ and do not require prior knowledge of the group order ℓ.

(iii) ***Index Calculus Method:*** The Index Calculus method [33] is an other sub-exponential time algorithm which can solve DLP with time complexity of the order $O(e^{\sqrt{2\log\ell\log\log\ell}})$.

2.4 Finite Fields Based PKC

The general mathematical structure of the PKCs based on finite field and cryptographic protocols such as Key Exchange, Encryption scheme, and Nyberg Rueppel Digital signature algorithm are basic building blocks for the PKC. Also the PKC is considered practical if we have cryptographic protocols and efficient algorithm for computations involved in these protocols. Keeping this in focus, we discuss DLP based PKC over subgroup G of a finite field.

2.4.1 DLP Based PKC over G

Let,

$$G =< \alpha > \subset \mathbb{F}_q^*, \quad q = p^r, \text{ i.e; prime power}$$
$$G =< \alpha > \cong \mathbb{Z}_\ell, \quad ord(\alpha) = \ell$$
$$f : \mathbb{Z}_\ell \to G; \text{ with } f(m) = \alpha^m;$$

The f is a computable trapdoor, and polynomial time algorithm is as follows.

Algorithm 1 To Compute $f(m) = \alpha^m \in G =< \alpha > \subset \mathbb{F}_q^*$

Require: $\alpha \in \mathbb{F}_q$ and $m = \sum_{j=0}^{l-1} \epsilon_j 2^j \in \mathbb{Z}$, with $\epsilon_j = \{0, 1\}$, $\epsilon_{l-1} = 1$
Ensure: α^m
  $\beta \leftarrow 1$
  for $j \leftarrow l - 1$ to 0 **do**
    $\beta \leftarrow \beta \cdot \beta$
    if $\epsilon_j = 1$ **then**
      $\beta \leftarrow \beta \cdot \alpha$
    end if
  end for
  return (β)

The above algorithm performs t squaring and $wt(m) - 1$ multiplications by α, where $t = \lfloor log_2 m \rfloor$ and $wt(m)$ is the number of 1's in the binary representation of m.

For the DLP on G such that given $y \in G$, find $f^{-1}(y)$, one has in general **Polard's Rho**$(O(e^{\sqrt{\log \ell}}))$ and **Index Calculus** $(O(e^{\sqrt{2 \log \ell \log \log \ell}}))$ algorithms which are sub-exponential time. By choosing the order G a large prime ℓ one can avoid these algorithms so that f becomes one-way function on $\mathbb{F}_q^*$. With this setup now we give the following cryptographic protocols:

2.4.2 Diffie-Hellman Key Exchange

1. ***System Public Parameters:*** $G =< \alpha > \subset \mathbb{F}_q^*$; $ord(\alpha) = \ell$.
2. **A**'s public key $P_A = \alpha^a$, and private key $a \in \mathbb{Z}_\ell$.
3. **B**'s public key $P_B = \alpha^b$, and private key $b \in \mathbb{Z}_\ell$.
4. Their common key, $K = P_{AB} = P_{BA} = \alpha^{ab}$:

$$\mathbf{A} \to P_{AB} = (P_B)^a = \alpha^{ab},$$
$$\mathbf{B} \to P_{BA} = (P_A)^b = \alpha^{ab}.$$

2.4.3 ElGamal Encryption Scheme

1. ***System Public Parameters:*** $G =< \alpha >\subset \mathbb{F}_q^*$; $ord(\alpha) = \ell$.
2. **A** selects a random $x \in \mathbb{Z}_\ell$. **A**'s public key: $h = \alpha^x$, and private key $=x$.
3. **Assumption** : message $M \in G$.
4. ***Encryption*** : **B** encrypts message M as follows:
   (i) Chooses a random $r \in \mathbb{Z}_\ell$, then computes $c_1 = \alpha^r$, $s = h^r$ and $c_2 = Ms$.
   (ii) Sends cipher text $C = \{c_1, c_2\} \in G^2$ to **A**.
5. ***Decryption***: **A** decrypts $C \in G^2$ based upon his private key x as follows;

$$c_1^{-x} c_2 = \alpha^{-xr} M h^r = M \alpha^{xr} \alpha^{-xr} = M.$$

2.4.4 Nyberg-Rueppel Digital Signature Algorithm

1. ***System Public Parameters:*** $G =< \alpha >\subset \mathbb{F}_q^*$, $ord(\alpha) = \ell$, and bijection map $f : G \longrightarrow \mathbb{Z}_\ell$.
2. **A**'s public key $P_A = \alpha^x$, and private key: $x \in \mathbb{Z}$.
3. Assumption: message $M \in G$.
4. ***Signature :*** **A** signs the message $M \in G$ as follows:
   (i) Chooses a random $k \in \mathbb{Z}_\ell$.
   (ii) Computes ($r = M\alpha^{-k}$ and $s = k^{-1}(1 - f(r)x) \pmod{\ell}$.
   (iii) Signature $=(r, s) \in (G \times \mathbb{Z}_\ell)$ and sends (M, r, s) to **B**.
5. ***Verification:*** **B** verifies by computing $(P_A)^{-f(r)} r^s = \alpha^{sk-1-sk} M^s = M^s \alpha^{-1}$.

On the DLP based PKC, the main point is the trapdoor function:

$$G =< \alpha >\subset \mathbb{F}_q^*, \quad ord(\alpha) = \ell,$$
$$f : \mathbb{Z}_\ell \longrightarrow \mathbb{F}_q^*, \quad f(m) = \alpha^m.$$

Note that the trapdoor function f is a group homomorphism; $f(m + k) = f(m)f(k)$, and one should be careful to choose $< \alpha > G \subset \mathbb{F}_q^*$ to avoid algorithms to attack DLP on G and choose, if possible, the smallest size G such that DLP on G is computationally equivalent to DLP on $\mathbb{F}_q^*$.

2.5 Elliptic Curve Based PKC

Since the advent of elliptic curve cryptosystems, independently by Miller (1985) and Koblitz (1987), arithmetic of elliptic curves have been in focus for cryptographic researchers. Since then, many methods to speed up the arithmetic of elliptic curves have been proposed. These methods can be divided into four different categories:

(i) Use optimum underlying finite field extensions,

(ii) Use optimum coordinates for representation of group elements,

(iii) Use efficient arithmetic methods,

(iv) Use alternate models of elliptic curves.

An elliptic curve in affine coordinates of simplified Weierstrass form over the finite field $\mathbb{F}_q$ with $char(\mathbb{F}_q) \neq 2, 3$ is defined by

$$E \ : \ y^2 = x^3 + ax + b, \tag{2.1}$$

where $a, b \in \mathbb{F}_q$. If we apply the process of homogenization with $x = X/Z$ and $y = Y/Z$ for $Z \neq 0$ to (2.1), we obtain the homogeneous equation in projective coordinates given by

$$E \ : \ Y^2Z = X^3 + aXZ^2 + bZ^3, \tag{2.2}$$

where $a, b \in \overline{\mathbb{F}}$ (the algebraic closure of $\mathbb{F}$). The curve E has exactly one point with coordinate Z equal to zero, namely $(0 : 1 : 0)$. This point is so called point at infinity and denoted by ∞. The curve E has an additive group structure together with the identity element ∞.

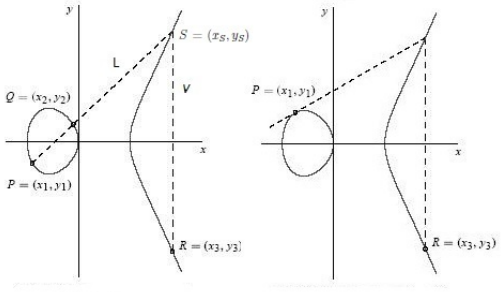

Figure 2.1: Addition and doubling over $\mathbb{F}_q$

The geometric interpretation of the addition law is given by the following way using divisor theory from algebraic geometry [15]: Let $P, Q \in E$. Suppose the

line between P and Q (tangent line if $P = Q$) has an equation $L(x, y) = 0$. By Bezout's theorem, this line L intersects E at a third point $S = (x_S, y_S)$ in the projective space. Then the divisor of L is $\text{div}(L) = (P) + (Q) + (S) - 3(\infty)$. The vertical line $V(x) = (x - x_S)$ passes through the points S and $R = P + Q$. Then $\text{div}(V) = (S) + (R) - 2(\infty)$. Therefore, the equation $R = P + Q$ corresponds to $\text{div}(L/V) = (P) + (Q) - (R) - (\infty)$.

This observation allows us to write down the explicit formula for point addition and point doubling of the curve E as follows: Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be the points on E with $P, Q \neq \infty$ and $Q \neq -P$. Then

- If $P \neq Q$, then $P + Q = R = (x_3, y_3)$, where

$$\begin{cases} x_3 = (\dfrac{y_2 - y_1}{x_2 - x_1})^2 - x_1 - x_2 \\ y_3 = (\dfrac{y_2 - y_1}{x_2 - x_1})(x_1 - x_3) - y_1 \end{cases}$$

- If $P = Q$, then $2P = R = (x_3, y_3)$, where

$$\begin{cases} x_3 = (\dfrac{3x_1^2 - a}{2y_1})^2 - 2x_1 \\ y_3 = (\dfrac{3x_1^2 - a}{2y_1})(x_1 - x_3) - y_1 \end{cases}$$

Formulas that do not involve field inversions for adding and doubling points in projective coordinates can be derived by first converting the points to affine coordinates, then using the formulae above to add the affine points, and finally clearing denominators. The computational cost of point addition and point doubling in projective coordinates are $12\mathbf{M} + 2\mathbf{S}$ and $5\mathbf{M} + 6\mathbf{S}$, respectively. Until today, much more point representations were used in simplified Weierstrass form of elliptic curves for fast computations, such as Jacobian, Chudnovsky and mixed coordinates. The mixed addition formulae can also be obtained by replacing $Z_2 = 1$ in this form that reduces the total costs to $9\mathbf{M} + 2\mathbf{S}$. In 2002, Brier and Joye [17] obtained the unified point addition formulae for simplified Weierstrass curves in projective coordinates such that the computational cost is $11\mathbf{M} + 5\mathbf{S} + 1\mathbf{D}$.

Here, we enumerate the cost of field operations in terms of multiplication $\mathbf{M}$, squaring $\mathbf{S}$, multiplication by a constant $\mathbf{D}$, and addition/subtraction $\mathbf{a}$ in $\mathbb{F}_q$. With this preamble now we define Elliptic Curve DLP and give Public Key Cryptosystem protocols.

Definition 2.7. Keeping the notations above, the Elliptic Curve DLP (ECDLP) is to compute n, given P and Q such that $Q = nP$, where $n \in \mathbb{Z}$.

For Elliptic Curve Cryptography (ECC), immediately, question arises that why Elliptic Curves are used in cryptographic applications? The simple and straight

forward answer is that ECC provide security equivalent to classical systems based on finite fields while using fever bits. The fever bits mean implementation will require smaller chip size, lesser power consumption and with comparable execution time. Following protocols are discussed here:

(i) Elliptic Curve (EC) Based Diffie-Hellman Key Exchange

(ii) EC Based ElGamal Public Key Encryption

(iii) EC Based ElGamal Digital Signatures

2.5.1 EC Based Diffie-Hellman Key Exchange

Alice and Bob agrees on $E(\mathbb{F}_q)$ such that the discrete log problem (DLP) is hard in $E(\mathbb{F}_q)$. They also agree on a point $P \in E(\mathbb{F}_q)$ such that the group generated by the P has large order usually a prime.

(i) ***Public Parameters:*** $E(\mathbb{F}_q), P \in E(\mathbb{F}_q)$.

(ii) ***Private Keys*** of Alice and Bob: random m, and n respectively.

(iii) Alice computes $P_m = mP$, and sends to Bob.

(iv) Bob computes $P_n = nP$, and sends to Alice.

(v) Alice and Bob compute mP_n and nP_m respectively. In other words both computes same key $Q = mnP$ without knowing the secret information of each other.

2.5.2 EC Based ElGamal Public Key Encryption

Let Alice wants to send message M to Bob.

1. ***System Public Parameters:*** An Elliptic Curve $E(\mathbb{F}_q)$ and P such that the DLP is hard.
2. ***Bob's Public Key:*** $Q = mP$.
3. ***Bob's Private Key:*** Random positive integer m.
4. ***Encryption:*** To encrypt message M for Bob Alice does the following:
   (i) Acquires Bob's public key and expresses M as a point on $E(\mathbb{F}_q)$.
   (ii) Chooses secret random integer r, computes $M_1 = rP$ and $M_2 = M + rQ$ and sends M_1, M_2 to Bob.
5. ***Decryption:*** Bob retrieve M form M_1 and M_2 by using his private key m by computing $M = M_2 - mM_1$.

2.5.3 EC Based ElGamal Digital Signature

1. ***System Public Parameters:*** $E(\mathbb{F}_q)$, $P \in E(\mathbb{F}_q)$, $ord(P) = \ell$, $f : E(\mathbb{F}_q) \to \mathbb{Z}_\ell$, $Q = rP$, where random $r \in \mathbb{Z}_\ell$ is Alice's ***Private key***.
2. ***Signatures:*** To sign message M Alice does the following:
   (i) Represent M as a point on $E(\mathbb{F}_q)$
   (ii) Choose random $k \in \mathbb{Z}_\ell$ such that $gcd(k, \ell) = 1$ and compute $R = kP$.
   (iii) Compute $s = k^{-1}(M - rf(R)) \pmod{\ell}$. The signatures are (M, R, s).
3. ***Verification:*** Bob verifies Alice's signature by computing:
   $V_1 = f(R)Q + sR$, $V_2 = MP$. Bob accepts if $V_1 = V_2$.

Why it works? $V_1 = f(R)Q + sR = f(R)rP + k^{-1}(M - rf(R))kP = f(R)rP + MP - f(R)rP = MP = V_2$.

2.6 Non-Singular Circulant Matrices Based PKC

Until 2010, it was believed and showed by Menezes and Wu [54] that working with DLP in a group of non singular matrices over finite fields do not offer advantage over working with DLP over finite fields. A. Mahalanobis showed in 2010 [52] showed that working with DLP in group of non singular circulant matrices over finite fields of characteristic 2 offer better performance as compared with DLP over finite fields and comparable performance when compared with elliptic curve based PKC.

Definition 2.8. The DLP in group on non singular matrices is, given non-singular $k \times k$ matrix D and $\widehat{D} = D^r$ over $\mathbb{F}_q$, compute $r \in \mathbb{Z}$.

A.Menezes and Wu in [54] exploited the fact that characteristic polynomial $g_D(x)$ of D is nothing but $g_D(x) = \prod_{i=0}^{k-1}(x - \alpha_i)$, where $\alpha_i \in \mathbb{F}_{q^k}$ are eigenvalues of D and characteristic polynomial $g_{\widehat{D}}(x)$ of $\widehat{D}$ is nothing but $g_{\widehat{D}}(x) = \prod_{i=0}^{k-1}(x - \beta_i)$, where $\beta_i \in \mathbb{F}_{q^k}$ are eigenvalues of $\widehat{D}$. To solve DLP in such cases one has solve DLP for eigenvalues and then apply Chinese Remainder Theorem (CRT). If k is large enough and $g_D(x)$ is irreducible then it gives us nice security of $\mathbb{F}_{q^{k-1}}$ but matrix multiplication becomes expensive as compared with working in $\mathbb{F}_{q^k}$. A. Mahalanobis introduced in [53] PKC based on non singular circulant matrices over finite fields of characteristic 2 with following properties are efficient.

(i) The circulant matrix D with determinant 1,

(ii) The circulant matrix D with row sum 1,

(iii) The polynomial $\frac{g_D(x)}{x-1}$ is irreducible,

(iv) The dimension of D i.e; k is prime, and

(v) $gcd(q, k) = 1$.

When above properties are satisfied then,

$$\frac{\mathbb{F}_q[x]}{x^k - 1} \cong \frac{\mathbb{F}_q[x]}{x - 1} \times \frac{\mathbb{F}_q[x]}{\psi(x)},$$

where $\psi(x) = \frac{x^k-1}{x-1}$. Therefore DLP reduces into two subgroups generated by $x-1$ and $\frac{x^k-1}{x-1}$. If $\frac{x^k-1}{x-1}$ is irreducible, which is of course for k odd prime [50], then DLP in circulant matrix over finite fields is computationally equivalent to DLP over $\mathbb{F}_{q^{k-1}}$. Note that the subgroup generated by $x - 1$ reveals no information as it is either 0 or 1, which is taken care of through property (ii) above.

2.6.1 Squaring Circulant $k \times k$ Matrix D : $q = 2^r$, $r \in \mathbb{Z}$

Computing square of a circulant matrix over finite field of characteristic 2 is efficient [69]. Let $D = (d_0, d_1, \cdots, d_{k-1})$ be a $k \times k$ circulant matrix over F_q with odd $k > 1 \in \mathbb{Z}$. Then computing D^2 is just computing $(d^2_{\lambda(0)}, d^2_{\lambda(1)}, \cdots, d^2_{\lambda(k-1)})$, where λ is the permutation of $0, 1, \cdots, k-1$. Note that, other row vectors of D^2 are obtained by right circulant shift of $(d^2_{\lambda(0)}, d^2_{\lambda(1)}, \cdots, d^2_{\lambda(k-1)})$. Following theorem is proved in [52].

Theorem 2.1. *Let $q = 2^r, r > 0 \in \mathbb{Z}$, D be a $k \times k$ circulant matrix over $\mathbb{F}_q$, with odd $k > 1 \in \mathbb{Z}$ such that $gcd(k, q) = 1$. Then D^2 is computed by squaring d_i^2, $0 \leq i < k$.*

Remark 2.1. Keeping notations above, computing D^m takes $\frac{k^2}{2} \log m$ field multiplications.

2.6.2 Circulant Matrices Based Diffie-Hellman Type Key Exchange

(1.) ***System Public Parameters:*** Circulant $k \times k$ matrix D, $\mathbb{F}_q$, $q = 2^r$, k-odd prime.

(2.) Alice selects private key a random $m > 1 \in \mathbb{Z}$ and computes $P_A = D^m$.

(3.) Similarly, Bob selects private key a random $n > 1 \in \mathbb{Z}$ and computes $P_B = D^n$.

(4.) Both compute common key $K = P_{AB} = P_{BA} = (D^n)^m = (D^m)^n = D^{mn}$.

2.6.3 Circulant Matrices Based ElGamal Type Encryption

(1.) ***System Public Parameters:*** Circulant $k \times k$ matrix D, $\mathbb{F}_q$, $q = 2^r$, k-odd prime.

(2.) ***Public key:*** Alice selects random $n > 1 \in \mathbb{Z}$ as ***private key*** and computes public key D^n.

(3.) ***Encryption:*** Bob wants to send message $M = (m_0, m_1, \cdots, m_{k-1})$ to Alice. He does the following:

(i) Selects random $u > 1 \in \mathbb{Z}$ and computes D^u, $K = (D^n)^u$,
(ii) Encrypts M by computing KM^T, where M^T is transpose of vector M.
(iii) The cipher text is $c = (D^u, KM^T)$.

(4.) ***Decryption:*** Alice decrypts c by computing $K = (D^u)^n$ and then $(KM^T)K^{-1}$.

2.7 Torus Based PKC

Torus based public key cryptosystem is yet another system that enjoys the extension field security and carry out group operations in prime or intermediate subfields. This system was proposed by K.Rubin and A. Silverberg in [64]. This system is an improvement over Lucas functions based PKC[71], and GH-PKC [30], where as, its performance is comparable with XTR [47] system.

Algebraic Tori. An algebraic torus T over $\mathbb{F}_p$ is an algebraic group defined over $\mathbb{F}_p$ which over some extension field is isomorphic to $(\mathbb{G}_m)^d$, where $\mathbb{G}_m$ is the multiplicative group and d is dimension of T. For every n an algebraic torus T_n can be defined with the property that $T_n(\mathbb{F}_p)$ consists of the elements in $\mathbb{F}_{p^n}^*$ of norm 1 over every subfield.

Definitions.

- ***Algebraic Variety.*** An algebraic variety is an object which can be defined in a purely algebraic way, starting from polynomials or more generally from finitely generated algebras over fields. The variety structure allows to express all elements and operations in terms of polynomials.
- ***Weil Restriction.*** The Weil restriction (also known as "Restriction of scalars ") is a functor which, for any finite extension of fields $\mathbb{F}_{p^n}/\mathbb{F}_p$ and any algebraic variety X over $\mathbb{F}_{p^n}$, produces another variety $Res_{\mathbb{F}_{p^n}/\mathbb{F}_p}X$, defined over $\mathbb{F}_p$. This means Weil restriction decomposes a multiplicative group $Res_{\mathbb{F}_{p^n}/\mathbb{F}_p}G_m$ as a product of algebraic tori, one for each divisor of n.

Fundamental Arithmetic for Torus Based Cryptography. As defined above the $Res_{\mathbb{F}_{p^n}/\mathbb{F}_p}\mathbb{G}_m$ is a torus. As per the property of Weil restriction of

scalars gives an isomorphism:

$$(Res_{\mathbb{F}_{p^n}/\mathbb{F}_p}\mathbb{G}_m)(\mathbb{F}_p) \cong \mathbb{G}_m(\mathbb{F}_{p^n}) = \mathbb{F}^*_{p^n}, \tag{2.3}$$

and norm map for $\mathbb{F}_p \subset \mathbb{F}_{p^r} \subset \mathbb{F}_{p^n}$:

$$Res_{\mathbb{F}_{p^n}/\mathbb{F}_p}\mathbb{G}_m \overset{N_{\mathbb{F}_{p^n}/\mathbb{F}_{p^r}}}{\longrightarrow} Res_{\mathbb{F}_{p^r}/\mathbb{F}_p}\mathbb{G}_m. \tag{2.4}$$

For $\mathbb{F}_p$ points we have:

$$T_n(\mathbb{F}_p) \cong \{\alpha \in \mathbb{F}^*_{p^n} : N_{\mathbb{F}_{p^n}/\mathbb{F}_{p^r}}(\alpha) = 1\}, \tag{2.5}$$

whenever $\mathbb{F}_p \subset \mathbb{F}_{p^r} \subset \mathbb{F}_{p^n}$. The dimension of T_n is $\varphi(n)$. $T_n(\mathbb{F}_p)$ is a cyclic subgroup $\mathbb{G}_{p,n} \subseteq \mathbb{F}^*_{p^n}$ of order $\Phi_n(p)$, where Φ_n is the $n-th$ cyclotomic polynomial. This implies that the security of cryptosystems based on the group T_n is that of $\mathbb{F}^*_{p^n}$

Lemma 4, [64].

i). $T_n(\mathbb{F}_p) \cong \mathbb{G}_{p,n}$,

ii). $\#T_n(\mathbb{F}_p) = \Phi_n(p)$.

iii). If $\alpha \in T_n(\mathbb{F}_p)$ is an element of prime order not dividing n, then α does not lie in any proper subfield of $\mathbb{F}_{p^n}/F_p$.

Definitions

- *Rationality of Tori [64]:* Let T be an algebraic torus over $\mathbb{F}_p$ of dimension d. Then T is rational if and only if there is a birational map $\rho : T \to \mathbb{A}^d$ defined over $\mathbb{F}_p$.
- *Rational Parametrization of T [64]:* There are *Zariski* open subsets $W \subset T$ and $U \subset \mathbb{A}^d$, and rational functions $\rho_1, \ldots, \rho_d \in \mathbb{F}_p[x_1, \ldots, x_t]$ and $\psi_1, \ldots, \psi_t \in \mathbb{F}_p[y_1, \ldots, y_d]$ such that $\rho : W \to U$ and $\psi : U \to W$ are inverse isomorphisms. Such a map is a rational parametrization of T.

In this way a compact representation of the group $T(\mathbb{F}_p)$ by means of rational parametrization of torus T is obtained. Thus every element of $W(\mathbb{F}_p) \subset T(\mathbb{F}_p)$ is represented by d coordinates in $\mathbb{F}_q$.

Rational parametrization of T_2. To develop understanding for rational parametrization for T_2 we proceed as follows. Let $char(p) \neq 2$ and $\mathbb{F}_{p^2} = \mathbb{F}_p(\gamma) : \gamma \in \mathbb{F}^*_{p^2}$ with $\omega = \gamma^2 \in \mathbb{F}^*_p$. Since $\gamma^p = -\gamma$, this implies

$$G_{p,2} = \{a + b\gamma : a, b \in \mathbb{F}_p \text{ and } (a + b\gamma)^{p+1} = 1\}, \tag{2.6}$$

$$= \{a + b\gamma : a, b \in \mathbb{F}_p \text{ and } (a^2 - \omega b^2) = 1\}. \tag{2.7}$$

Define maps,

$$\rho : G_{p,2} - \{\pm 1\} \to \mathbb{F}_p^* \quad : \rho(c + d\gamma) = \frac{1+c}{d} \tag{2.8}$$

and

$$\psi : \mathbb{F}_p^* \to G_{p,2} \quad : \psi(a) = \frac{a+\gamma}{a-\gamma}. \tag{2.9}$$

Now $\psi(a) * \psi(b) = \psi\left(\frac{ab+d}{a+b}\right)$. Therefore, all operations are carried out in $\mathbb{F}_p$ directly and multiplication in T_2 has been translated into the map $(a,b) \mapsto \frac{ab+d}{a+b}$ from $(\mathbb{F}_p^*)^2$ to $\mathbb{F}_p^*$.

Adaptation to Cryptographic Protocols[64]. As demo for cryptographic adaptation, we give Diffie Hellman key exchange protocol based on concept of torus.

- ***Torus Based DH Key Exchange***
  - ***System Public Parameters:*** T_k be a rational algebraic torus over $\mathbb{F}_q$ with rational parameterization $\rho : T_k \to \mathbb{A}^m$ with its inverse ψ where $m = \phi(k)$, $G =< \alpha >\subset T_k(\mathbb{F}_q)$,
    $\alpha \in T_k(\mathbb{F}_q)$ with $ord(\alpha) = \ell$ and $\rho(\alpha) = g \in \mathbb{F}_q^m$.
  - $\boldsymbol{A}$'s public $P_{\boldsymbol{A}} = \rho(\alpha^e)$, private key$= e$
    $\boldsymbol{B}$'s public $P_{\boldsymbol{B}} = \rho(\alpha^t)$, private key$= t$.
  - Common key $K = \rho(\alpha^{et})$:

$$\mathbf{A} \to P_{AB} = \rho(\psi(P_B)^e) = K,$$
$$\mathbf{B} \to P_{BA} = \rho(\psi(P_A)^t) = K.$$

  Why it works? Since $\rho \circ \psi$ is the identity, we have $\rho(\psi(P_{\boldsymbol{B}})^a) = \rho(\alpha^{ab}) = \rho(\psi(P_{\boldsymbol{A}})^b)$.

- ***Torus Based ElGamal Type Encryption Scheme***
  - ***System Public Parameters:*** T_k be a rational algebraic torus over $\mathbb{F}_q$ with rational parameterization $\rho : T_k \to \mathbb{A}^m$ with its inverse ψ where $m = \phi(k)$, $G =< \alpha >\subset T_k(\mathbb{F}_q)$,
    $\alpha \in T_k(\mathbb{F}_q)$ with $ord(\alpha) = \ell$ and $\rho(\alpha) = g \in \mathbb{F}_q^m$.
  - $\mathbf{A}$'s public key $P_{\boldsymbol{A}} = \rho(\alpha^e) \in \mathbb{F}_q^m$, private key $e \in \mathbb{Z}_\ell$.
  - Assumption: messages are $\in G$.
  - **Encryption:** $\boldsymbol{B}$ encrypts $M \in G$ as follows:
    Chooses a random $r \in \mathbb{Z}_\ell$ and computes
    - $u = \rho(\alpha^r) \in \mathbb{F}_q^m$
    - $v = \rho(M\psi(P_{\boldsymbol{A}})^r) \in \mathbb{F}_q^m$, and sends the ciphertext c to $\boldsymbol{A}$
    - $c = (u, v) \in \mathbb{F}_q^m$.

- ***Decryption:*** ***A*** decrypts c as follows:
  - $M = \psi(v)\psi(u)^{-e}$

- ***Torus Based ElGamal Type Signature Scheme***
  - **System Public Parameters:** $G =< \alpha >\subset T_k(\mathbb{F}_q)$, $ord(\alpha) = \ell$, rational maps $\rho,\ \psi$; $\phi(k) = m$, and $H : G \to \mathbb{Z}_\ell$ hash function.
  - **A**'s public key $P_A = \rho(\alpha^e) \in \mathbb{F}_q^m$, private key $e \in \mathbb{Z}_\ell$.
  - ***Assumption***: messages M are in G.
  - ***Signing Process:*** **A** signs message $M \in G$ as follows:
  - Chooses a random integer $r \in \mathbb{Z}_\ell$, computes $u = \rho(\alpha^r) \in \mathbb{F}_q^m$, and $n = r^{-1}(H(M) - eH(\alpha^r)) \pmod{\ell}$.
  - **A**'s signature is (u, n, M).
  - **Verification: B** accepts A's signature on M if and only if $\psi(P_A)^{H(\alpha^r)}\psi(u)^n = \alpha^{H(M)} \in T_k(\mathbb{F}_q)$.

2.8 Trace Based PKC

Over the last few years, efforts were made to improve efficiency of Public Key Cryptosystems (PKCs) in terms of compact representation and increased security. These improvements resulted in form of LUC-PKC [72], GH-PKC [30], XTR-PKC [47], GG-PKC [27] and KK-PKC [41]. In these systems, group operations are carried out in intermediate or prime subfields of an extension field (i.e; $\mathbb{F}_{q^r},\ r \mid k$ or $\mathbb{F}_q$) whereas, security of extension field is ensured. Hence, high security is achieved with operands of lesser number of bits. In contrast, for Diffie-Hellman PKC both group operations and DLP are in same finite field. The Trace Based PKC are discussed in the following chapters.

CHAPTER 3

ALTERNATE MODELS OF ELLIPTIC CURVES

3.1 Introduction

In the recent years, alternate models of elliptic curves have been studied. Such well-known models are Edwards curves, Jacobi intersections and Jacobi quartics, Hessian curves, Huff curves, and their variants to the more common Weierstrass curve. These models sometimes allow for more efficient computation on elliptic curves or provide other features of interest to cryptographers, such as resistance to side-channel attacks. In this chapter, we first give the alternate models of elliptic curves emphasizing point addition and point doubling formulae with computational costs, the suggested improvements in each model and then countermeasures to side channel attacks if any. We also describe the geometric interpretation of the addition law in each model. Note that the contents of this chapter were published in International Journal of Information Security Sciences [6].

From the advent of elliptic curve cryptosystems, independently by Miller (1985) and Koblitz (1987), arithmetic of elliptic curves have been so much interest from cryptographic researchers. Along this period, they proposed many methods to speed up the arithmetic of elliptic curves. These methods can be divided into four different categories:

- Use optimum underlying finite field extensions,
- Use optimum coordinates for representation of group elements,
- Use efficient arithmetic methods,
- Use alternate models of elliptic curves.

In this work, we are dealing with the alternate models of elliptic curves to the more common Weierstrass curve: Edwards curves, Jacobi intersections and Jacobi quartics, Hessian curves, Huff curves, and their variants. These models allow for more efficient computation on elliptic curves such that the group structure of these curves have already been studied in [8], or provide other features of interest to cryptographers, such as resistance to side-channel attacks. These attacks reveal secret information of an elliptic curve cryptosystem based on the point

multiplication operation, in which a point is multiplied by a scalar. The basic method for implementing point multiplication is the *double-and-add technique*, which uses a binary representation of the scalar and performs a sequence of point additions and point doubling depending on the bits of the scalar. In double-and-add point multiplication, a point doubling is done for every bit of the key k, but a point addition is done only when a bit of the key is 1. If, in a side-channel analysis, a point addition is distinguishable from a point doubling, then the bits of the secret key can be determined. This is done by analyzing side channel information such as power consumption [44], running time [43], differential fault analysis [13], electromagnetic emissions [3] and so on. As a countermeasure of this attack, one can use the unified addition formula which means point addition formula that can be used for doubling. The unified formulae for point addition and point doubling use the same sequence of field operations and hence are indistinguishable.

Edwards introduced the normal form of elliptic curves together with an explicit addition law in [23]. He also showed that every elliptic curve over a non-binary field is bi-rationally equivalent to a curve in Edwards form over an extension of the field, and in many cases over the original field. In [7], Bernstein and Lange introduced the notion of Edwards curves which is covering more curves than original Edwards curves when those defined on finite fields. Twisted Edwards curve was introduced by Bernstein et al. in [12] as a generalization of Edwards curves. Hisil et al. introduced the extended twisted Edwards coordinates, and obtained efficient point addition algorithm in [36] that is the fastest one in the literature. These algorithms provide a natural protection from side channel attacks based on Simple Power Analysis (SPA). Bernstein et al. [11] introduced Edwards curves over finite fields of characteristic 2, and obtained addition and doubling formulae.

Jacobi form of the elliptic curves are introduced as the intersection of two quartics in projective space of dimension 3 by Liardet and Smart in [49]. They showed that these curves could provide a defense against Simple and Differential Power Analysis (SPA/DPA) style attacks. The twisted Jacobi intersections which contains Jacobi intersections as a special case is introduced by Feng et al. in [25]. Billet and Joye [14] re-investigated the Jacobi Form suggested by Liardet and Smart. They showed that the addition law is directly derived from the underlying quartics.

Hessian curves are investigated as a step towards resistance against side-channel attacks by Joye and Quisquater in [39]. The efficient arithmetic on Hessian elliptic curves are studied in [70] and [35]. They proposed efficient point multiplication algorithms using the Hessian form over finite fields of characteristic 3. In 2010, the family of generalized Hessian curves are proposed by Farashahi and Joye [24]. They showed that these curves cover more isomorphism classes of elliptic curves and that have efficient unified addition formulas. In [10], Bernstein, Kohel and Lange introduced the twisted Hessian form that is similar to the generalized Hessian curves up to the order of the coordinates.

Huff model of elliptic curve is introduced by Huff [38] over rational fields $\mathbb{Q}$ in 1948. Joye et al. [40] improved these curves to the fields of characteristic different

than 2. They obtained point addition and doubling formulae on Huff curves. The generalized form of Huff curve is introduced by Feng and Wu in [77]. Ciss and Sow [20] investigated the new generalized Huff curve that the addition law in projective coordinates is as fast as in the previous particular cases. In 2011, Devigne and Joye [21] obtained the unified point addition formula for Huff curves over fields of characteristic 2.

This chapter is organized as follows. In section II, we give Weierstrass form of elliptic curves and its group law. In section III, Edwards curves and their variants are discussed. In section IV, Jacobi intersection and Jacobi quartic with related modifications and improvements are discussed. In section V, Hessian curves are elaborated. In section VI, Huff curves and its generalizations are described. In section VII, we compare the alternate models of elliptic curves and conclude the paper.

3.2 Weierstrass Curves

This form of elliptic curve along with cryptographic protocols is discussed in Chapter 1.

For the rest of the chapter, we enumerate the cost of field operations in terms of multiplication **M**, squaring **S**, multiplication by a constant **D**, and addition/subtraction **a** in $\mathbb{F}$.

3.3 Normal Form of Elliptic Curves

In this section, we will discuss salient features of Edwards curves and their variants in respect of point addition and point doubling.

3.3.1 Edwards Curves

Edwards [23] introduced a new model of elliptic curves over $\mathbb{F}$ with $\text{char}(\mathbb{F}) \neq 2$ which is defined by

$$E_c : \ x^2 + y^2 = c^2(1 + x^2y^2), \tag{3.1}$$

where $c \in \mathbb{F}$. He obtained an efficient explicit formula for point addition of these curves as follows: Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on E_c. Then $P + Q = R = (x_3, y_3)$, where

$$\begin{cases} x_3 = \dfrac{x_1y_2 + x_2y_1}{c(1 + x_1x_2y_1y_2)} \\ y_3 = \dfrac{y_1y_2 - x_1x_2}{c(1 - x_1x_2y_1y_2)} \end{cases}$$

Edwards showed that all elliptic curves over non-binary finite field $\mathbb{F}$ can be transformed to Edwards curves if $\mathbb{F}$ is algebraically closed. However, over the finite field $\mathbb{F}$, only a small number of elliptic curves can be expressed in this form.

Bernstein and Lange [7] improved the notion of Edwards form defined by

$$E_d : \ x^2 + y^2 = 1 + dx^2y^2, \tag{3.2}$$

where $d \in \mathbb{F} \setminus \{0, 1\}$. They showed that more than $1/4$ of all isomorphism classes of elliptic curves over the finite field $\mathbb{F}$ could be transformed to Edwards curve over the same field. The curve E_d has an additive group structure together with the identity (neutral) element $\mathcal{O} = (0, 1)$. The point $\mathcal{O}' = (0, -1)$ has order 2. The points $(1, 0)$ and $(-1, 0)$ have order 4.

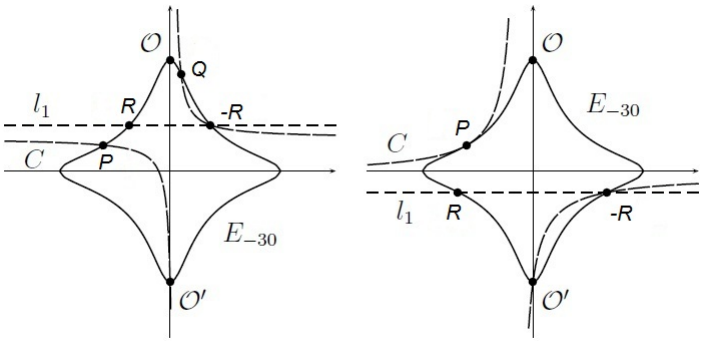

Figure 3.1: Addition and doubling over $\mathbb{R}$ for $d < 0$

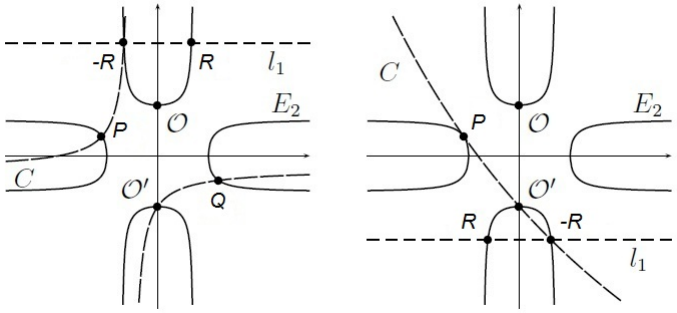

Figure 3.2: Addition and doubling over $\mathbb{R}$ for $d < 1$

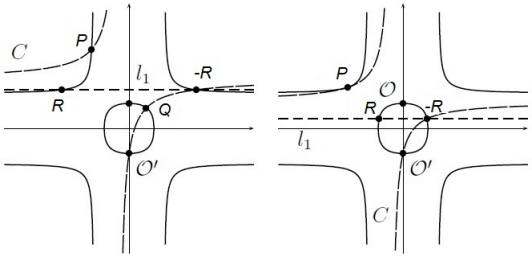

Figure 3.3: Addition and doubling over $\mathbb{R}$ for $0 < d < 1$

The geometric interpretation of the addition law for Edwards curves is given by the following way [4]: We first observe that $\Omega_1 = (1 : 0 : 0)$ and $\Omega_2 = (0 : 1 : 0)$ are the points at infinity that have multiplicity 2. There is a conic C determined by passing through the 5 points $P, Q, \mathcal{O}^{'}, \Omega_1$ and Ω_2 has only one more intersection point $-R$ with the curve E. Let h_1 be the function corresponding to C with $\mathrm{div}(h_1) = (P) + (Q) + (\mathcal{O}^{'}) + (-R) - 2(\Omega_1) - 2(\Omega_2)$. In order to replace $\mathcal{O}^{'}$ by $\mathcal{O}$ and $-R$ by R, one can use another function h_2 that is the product $h_2 = \ell_1\ell_2$ of two lines. A horizontal line ℓ_1 passing through the point R is with $\mathrm{div}(\ell_1) = (R) + (-R) - 2(\Omega_2)$, and a vertical line ℓ_2 passing through the point $\mathcal{O}$ is with $\mathrm{div}(\ell_2) = (\mathcal{O}) + (\mathcal{O}^{'}) - 2(\Omega_1)$. Therefore, the equation $R = P + Q$ corresponds to $\mathrm{div}(h_1/\ell_1\ell_2) = (P) + (Q) - (R) - (\mathcal{O})$.

Using this observation, Bernstein and Lange write down the explicit formula for point addition and point doubling of the curve E_d as follows: Let $P = (x_1, y_1)$

and $Q = (x_2, y_2)$ be two points on E_d. Then $P + Q = R = (x_3, y_3)$, where

$$\begin{cases} x_3 = \dfrac{x_1y_2 + x_2y_1}{1 + dx_1x_2y_1y_2} \\ y_3 = \dfrac{y_1y_2 - x_1x_2}{1 - dx_1x_2y_1y_2} \end{cases}$$

These formulae are strongly unified. If d is a non-square in $\mathbb{F}$, the addition law is complete, i.e, it works for all pairs of inputs. The inverse of the point (x_1, y_1) on E_d is $(-x_1, y_1)$.

In order to avoid the inversion in addition formulae, the notion of Edwards curves in projective coordinates [7] is defined by

$$(X^2 + Y^2)Z^2 = (Z^4 + dX^2Y^2). \tag{3.3}$$

The point addition for (3.3) is obtained by the following formulae: Let $P = (X_1 : Y_1 : Z_1)$ and $Q = (X_2 : Y_2 : Z_2)$ be two points on (3.3), then $P + Q = R = (X_3 : Y_3 : Z_3)$, where

$$\begin{cases} X_3 = Z_1Z_2(Z_1^2Z_2^2 - dX_1X_2Y_1Y_2)[(X_1 + Y_1)(X_2 + Y_2) \\ \qquad -X_1X_2 - Y_1Y_2] \\ Y_3 = Z_1Z_2(Z_1^2Z_2^2 + dX_1X_2Y_1Y_2)(Y_1Y_2 - X_1X_2) \\ Z_3 = (Z_1^2Z_2^2 - dX_1X_2Y_1Y_2)(Z_1^2Z_2^2 + dX_1X_2Y_1Y_2) \end{cases}$$

These formulae are also unified. The point $(0 : 1 : 1)$ is the identity element of addition law. The inverse of $(X_1 : Y_1 : Z_1)$ is $(-X_1 : Y_1 : Z_1)$. The computational cost for addition, doubling, and unified addition is $10\mathbf{M} + 1\mathbf{S} + 1\mathbf{D} + 7\mathbf{a}$, $3\mathbf{M} + 4\mathbf{S} + 6\mathbf{a}$, and $10\mathbf{M} + 1\mathbf{S} + 1\mathbf{D} + 7\mathbf{a}$, respectively. The mixed addition formulae can also be obtained by replacing $Z_2 = 1$ in the above formulae that reduces the total costs to $9\mathbf{M}+1\mathbf{S}+1\mathbf{D}+7\mathbf{a}$. The presence of point of order 4 in the group of elliptic curves in equation (3.3), reduces the number of elliptic curves in Edwards model over $\mathbb{F}$. To overcome this problem Bernstein et al. [12] introduced further variation of Edwards curves which is so called Twisted Edwards curves.

3.3.2 Twisted Edwards Curves

Let $\mathbb{F}$ be a field with char($\mathbb{F}$)$\neq 2$. Then twisted Edwards curve is defined by

$$E_{a,d} : \ ax^2 + y^2 = 1 + dx^2y^2, \tag{3.4}$$

where $a,\ d \in \mathbb{F} \setminus \{0\}$. The twisted Edwards curve $E_{a,d}$ is a quadratic twist of the Edwards curve $E_{1,d/a}$. If a is square in $\mathbb{F}$, then $E_{a,d}$ is isomorphic to $E_{1,d/a}$ over $\mathbb{F}$. The set of these curves is invariant under quadratic twists, in other words, every quadratic twist of a twisted Edwards curve is isomorphic to a twisted Edwards curve. The point addition for (3.4) is obtained by the following formulae: Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on $E_{a,d}$. Then $P + Q = R = (x_3, y_3)$,

where

$$\begin{cases} x_3 = \dfrac{x_1y_2 + x_2y_1}{1 + dx_1x_2y_1y_2} \\ y_3 = \dfrac{y_1y_2 - ax_1x_2}{1 - dx_1x_2y_1y_2} \end{cases}$$

These formulae are unified. The point $(0, 1)$ is the identity element of addition law and the inverse of the point (x_1, y_1) on $E_{a,d}(\mathbb{F})$ is $(-x_1, y_1)$. If a is square in $\mathbb{F}$ and d is non-square in $\mathbb{F}$, then the addition law for Twisted Edwards curve is complete.

In order to avoid inversion in addition formulae given above, twisted Edwards curves in projective coordinates is defined by

$$(aX^2 + Y^2)Z^2 = Z^4 + dX^2Y^2. \tag{3.5}$$

For $Z_1 \neq 0$, the homogeneous point $(X_1 : Y_1 : Z_1)$ represents the affine point $(X_1/Z_1, Y_1/Z_1)$ on $E_{a,d}$. Bernstein et al. [12] obtained the following explicit formulae for addition and doubling on twisted Edwards curves in projective coordinates as follows: Let $P = (X_1 : Y_1 : Z_1)$ and $Q = (X_2 : Y_2 : Z_2)$ be two points on (3.5), then $P + Q = R = (X_3 : Y_3 : Z_3)$, where

$$\begin{cases} X_3 = (X_1Y_2 - Y_1X_2)(X_1Y_1Z_2^2 + X_2Y_2Z_1^2) \\ Y_3 = (Y_1Y_2 + aX_1X_2)(X_1Y_1Z_2^2 - X_2Y_2Z_1^2) \\ Z_3 = Z_1Z_2(X_1Y_2 - Y_1X_2)(Y_1Y_2 + aX_1X_2) \end{cases}$$

and $2P = R = (X_3 : Y_3 : Z_3)$, where

$$\begin{cases} X_3 = (aX_1^2 + Y_1^2 - 2Z_1^2)[(X_1 + Y_1)^2 \\ \qquad -X_1^2 - Y_1^2] \\ Y_3 = (aX_1^2 + Y_1^2)(aX_1^2 - Y_1^2) \\ Z_3 = (aX_1^2 + Y_1^2)(aX_1^2 + Y_1^2 - 2Z_1^2) \end{cases}$$

The computational cost of point addition and point doubling are $11\mathbf{M} + 2\mathbf{D} + 9\mathbf{a}$ and $3\mathbf{M} + 4\mathbf{S} + 1\mathbf{D} + 7\mathbf{a}$, respectively. It turns out that a mixed addition requires $9\mathbf{M} + 2\mathbf{D} + 9\mathbf{a}$ by setting $Z_2 = 1$.

The unified addition formulae for twisted Edwards curves in projective coordinates are also obtained as follows: Let $P = (X_1 : Y_1 : Z_1)$ and $Q = (X_2 : Y_2 : Z_2)$ be two points on (3.5), then $P + Q = R = (X_3 : Y_3 : Z_3)$, where

$$\begin{cases} X_3 = Z_1Z_2(Z_1^2Z_2^2 - dX_1X_2Y_1Y_2)[(X_1 + Y_1) \\ \qquad (X_2 + Y_2) - X_1X_2 - Y_1Y_2] \\ Y_3 = Z_1Z_2(Z_1^2Z_2^2 + dX_1X_2Y_1Y_2)(Y_1Y_2 - aX_1X_2) \\ Z_3 = (Z_1^2Z_2^2 + dX_1X_2Y_1Y_2)(Z_1^2Z_2^2 - dX_1X_2Y_1Y_2) \end{cases}$$

The computational cost of unified addition is $10\mathbf{M} + 1\mathbf{S} + 2\mathbf{D} + 7\mathbf{a}$.

Another way to avoid inversions is to define inverted coordinates as follows:

$$(X^2 + aY^2)Z^2 = X^2Y^2 + dZ^4. \tag{3.6}$$

where $XYZ \neq 0$. The homogeneous point $(X_1 : Y_1 : Z_1)$ with $X_1Y_1Z_1 \neq 0$ represents the affine point $(Z_1/X_1, Z_1/Y_1)$ on $E_{a,d}$. In [9], Bernstein and Lange introduced these inverted coordinates for the case $a = 1$, and observed that the coordinates save time in addition. Bernstein et al. generalized to arbitrary a in [12]. They also obtained the following explicit formulae for unified addition and doubling on twisted Edwards curves in inverted coordinates as follows: Let $P = (X_1 : Y_1 : Z_1)$ and $Q = (X_2 : Y_2 : Z_2)$ be two points on (3.6), then $P + Q = R = (X_3 : Y_3 : Z_3)$, where

$$\begin{cases} X_3 = Z_1Z_2(X_1X_2 + aY_1Y_2)(X_1Y_1Z_2^2 - Z_1^2X_2Y_2) \\ Y_3 = Z_1Z_2(X_1Y_2 - Y_1X_2)(X_1Y_1Z_2^2 + Z_1^2X_2Y_2) \\ Z_3 = (X_1Y_1Z_2^2 - Z_1^2X_2Y_2)(X_1Y_1Z_2^2 - Z_1^2X_2Y_2) \end{cases}$$

and $2P = R = (X_3 : Y_3 : Z_3)$, where

$$\begin{cases} X_3 = (X_1^2 + aY_1^2)(X_1^2 - aY_1^2) \\ Y_3 = [(X_1 + Y_1)^2 - X_1^2 - Y_1^2](X_1^2 + aY_1^2 - 2dZ_1^2) \\ Z_3 = (X_1^2 - aY_1^2)[(X_1 + Y_1)^2 - X_1^2 - Y_1^2] \end{cases}$$

The unified addition formulae for twisted Edwards curves in inverted coordinates are also obtained as follows: Let $P = (X_1 : Y_1 : Z_1)$ and $Q = (X_2 : Y_2 : Z_2)$ be two points on (3.5), then $P + Q = R = (X_3 : Y_3 : Z_3)$, where

$$\begin{cases} X_3 = (X_1X_2Y_1Y_2 + dZ_1^2Z_2^2)(X_1X_2 - aY_1Y_2) \\ Y_3 = (X_1X_2Y_1Y_2 - dZ_1^2Z_2^2)[(X_1 + Y_1)(X_2 + Y_2) \\ \qquad -X_1X_2 - Y_1Y_2] \\ Z_3 = Z_1Z_2(X_1X_2 - aY_1Y_2)[(X_1 + Y_1)(X_2 + Y_2) \\ \qquad -X_1X_2 - Y_1Y_2] \end{cases}$$

The computational cost of point addition, point doubling and unified addition are $11\mathbf{M}+2\mathbf{D}+9\mathbf{a}$, $3\mathbf{M}+4\mathbf{S}+2\mathbf{D}+6\mathbf{a}$, and $9\mathbf{M}+1\mathbf{S}+2\mathbf{D}+7\mathbf{a}$, respectively. The mixed addition formulae can also be obtained by replacing $Z_2 = 1$, which gives an obvious saving of $2\mathbf{M}$ since $Z_1 \cdot Z_2 = Z_1$, leading to a total cost of $9\mathbf{M} + 2\mathbf{D} + 9\mathbf{a}$.

Hisil et al. [36] introduced the extended Twisted Edwards coordinates by defining an auxiliary coordinate $t = xy$ to represent a point (x, y) on $E_{a,d}$ in extended affine coordinates (x, y, t). One can pass to the projective representation $(X : Y : T : Z)$ which satisfies (3.5) and corresponds to the extended affine point $(X/Z, Y/Z, T/Z)$ with $Z \neq 0$. The auxiliary coordinate T has the property $T = XY/Z$. Let $P = (X_1 : Y_1 : T_1 : Z_1)$ and $Q = (X_2 : Y_2 : T_2 : Z_2)$ be two points on (3.5) with $Z_1 \neq 0$ and $Z_2 \neq 0$, then $P + Q = R = (X_3 : Y_3 : T_3 : Z_3)$, where

$$\begin{cases} X_3 = (X_1Y_2 + Y_1X_2)(Z_1Z_2 - dT_1T_2) \\ Y_3 = (Y_1Y_2 - aX_1X_2)(Z_1Z_2 + dT_1T_2) \\ T_3 = (Y_1Y_2 - aX_1X_2)(X_1Y_2 + Y_1X_2) \\ Z_3 = (Z_1Z_2 - dT_1T_2)(Z_1Z_2 + dT_1T_2) \end{cases}$$

These formulae are unified that derived from the addition formulae on $E_{a,d}$. It is deduced from [7] and [12] that these formulae are also complete when d is not a square in $\mathbb{F}$ and a is a square in $\mathbb{F}$. The identity element is represented by $(0:1:0:1)$. The negative of $(X_1:Y_1:T_1:Z_1)$ on (3.5) is $(-X_1:Y_1:-T_1:Z_1)$. The computational cost of point addition, point doubling and unified addition are $9\mathbf{M}+1\mathbf{D}+7\mathbf{a}$, $4\mathbf{M}+4\mathbf{S}+1\mathbf{D}+7\mathbf{a}$, and $9\mathbf{M}+2\mathbf{D}+7\mathbf{a}$, respectively. The mixed addition formulae can also be obtained by setting $Z_2=1$ in the above formulae, reduces the total costs to $8\mathbf{M}+1\mathbf{D}+7\mathbf{a}$. This means that one can add $(X_1:Y_1:T_1:Z_1)$ and an extended affine point (x_2,y_2,x_2y_2), which is equally written as $(x_2:y_2:x_2y_2:1)$.

3.3.3 Binary Edwards Curves

Let $\mathbb{F}$ be a field with char($\mathbb{F}$)$=2$. Then Binary Edwards curve is defined by

$$E_{B,d_1,d_2}:\ d_1(x+y)+d_2(x^2+y^2)=xy+xy(x+y)+x^2y^2,$$

where $d_1\neq 0$ and $d_2\neq d_1^2+d_1$. The point addition is obtained by the following formulae: Let $P=(x_1,y_1)$ and $Q=(x_2,y_2)$ be two points on E_{B,d_1,d_2}. Then $P+Q=R=(x_3,y_3)$, where

$$\begin{cases} x_3=\dfrac{d_1(x_1+x_2)+d_2(x_1+y_1)(x_2+y_2)}{d_1+(x_1+x_1^2)(x_2+y_2)} \\ \qquad\quad +\dfrac{(x_1+x_1^2)(x_2(y_1+y_2+1)+y_1y_2)}{d_1+(x_1+x_1^2)(x_2+y_2)} \\ y_3=\dfrac{d_1(y_1+y_2)+d_2(x_1+y_1)(x_2+y_2)}{d_1+(y_1+y_1^2)(x_2+y_2)} \\ \qquad\quad +\dfrac{(y_1+y_1^2)(y_2(x_1+x_2+1)+x_1x_2)}{d_1+(y_1+y_1^2)(x_2+y_2)} \end{cases}$$

The addition law on E_{B,d_1,d_2} is strongly unified. The point $(0,0)$ is the identity element of addition law and the inverse of the point (x_1,y_1) on E_{B,d_1,d_2} is (y_1,x_1). The computational cost of addition and doubling in projective coordinates are $21\mathbf{M}+1\mathbf{S}+4\mathbf{D}$ and $2\mathbf{M}+6\mathbf{S}+3\mathbf{D}$, respectively. When $t^2+t+d_2\neq 0$ for all $t\in\mathbb{F}$, the addition law on the binary Edwards curve $E_{B,d_1,d_2}(\mathbb{F})$ is complete. The mixed addition formulae lead to a total cost of $13\mathbf{M}+3\mathbf{S}+3\mathbf{D}$ that can be obtained by $(X_3:Y_3:Z_3)=(X_1:Y_1:Z_1)+(x_2,y_2)$, where $(X_1:Y_1:Z_1)$ and (x_2,y_2) on $E_{B,d_1,d_2}(\mathbb{F})$. Bernstein et al. introduced different methods for computing point addition and point doubling with that of computational costs in [11].

3.4 Jacobi Curves

Jacobi curves gained special attention due to resistance against SPA attacks. In this section, the various forms of Jacobi curves are discussed in respect of point addition and point doubling.

3.4.1 Jacobi Intersections

Liardet and Smart [49] introduced the Jacobi Intersections over the finite field $\mathbb{F}$ with $\text{char}(\mathbb{F}) \neq 2$ which are defined by

$$E_{J,b} : \begin{cases} x^2 + y^2 = 1 \\ bx^2 + t^2 = 1 \end{cases} \tag{3.7}$$

where $b \in \mathbb{F}$ and $b(1-b) \neq 0$. They obtained an explicit unified formulae for point addition on $E_{J,b}$ as follows: Let $P = (x_1, y_1, t_1)$ and $Q = (x_2, y_2, t_2)$ be two points on $E_{J,b}$. Then $P + Q = R = (x_3, y_3, t_3)$, where

$$\begin{cases} x_3 = \dfrac{x_1y_2t_2 + x_2y_1t_1}{y_2^2 + x_2^2t_1^2} \\ y_3 = \dfrac{y_1y_2 - x_1t_1x_2t_2}{y_2^2 + x_2^2t_1^2} \\ t_3 = \dfrac{t_1t_2 - bx_1y_1x_2y_2}{y_2^2 + x_2^2t_1^2} \end{cases}$$

and $2P = R = (x_3, y_3)$, where

$$\begin{cases} x_3 = \dfrac{2x_1y_1t_1}{y_1^2 + x_1^2t_1^2} \\ y_3 = \dfrac{y_1^2 - x_1^2t_1^2}{y_1^2 + x_1^2t_1^2} \\ t_3 = \dfrac{t_1^2 - bx_1^2y_1^2}{y_1^2 + x_1^2t_1^2} \end{cases}$$

The point $(0, 1, 1)$ is the identity element of addition law and the inverse of the point (x_1, y_1, t_1) on $E_{J,b}$ is $(-x_1, y_1, t_1)$.

The affine version of Jacobi Intersections has inherent greater computational cost due to the field inversion involved in addition formulae. In order to avoid inversions in addition formulae, Jacobi Intersections in projective coordinates [49] is defined by

$$E_{J,b} : \begin{cases} X^2 + Y^2 = Z^2 \\ bX^2 + T^2 = Z^2 \end{cases} \tag{3.8}$$

with the map $(x, y, t) = (X/Z, Y/Z, T/Z) \mapsto (X : Y : T : Z)$ for $Z \neq 0$. The unified point addition for (3.8) is obtained by the following formulae: Let $P = (X_1 : Y_1 : T_1 : Z_1)$ and $Q = (X_2 : Y_2 : T_2 : Z_2)$ be two points on (3.8), then $P + Q = R = (X_3 : Y_3 : T_3 : Z_3)$, where

$$\begin{cases} X_3 = X_1Z_1Y_2T_2 + Y_1T_1X_2Z_2 \\ Y_3 = Y_1Z_1Y_2Z_2 - X_1T_1X_2T_2 \\ T_3 = T_1Z_1T_2Z_2 - bX_1Y_1X_2Y_2 \\ Z_3 = Z_1^2Y_2^2 + X_2^2T_1^2 \end{cases}$$

The point $(0:1:1:1)$ is the identity element of addition law and the inverse of the point $(X_1:Y_1:T_1:Z_1)$ on $E_{J,b}$ is $(-X_1:Y_1:T_1:Z_1)$. There are three points of order 2, namely, $(0:1:1:1)$, $(0:1:1:1)$ and $(0:1:1:1)$. In this case, the computational cost of point addition, point doubling, and unified addition are $13\mathbf{M}+2\mathbf{S}+1\mathbf{D}+7\mathbf{a}$, $4\mathbf{M}+3\mathbf{S}+5\mathbf{a}$, and $13\mathbf{M}+2\mathbf{S}+1\mathbf{D}+7\mathbf{a}$, respectively. The mixed addition formulae can also be obtained by replacing $Z_2=1$ in the above formulae that reduces the total costs to $11\mathbf{M}+2\mathbf{S}+1\mathbf{D}+7\mathbf{a}$.

3.4.2 Twisted Jacobi Intersections

In [25], the twisted Jacobi Intersections which contains Jacobi intersections as a special case was introduced by Feng et al. These curves encompass more number of elliptic curves and have explicit formulae for addition and doubling with almost as fast as the Jacobi Intersections. The twisted Jacobi Intersections over $\mathbb{F}$ with $\mathrm{char}(\mathbb{F})\neq 2$ are defined by

$$E_{J,a,b}: \begin{cases} ax^2+y^2=1 \\ bx^2+t^2=1 \end{cases} \tag{3.9}$$

where $a,b\in\mathbb{F}$ and $ab(a-b)\neq 0$. They obtained an explicit unified addition formulae on $E_{J,a,b}$ as follows: Let $P=(x_1,y_1,t_1)$ and $Q=(x_2,y_2,t_2)$ be two points on $E_{J,a,b}$. Then $P+Q=R=(x_3,y_3,t_3)$, where

$$\begin{cases} x_3=\dfrac{x_1y_2t_2+x_2y_1t_1}{y_2^2+ax_1^2t_1^2} \\ y_3=\dfrac{y_1y_2-ax_1t_1x_2t_2}{y_2^2+ax_1^2t_1^2} \\ t_3=\dfrac{t_1t_2-bx_1y_1x_2y_2}{y_2^2+ax_1^2t_1^2} \end{cases}$$

The point $(0,1,1)$ is the identity element of addition law and the inverse of the point (x_1,y_1,t_1) on $E_{J,a,b}$ is $(-x_1,y_1,t_1)$.

The twisted Jacobi Intersections in projective coordinates [49] is defined by

$$E_{J,a,b}: \begin{cases} aX^2+Y^2=Z^2 \\ bX^2+T^2=Z^2 \end{cases} \tag{3.10}$$

with the map $(x,y,t)=(X/Z,Y/Z,T/Z)\mapsto(X:Y:T:Z)$ for $Z\neq 0$. The unified point addition on (3.10) is obtained by the following formulae: Let $P=(X_1:Y_1:T_1:Z_1)$ and $Q=(X_2:Y_2:T_2:Z_2)$ be two points on (3.10), then $P+Q=R=(X_3:Y_3:T_3:Z_3)$, where

$$\begin{cases} X_3=X_1Z_1Y_2T_2+Y_1T_1X_2Z_2 \\ Y_3=Y_1Z_1Y_2Z_2-aX_1T_1X_2T_2 \\ T_3=T_1Z_1T_2Z_2-bX_1Y_1X_2Y_2 \\ Z_3=Z_1^2Y_2^2+aX_2^2T_1^2 \end{cases}$$

The point $(0:1:1:1)$ is the identity element of addition law and the inverse of the point $(X_1 : Y_1 : T_1 : Z_1)$ on $E_{J,a,b}$ is $(-X_1 : Y_1 : T_1 : Z_1)$. In this case, the computational cost of addition, doubling, and unified addition are $12\mathbf{M} + 11\mathbf{a}$, $3\mathbf{M} + 4\mathbf{S} + 1\mathbf{D} + 7\mathbf{a}$, and $13\mathbf{M} + 2\mathbf{S} + 5\mathbf{D} + 13\mathbf{a}$, respectively. For a mixed point addition, the number of required multiplications drops to $10\mathbf{M} + 11\mathbf{a}$.

3.4.3 Jacobi Quartics

Jacobi Quartic [14] curves over the finite field $\mathbb{F}$ with $\text{char}(\mathbb{F}) \neq 2, 3$ are first defined by

$$E_{J,k} \ : \ y^2 = k^2x^4 - (k^2 + 1)x^2 + 1, \tag{3.11}$$

where $k \neq 0, \pm 1$. As usual to improve the security parameter that is to increase number of elliptic curves, a modification in Jacobi Quartic was introduced by Hisil et al. in [37]. The modified Jacobi Quartic curves are so called extended Jacobi Quartic curves which are defined by

$$E_{J,d,a} \ : \ y^2 = dx^4 + 2ax^2 + 1, \tag{3.12}$$

where $a, d \in \mathbb{F}$ with $\text{char}(\mathbb{F}) \neq 2, 3$. They showed that arithmetic on Jacobi Quartics is faster as compared with Jacobi Intersections. Moreover, unified point addition formulae offer additional security against some side channel attacks as well. The curve $E_{J,d,a}$ has an additive group structure together with the identity element $\mathcal{O} = (0, 1)$. Note the fact $\mathcal{O}' = (0, -1)$ is a point on the curve.

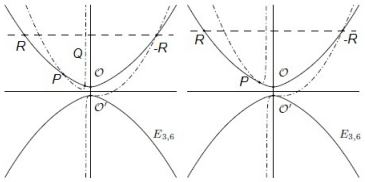

Figure 3.4: Addition and doubling over $\mathbb{R}$

The geometric interpretation of the addition law for Jacobi Quartics is shown in the following [76]: We first observe that there is a singular point $\Omega = (0 : 1 : 0)$ in projective space, which is a point at infinity in affine plane. Let C be a conic passing through the points $P, Q, \mathcal{O}'$ and $-R$ with $\text{div}(C) = (P) + (Q) + (-R) + 3(\mathcal{O}') - 6(\Omega)$. Let ℓ be the vertical line passing through the points $\mathcal{O}$ and $\mathcal{O}'$, then $\text{div}(\ell) = (\mathcal{O}) + (\mathcal{O}') - 2(\Omega)$. Let f_R be a function with $\text{div}(f_R) = (R) + (-R) - 2(\mathcal{O})$. Therefore, the equation $R = P + Q$ corresponds to $\text{div}(C/f_R\ell^3) = (P) + (Q) - (R) - (\mathcal{O})$.

Using this observation, the explicit formulae for point addition and point doubling of the curve $E_{J,d,a}$ are adapted from [37] as follows: Let $P = (x_1, y_1)$ and $Q =$

(x_2, y_2) be two points on $E_{J,d,a}$, then $P + Q = R = (x_3, y_3)$, where

$$\begin{cases} x_3 = \dfrac{x_1y_2 + y_1x_2}{1 - dx_1^2x_2^2} \\ y_3 = \dfrac{(y_1y_2 + 2ax_1x_2)(1 + dx_1^2x_2^2)}{(1 - dx_1^2x_2^2)^2} \\ \qquad + \dfrac{2dx_1x_2(x_1^2 + x_2^2)}{(1 - dx_1^2x_2^2)^2} \end{cases}$$

and $2P = R = (x_3, y_3)$, where

$$\begin{cases} x_3 = \delta x_1 \\ y_3 = \delta(\delta - y_1) - 1 \end{cases}$$

where $\delta = 2y_1/(2 + 2ax_1^2 - y_1^2)$. The inverse of the point (x_1, y_1) on $E_{J,d,a}$ is $(-x_1, y_1)$.

In order to avoid inversion in addition formulae given above, the extended Jacobi Quartic curves in Jacobian coordinates with $x = X/Z$ and $y = Y/Z^2$ are defined by

$$Y^2 = dX^4 - 2aX^2Z^2 + Z^4, \tag{3.13}$$

where $a, d \in \mathbb{F}$ with $\text{char}(\mathbb{F}) \neq 2, 3$. Billet and Joye [14] proposed a faster inversion-free unified addition algorithm on (3.13) as follows: Let $P = (X_1 : Y_1 : Z_1)$ and $Q = (X_2 : Y_2 : Z_2)$ be two points on (3.13), then $P + Q = R = (X_3 : Y_3 : Z_3)$, where

$$\begin{cases} X_3 = X_1Y_2Z_1 + X_2Y_1Z_2 \\ Y_3 = [(Z_1Z_2)^2 + d(X_1X_2)^2](Y_1Y_2 - 2aX_1X_2Z_1Z_2) \\ \qquad +2dX_1X_2Z_1Z_2(X_1^2Z_2^2 + Z_1^2X_2^2) \\ Z_3 = (Z_1Z_2)^2 - d(X_1X_2)^2 \end{cases}$$

The identity element of addition law is given by $(0 : 1 : 1)$ and the negative of the point $(X_1 : Y_1 : Z_1)$ on (3.13) is $(-X_1 : Y_1 : Z_1)$. The computational cost of addition, doubling, and unified addition are $10\mathbf{M} + 3\mathbf{S} + 1\mathbf{D}$, $1\mathbf{M} + 9\mathbf{S} + 1\mathbf{D}$, and $10\mathbf{M} + 3\mathbf{S} + 1\mathbf{D}$, respectively. The total cost of point addition reduces to $8\mathbf{M} + 3\mathbf{S} + 1\mathbf{D}$ by replacing $Z_2 = 1$.

An other improvement was obtained by Hisil et al. in [37]. They showed that the extended Jacobi Quartic curves in extended projective coordinates was defined by

$$\begin{cases} X^2 - TZ = 0 \\ Y^2 - dT^2 - 2aX^2 - Z^2 = 0 \end{cases} \tag{3.14}$$

or simply

$$Y^2Z^2 = dX^4 + 2aX^2Z^2 + Z^4, \tag{3.15}$$

where T is omitted in the latter case. In this case, a point $(x, y) \in E_{J,d,a}(\mathbb{F})$ corresponds to the point $(X : Y : T : Z)$, where $T = X^2/Z$. The identity

element is represented by $(0:1:0:1)$ and negative of $(X_1 : Y_1 : T_1 : Z_1)$ on (3.15) is $(-X_1 : Y_1 : T_1 : Z_1)$. They obtained the following explicit formulae for addition and doubling on the extended Jacobi Quartic curves in extended projective coordinates as follows: Let $P = (X_1 : Y_1 : T_1 : Z_1)$ and $Q = (X_2 : Y_2 : T_2 : Z_2)$ be two points on (3.15), then $P + Q = R = (X_3 : Y_3 : T_3 : Z_3)$ with $Z_1 \neq 0$, $Z_2 \neq 0$ and $P \neq Q$, where

$$\begin{cases} X_3 = (X_1Y_2 - Y_1X_2)(T_1Z_2 - Z_1T_2) \\ Y_3 = (T_1Z_2 + Z_1T_2 - 2X_1X_2)(Y_1Y_2 - 2aX_1X_2 \\ \qquad +Z_1Z_2 + dT_1T_2) - Z_3 \\ T_3 = (T_1Z_2 - Z_1T_2)^2 \\ Z_3 = (X_1Y_2 - Y_1X_2)^2 \end{cases}$$

and $2P = R = (X_3 : Y_3 : T_3 : Z_3)$, where

$$\begin{cases} X_3 = X_1Y_2Z_1 + X_2Y_1Z_2 \\ Y_3 = [(Z_1Z_2)^2 + d(X_1X_2)^2](Y_1Y_2 - 2aX_1X_2Z_1Z_2) \\ \qquad +2dX_1X_2Z_1Z_2(X_1^2Z_2^2 + Z_1^2X_2^2) \\ T_3 = (2X_1Y_1)^2 \\ Z_3 = (Z_1Z_2)^2 - d(X_1X_2)^2 \end{cases}$$

If $a = -1/2$, the computational cost of point addition and point doubling are $7\mathbf{M} + 3\mathbf{S} + 2\mathbf{D}$ and $8\mathbf{S}$, respectively. The total cost of point addition reduces to $6\mathbf{M} + 3\mathbf{S} + 2\mathbf{D}$ by replacing $Z_2 = 1$.

They also obtained the unified addition formulae for extended Jacobi Quartics in extended projective coordinates as follows: Let $P = (X_1 : Y_1 : T_1 : Z_1)$ and $Q = (X_2 : Y_2 : T_2 : Z_2)$ be two points on (3.15), then $P + Q = R = (X_3 : Y_3 : T_3 : Z_3)$ with $Z_1 \neq 0$ and $Z_2 \neq 0$, where

$$\begin{cases} X_3 = (X_1Y_2 + Y_1X_2)(Z_1Z_2 - dZ_1T_2) \\ Y_3 = (Y_1Y_2 + 2aX_1X_2)(Z_1Z_2 + dT_1T_2) \\ \qquad +2dX_1X_2(T_1Z_2 + Z_1T_2) \\ T_3 = (X_1Y_2 + Y_1X_2)^2 \\ Z_3 = (Z_1Z_2 - dT_1T_2)^2 \end{cases}$$

If d is not a square in $\mathbb{F}$, then the unified addition formulae are complete. The computational cost is $8\mathbf{M} + 3\mathbf{S} + 2\mathbf{D} + 17\mathbf{a}$ when $a = -1/2$. More on formulae and operation counts can be found Appendix B in [37].

3.5 Hessian Curves

In [39], Hessian elliptic curves are investigated by Joye and Quisquater. They obtained the formulae of point addition, point doubling and unified addition.

The Hessian elliptic curve over $\mathbb{F}$ with $\text{char}(\mathbb{F}) \neq 2, 3$ is a plane cubic curve given by

$$H_d \ : \ x^3 + y^3 + 1 = 3dxy, \tag{3.16}$$

or in projective coordinates,

$$H_d \ : \ X^3 + Y^3 + Z^3 = 3dXYZ, \tag{3.17}$$

where $d \in \mathbb{F}$ and $d^3 \neq 1$. The curve H_d has an additive group structure together with the identity element $\mathcal{O} = (1 : -1 : 0)$. The points $(0 : 1 : -1)$ and $(1 : 0 : -1)$ are two points of order 3.

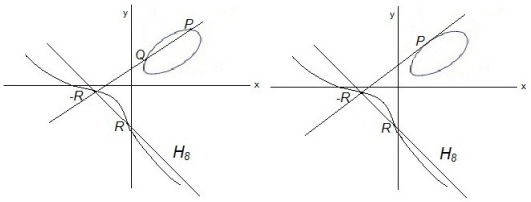

Figure 3.5: Addition and doubling over $\mathbb{R}$

The geometric interpretation of the addition law for Hessian curves is given by the following way [34]: For $P, Q \in H_d$, let ℓ_1 be the line passing through the points P and Q. Then the divisor of L is $\text{div}(\ell_1) = (P) + (Q) + (-R) - 3(\mathcal{O})$. Let ℓ_2 be the line passing through the points $-R$ and R. Then $\text{div}(\ell_2) = (R) + (-R) - 2(\mathcal{O})$. Therefore, the equation $R = P + Q$ corresponds to $\text{div}(\ell_1/\ell_2) = (P) + (Q) - (R) - (\infty)$.

This observation allows us to write down the explicit formula for point addition and point doubling of the curve H_d as follows [39]: Let $P = (X_1 : Y_1 : Z_1)$ and $Q = (X_2 : Y_2 : Z_2)$ be two points on (3.17), then $P + Q = R = (X_3 : Y_3 : Z_3)$, where

$$\begin{cases} X_3 = Y_1^2 X_2 Z_2 - Y_2^2 X_1 Z_1 \\ Y_3 = X_1^2 Y_2 Z_2 - X_2^2 Y_1 Z_1 \\ Z_3 = Z_1^2 Y_2 X_2 - Z_2^2 Y_1 X_1 \end{cases}$$

and $2P = R = (X_3 : Y_3 : Z_3)$, where

$$\begin{cases} X_3 = Y_1(Z_1^3 - X_1^3) \\ Y_3 = X_1(Y_1^3 - Z_1^3) \\ Z_3 = Z_1(X_1^3 - Y_1^3) \end{cases}$$

The inverse of the point $(X_1 : Y_1 : Z_1)$ on H_d is $(Y_1 : X_1 : Z_1)$. Owing to the formulae $2(X_1 : Y_1 : Z_1) = (Z_1 : X_1 : Y_1) + (Y_1 : Z_1 : X_1)$ and $(X_1 : Y_1 : Z_1) - (X_2 : Y_2 : Z_2) = (X_1 : Y_1 : Z_1) + (Y_2 : X_2 : Z_2)$, the following addition algorithm given by Joye and Quisquater in [39] can be used also doubling and subtraction as well

as addition: Let $P = (X_1 : Y_1 : Z_1)$ and $Q = (X_2 : Y_2 : Z_2)$ be two points on (3.17), then $P + Q = R = (X_3 : Y_3 : Z_3)$, where

$$\begin{aligned}
\text{Step 1} &\begin{cases} L_1 \leftarrow X_1;\ L_2 \leftarrow Y_1;\ L_3 \leftarrow Z_1 \\ L_4 \leftarrow X_2;\ L_5 \leftarrow Y_2;\ L_6 \leftarrow Z_2 \end{cases} \\
\text{Step 2} &\begin{cases} L_7 \leftarrow L_1 \cdot L_6;\ L_1 \leftarrow L_1 \cdot L_5 \\ L_5 \leftarrow L_3 \cdot L_5;\ L_3 \leftarrow L_3 \cdot L_4 \\ L_4 \leftarrow L_2 \cdot L_4;\ L_2 \leftarrow L_2 \cdot L_6 \end{cases} \\
\text{Step 3} &\begin{cases} L_6 \leftarrow L_2 \cdot L_7;\ L_2 \leftarrow L_2 \cdot L_4 \\ L_4 \leftarrow L_3 \cdot L_4;\ L_3 \leftarrow L_3 \cdot L_5 \\ L_5 \leftarrow L_1 \cdot L_5;\ L_1 \leftarrow L_1 \cdot L_7 \end{cases} \\
\text{Step 4} &\begin{cases} X_3 \leftarrow L_2 - L_5;\ Y_3 \leftarrow L_1 - L_4 \\ Z_3 \leftarrow L_3 - L_6 \end{cases}
\end{aligned}$$

The computational cost of these operations are $12\mathbf{M}+3\mathbf{a}$. The total cost of point addition reduces to $10\mathbf{M} + 3\mathbf{a}$ by replacing $Z_2 = 1$.

More recently, Farashahi and Joye [24] considered a generalized form of Hessian curves that covers more isomorphism classes of elliptic curves. These curves are similar to the twisted Hessian form [10], introduced by Bernstein, Kohel and Lange, up to the order of the coordinates. The generalized Hessian curve over $\mathbb{F}$ is defined by

$$H_{c,d} \ : \ x^3 + y^3 + c = dxy, \tag{3.18}$$

where $c, d \in \mathbb{F}$ with $c \neq 0$ and $d^3 \neq 27c$. A generalized Hessian curve over $\mathbb{F}$ is isomorphic over $\mathbb{F}$ to a Hessian curve if and only if c is a cube in $\mathbb{F}$. It is easy to adapt the addition and doubling formulae for generalized Hessian curves which are so-called Sylvester formulas as follows: Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on $H_{c,d}$, then $P + Q = R = (x_3, y_3)$, where

$$\begin{cases} x_3 = \dfrac{y_1^2 x_2 - y_2^2 x_1}{x_2 y_2 - x_1 y_1} \\ y_3 = \dfrac{x_1^2 y_2 - x_2^2 y_1}{x_2 y_2 - x_1 y_1} \end{cases}$$

and $2P = R = (x_3, y_3)$, where

$$\begin{cases} x_3 = \dfrac{y_1(c - x_1^3)}{x_1^3 - y_1^3} \\ y_3 = \dfrac{x_1(c - y_1^3)}{x_1^3 - y_1^3} \end{cases}$$

Furthermore, the inverse of the point (x_1, y_1) on $H_{c,d}$ is the point (y_1, x_1). The generalized Hessian curves in projective coordinates are defined by

$$H_{c,d} \ : \ X^3 + Y^3 + cZ^3 = dXYZ. \tag{3.19}$$

The point $(1 : -1 : 0)$ is identity element and the inverse of the point $(X_1 : Y_1 : Z_1)$ on $H_{c,d}$ is $(Y_1 : X_1 : Z_1)$. They obtained point addition and doubling formulae on generalized Hessian curves in projective coordinates as follows: Let $P = (X_1 : Y_1 : Z_1)$ and $Q = (X_2 : Y_2 : Z_2)$ be two points on $H_{c,d}$, then $P + Q = R = (X_3 : Y_3 : Z_3)$, where

$$\begin{cases} X_3 = X_2Z_2Y_1^2 - X_1Z_1Y_2^2 \\ Y_3 = Y_2Z_2X_1^2 - Y_1Z_1X_2^2 \\ Z_3 = X_2Y_2Z_1^2 - X_1Y_1Z_2^2 \end{cases}$$

and $2P = R = (X_3 : Y_3 : Z_3)$, where

$$\begin{cases} X_3 = Y_1(cZ_1^3 - X_1^3) \\ Y_3 = X_1(Y_1^3 - cZ_1^3) \\ Z_3 = Z_1(X_1^3 - Y_1^3) \end{cases}$$

In this case, the computational cost of point addition is $4\mathbf{M}$, $3\mathbf{M}$, or $2\mathbf{M}$ correspond to use of $3, 4$ or 6 processors, respectively. The point addition formulae are complete if the difference of all pairs of points on $H_{c,d}$ is not equal the identity. The cost of point doubling is $6\mathbf{M} + 3\mathbf{S} + 1\mathbf{D}$. The point doubling formulae are complete for all inputs.

The unified addition formulae for generalized Hessian curves in projective coordinates are also obtained as follows: Let $P = (X_1 : Y_1 : Z_1)$ and $Q = (X_2 : Y_2 : Z_2)$ be two points on $H_{c,d}$, then $P + Q = R = (X_3 : Y_3 : Z_3)$, where

$$\begin{cases} X_3 = cY_2Z_2Z_1^2 - X_1Y_1X_2^2 \\ Y_3 = X_2Y_2Y_1^2 - cX_1Z_1Z_2^2 \\ Z_3 = X_2Z_2X_1^2 - Y_1Z_1Y_2^2 \end{cases}$$

The computational cost of unified point addition is $12\mathbf{M} + 1\mathbf{D}$. The unified point addition formulae on $H_{c,d}$ are complete if c is not a cube in $\mathbb{F}$. It turns out that a mixed addition requires $10\mathbf{M} + 1\mathbf{D}$ by setting $Z_2 = 1$.

Farashahi and Joye [24] obtained point addition, doubling and tripling formulae for binary generalized Hessian curves with the computational cost of them. Differential addition, that is, point addition with a known difference was also devised for binary Hessian curves by them.

3.6 Huff Model of Elliptic Curves

In this section, we will give the details of the models of Huff curves, especially their group structure and related formulae with associated computational costs.

3.6.1 Huff Curves

In [38], Huff investigated the Huff elliptic curves over rational fields $\mathbb{Q}$ in 1948. Joye et al. [40] improved these curves to the finite field $\mathbb{F}$ with $\text{char}(\mathbb{F}) \neq 2$ that are given by

$$E_{a,b} \ : \ ax(y^2 - 1) = by(x^2 - 1), \tag{3.20}$$

where $a, b \neq 0$ and $a^2 - b^2 \neq 0$. The unified point addition for (3.20) is obtained by the following formulae: Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on (3.20). Then $P + Q = R = (x_3, y_3)$, where

$$\begin{cases} x_3 = \dfrac{(x_1 + x_2)(1 + y_1y_2)}{(1 + x_1x_2)(1 - y_1y_2)} \\ y_3 = \dfrac{(y_1 + y_2)(1 + x_1x_2)}{(1 - x_1x_2)(1 + y_1y_2)} \end{cases}$$

These formulae are complete whenever $x_1x_2 \neq \pm 1$ and $y_1y_2 \neq \pm 1$. The Huff model of elliptic curves in projective coordinates are defined by

$$aX(Y^2 - Z^2) = bY(X^2 - Z^2), \tag{3.21}$$

where $a, b \neq 0$ and $a^2 - b^2 \neq 0$. Huff curves has an additive group structure together with the identity element $\mathcal{O} = (0 : 0 : 1)$. We note that a point at infinity is its own inverse. Hence, there are three points at infinity, namely, $(1 : 0 : 0)$, $(0 : 1 : 0)$ and $(a : b : 0)$. The sum of any two of them is equal to the third one.

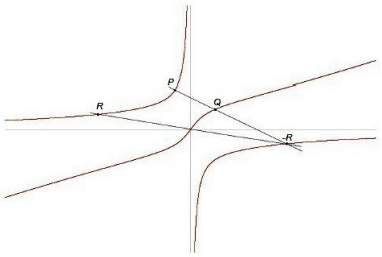

Figure 3.6: Addition over $\mathbb{R}$

The geometric interpretation of the addition law for Huff curves is given by the following way [40]: For $P, Q \in E_{a,b}$, let ℓ be the rational function passing through the points P and Q with $\text{div}(\ell) = (P)+(Q)+(-R)-(1 : 0 : 0)-(0 : 1 : 0)-(a : b : 0)$, where $-R$ is the third point of intersection of the line ℓ with the elliptic curve. The neutral element of the group law is $\mathcal{O} = (0 : 0 : 1)$. Let f be the line function with $\text{div}(f) = (R) + (-R) + (\mathcal{O}) - (1 : 0 : 0) - (0 : 1 : 0) - (a : b : 0)$. Therefore, the equation $R = P + Q$ corresponds to $\text{div}(\ell/f) = (P) + (Q) - (R) - (\mathcal{O})$.

This observation allows us to write down the explicit unified addition formulae on (3.21) as follows: Let $P = (X_1 : Y_1 : Z_1)$ and $Q = (X_2 : Y_2 : Z_2)$ be two points

on (3.21), then $P+Q=R=(X_3:Y_3:Z_3)$, where

$$\begin{cases} X_3=(X_1Z_2+X_2Z_1)(Y_1Y_2+Z_1Z_2)(Y_1Z_2+Y_2Z_1) \\ Y_3=(X_1X_2-Z_1Z_2)(Z_1^2Z_2^2-Y_1^2Y_2^2) \\ Z_3=(Y_1Z_2+Y_2Z_1)(X_1X_2+Z_1Z_2)(Y_1Y_2-Z_1Z_2) \end{cases}$$

These formulae are obtained by choosing $\mathcal{O}^{'}=(0:1:0)$ as the neutral element results in translating the group law, in other words, the point addition $P+Q$ transforms to $P+Q+\mathcal{O}^{'}$. The inverse of the point $(X_1:Y_1:Z_1)$ on (3.21) is $(X_1:Y_1:-Z_1)$, which is unchanged. Note also that the above formulae are complete provided that $X_1X_2\neq Z_1Z_2$ and $Y_1Y_2\neq Z_1Z_2$, and are independent of curve parameters $a,b\in\mathbb{F}$. The computational cost of point addition, point doubling, and unified addition are $12\mathbf{M}$, $6\mathbf{M}+5\mathbf{S}$, and $11\mathbf{M}$, respectively. For a mixed point addition (i.e., when $Z_2=1$), the number of required multiplications drops to $10\mathbf{M}$.

Joye et al. [40] investigated twisted Huff curves defined by

$$ax(y^2-d)=by(x^2-d), \tag{3.22}$$

where $abd(a^2-b^2)\neq 0$. These curves are also defined in projective coordinates as follows:

$$aX(Y^2-dZ^2)=bY(X^2-dZ^2), \tag{3.23}$$

where $abd(a^2-b^2)\neq 0$. They obtained point addition formulae with performing $12\mathbf{M}$. For more formulae, one can look at [40].

In order to improve the number of isomorphism classes, a generalized Huff curves was introduced by Wu and Feng in [77]. It is note worthy that the Huff curve family is included in the generalized Huff curves. These curves over the finite field $\mathbb{F}$ with $char(\mathbb{F})\neq 2$ are defined by

$$x(ay^2-1)=y(bx^2-1), \tag{3.24}$$

where $ab(a-b)\neq 0$. They obtained the following formulae for addition and doubling on generalized Huff curves in affine coordinates as follows: Let $P=(x_1,y_1)$ and $Q=(x_2,y_2)$ be two points on (3.24), then $P+Q=R=(x_3,y_3)$, where

$$\begin{cases} x_3=\dfrac{(x_1+x_2)(ay_1y_2+1)}{(bx_1x_2+1)(ay_1y_2-1)} \\ y_3=\dfrac{(y_1+y_2)(bx_1x_2+1)}{(bx_1x_2-1)(ay_1y_2+1)} \end{cases}$$

and $2P=R=(x_3,y_3)$, where

$$\begin{cases} x_3=\dfrac{2x_1(ay_1^2+1)}{(bx_1^2+1)(ay_1^2-1)} \\ y_3=\dfrac{2y_1(bx_1^2+1)}{(bx_1^2-1)(ay_1^2+1)} \end{cases}$$

In order to avoid inversion for addition formulae in affine coordinates, the generalized Huff curves in projective coordinates are defined by

$$X(aY^2 - Z^2) = Y(bX^2 - Z^2), \tag{3.25}$$

with the map $(x, y) \mapsto (X : Y : Z)$ for $Z \neq 0$. In this case, there are three infinite points, namely $(1 : 0 : 0), (0 : 1 : 0)$ and $(a : b : 0)$. We will now discuss addition of any two points by selecting $(1 : 0 : 0)$ as identity element. The negative of $(X_1 : Y_1 : Z_1)$ on (3.25) is $(X_1 : Y_1 : -Z_1)$. Let $P = (X_1 : Y_1 : Z_1)$ and $Q = (X_2 : Y_2 : Z_2)$ be two points on (3.25). Then $P + Q = R = (X_3 : Y_3 : Z_3)$, where

$$\begin{cases} X_3 = (bX_1X_2 - Z_1Z_2)(bX_1X_2 + Z_1Z_2) \\ \qquad (Z_1Z_2 - aY_1Y_2) \\ Y_3 = b(X_1Z_2 + X_2Z_1)(bX_1X_2 + Z_1Z_2) \\ \qquad (Y_1Z_2 + Y_2Z_1) \\ Z_3 = b(X_1Z_2 + X_2Z_1)(bX_1X_2 - Z_1Z_2) \\ \qquad (aY_1Y_2 + Z_1Z_2) \end{cases}$$

These formulae are unified. The computational cost of point addition and doubling corresponding the identity element $(1 : 0 : 0)$ are $11\mathbf{M} + 3\mathbf{D}$ and $6\mathbf{M} + 5\mathbf{S} + 3\mathbf{D}$, respectively. For a mixed point addition (i.e., when $Z_2 = 1$), the number of required multiplications drops to $10\mathbf{M} + 3\mathbf{D}$. It is possible to choose $(0 : 1 : 0)$ and $(a : b : 0)$ as identity elements. In each case, the negative of $(X_1 : Y_1 : Z_1)$ on (3.25) is $(X_1 : Y_1 : -Z_1)$. In order to examine the related point addition formulae and more, we refer the reader to [77].

In 2011, Ciss and Sow [20] introduced the new generalized Huff curves over the finite field $\mathbb{F}$ of $\mathrm{char}(\mathbb{F}) \neq 2$. These curves are defined by

$$ax(y^2 - c) = by(x^2 - d), \tag{3.26}$$

where $a, b, c, d \in \mathbb{F}$ with $abcd(a^2c - b^2d) \neq 0$. The new generalized Huff curves contains the generalized Huff's model $ax(y^2 - d) = by(x^2 - d)$ with $abd(a^2 - b^2) \neq 0$ of Joye et al. [40] and the generalized Huff curves $x(ay^2 - 1) = y(bx^2 - 1)$ with $ab(a - b) \neq 0$ of Wu and Feng [77] as a special case. Ciss and Sow obtained the addition and doubling formulae of new generalized Huff curves in affine coordinates as follows: Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on (3.26). Then $P + Q = R = (x_3, y_3)$, where

$$\begin{cases} x_3 = \dfrac{d(x_1 + x_2)(c + y_1y_2)}{(d + x_1x_2)(c - y_1y_2)} \\ y_3 = \dfrac{c(y_1 + y_2)(d + x_1x_2)}{(d - x_1x_2)(c + y_1y_2)} \end{cases}$$

and $2P = R = (x_3, y_3)$, where

$$\begin{cases} x_3 = \dfrac{2dx_1(c + {y_1}^2)}{(d + {x_1}^2)(c - {y_1}^2)} \\ y_3 = \dfrac{2cy_1(d + {x_1}^2)}{(d - {x_1}^2)(c + {y_1}^2)} \end{cases}$$

The addition formulae are complete if $x_1x_2 \neq \pm c$ and $y_1y_2 \neq \pm d$ and the doubling formulae are complete if ${x_1}^2 \neq \pm c$ and ${y_1}^2 \neq \pm d$ and in particular if c and d are not square in $\mathbb{F}$.

In order to avoid inversion in addition formulae, the new generalized Huff curves in projective coordinates are defined by

$$aX(Y^2 - cZ^2) = bY(X^2 - dZ^2), \tag{3.27}$$

where $a, b, c, d \in \mathbb{F}$ with $abcd(a^2c - b^2d) \neq 0$. The neutral element of the group law is $\mathcal{O} = (0 : 0 : 1)$ and the negative of $(X_1 : Y_1 : Z_1)$ on (3.27) is $(X_1 : Y_1 : -Z_1)$. The addition law in projective coordinates is as fast as in the previous particular cases. They obtained point addition and point doubling on (3.27) by the following formulae: Let $P = (X_1 : Y_1 : Z_1)$ and $Q = (X_2 : Y_2 : Z_2)$ be two points on (3.27), then $P + Q = R = (X_3 : Y_3 : Z_3)$, where

$$\begin{cases} X_3 = d(X_1Z_2 + X_2Z_1)(cZ_1Z_2 + Y_1Y_2)^2 \\ \qquad (dZ_1Z_2 - X_1X_2) \\ Y_3 = c(Y_1Z_2 + Y_2Z_1)(dZ_1Z_2 + X_1X_2) \\ \qquad (cZ_1Z_2 - Y_1Y_2) \\ Z_3 = (d^2Z_1^2Z_2^2 - X_1^2X_2^2)(c^2Z_1^2Z_2^2 - Y_1^2Y_2^2) \end{cases}$$

and $2P = R = (X_3 : Y_3 : Z_3)$, where

$$\begin{cases} X_3 = 2dX_1(cZ_1^2 + Y_1^2)^2(dZ_1^2 - X_1^2) \\ Y_3 = 2cY_1(dZ_1^2 + X_1^2)^2(cZ_1^2 - Y_1^2) \\ Z_3 = (d^2Z_1^4 - X_1^4)(c^2Z_1^4 - Y_1^4) \end{cases}$$

The above point addition formulae are complete provided that $X_1X_2 \neq dZ_1Z_2$ and $Y_1Y_2 \neq cZ_1Z_2$. The computational cost of point addition and point doubling formulae are $12\mathbf{M} + 4\mathbf{D}$ and $7\mathbf{M} + 5\mathbf{S} + 4\mathbf{D}$, respectively. The total cost of point addition reduces to $11\mathbf{M} + 4\mathbf{D}$ by replacing $Z_2 = 1$. For further details, we refer the reader to look at [20].

3.6.2 Binary Huff Curves

Joye et al. [40] introduced the binary Huff curves over $\mathbb{F}$ with $\mathrm{char}(\mathbb{F}) = 2$ which are defined by

$$ax(y^2 + y + 1) = by(x^2 + x + 1), \tag{3.28}$$

or in projective coordinates

$$aX(Y^2 + YZ + Z^2) = bY(X^2 + XZ + Z^2), \tag{3.29}$$

where $ab(a-b) \neq 0$. Devigne and Joye [21] described the addition law for binary Huff curves. They showed that there are three points at infinity satisfying the curve equation, namely $(a : b : 0)$, $(1 : 0 : 0)$, and $(0 : 1 : 0)$. They obtained unified point addition formulae as follows: Let $P = (X_1 : Y_1 : Z_1)$ and $Q = (X_2 : Y_2 : Z_2)$ be two points on (3.29), then $P + Q = R = (X_3 : Y_3 : Z_3)$ with

$$\begin{cases} X_3 = (Z_1Z_2 + Y_1Y_2)[(X_1Z_2 + X_2Z_1)(Z_1^2Z_2^2 \\ \qquad +X_1X_2Y_1Y_2) + \alpha X_1X_2Z_1Z_2(Z_1Z_2 + Y_1Y_2)] \\ Y_3 = (Z_1Z_2 + X_1X_2)[(Y_1Z_2 + Y_2Z_1)(Z_1^2Z_2^2 \\ \qquad +X_1X_2Y_1Y_2) + \beta Y_1Y_2Z_1Z_2(Z_1Z_2 + X_1X_2)] \\ Z_3 = (Z_1Z_2 + X_1X_2)(Z_1Z_2 + Y_1Y_2) \\ \qquad (Z_1^2Z_2^2 + X_1X_2Y_1Y_2) \end{cases}$$

where $\alpha = (a+b)/b$ and $\beta = (a+b)/a$. The computational cost of unified point addition is $15\mathbf{M} + 2\mathbf{D}$.

The generalized binary Huff curves over $\mathbb{F}$ with $\mathrm{char}(\mathbb{F}) = 2$ are also defined by Devigne and Joye [21] which are of the form

$$ax(y^2 + fy + 1) = by(x^2 + fx + 1), \tag{3.30}$$

or in projective coordinates

$$aX(Y^2 + fYZ + Z^2) = bY(X^2 + fXZ + Z^2), \tag{3.31}$$

where $abf(a-b) \neq 0$. They obtained the unified addition formulae of generalized binary Huff curves in affine coordinates which are given by the following formulae: Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on (3.30). Then $P + Q = R = (x_3, y_3)$, where

$$\begin{cases} x_3 = \dfrac{b(x_1 + x_2)(1 + x_1x_2y_1y_2)}{b(1 + x_1x_2)(1 + x_1x_2y_1y_2)} \\ \qquad + \dfrac{f(a+b)x_1x_2(1 + y_1y_2)}{b(1 + x_1x_2)(1 + x_1x_2y_1y_2)} \\ y_3 = \dfrac{a(y_1 + y_2)(1 + x_1x_2y_1y_2)}{a(1 + y_1y_2)(1 + x_1x_2y_1y_2)} \\ \qquad + \dfrac{f(a+b)y_1y_2(1 + x_1x_2)}{a(1 + y_1y_2)(1 + x_1x_2y_1y_2)} \end{cases}$$

The computational cost of point addition, point doubling and unified addition formulae are $15\mathbf{M}$, $6\mathbf{M} + 2\mathbf{D}$ and $15\mathbf{M} + 2\mathbf{D}$, respectively. For more information and formulae, we refer the reader to [21].

3.7 Comparison and Conclusions

In this chapter, the alternative models of elliptic curves are surveyed by pinning down group operations, and performance in various coordinate systems. Table 1 summarizes the costs of addition, doubling, mixed addition and unified addition on alternate models of elliptic curves. The comparison in affine coordinates is skipped because the cost of field inversion is so expensive. We enumerate the cost of field operations in terms of multiplication **M**, squaring **S**, and multiplication by a constant **D** in $\mathbb{F}$.

Table 3.1: Cost of Arithmetic on Alternate Models of Elliptic Curves

EC Model	**Coordinates**	**Add**	**Doubling**	**Mixed Add**	**Unified Add**
Weierstrass	Projective	12M + 2S	5M+6S	9M+2S	11M + 5S + 1D
Edwards	Projective	10M+1S+1D	3M+4S	9M+1S+1D	10M+1S+1D
Twisted Edwards	Projective	11M+2D	3M+4S+1D	9M+2D	10M+1S+2D
	Inverted	11M+2D	3M+4S+2D	9M+2D	9M+1S+2D
	Extended	9M+1D	4M+4S+1D	8M+1D	9M+2D
Jacobi Intersections	Projective	13M+2S+1D	4M+3S	11M+2S+1D	13M+2S+1D
Twisted Jacobi Intersections	Projective	12M	3M+4S+1D	10M	13M+2S+5D
Extended Jacobi Quartics	Jacobian	10M+3S+1D	1M+9S+1D	8M+3S+1D	10M+3S+1D
	Extended Proj.	7M+3S+2D	8S	6M+3S+2D	8M+3S+2D
Hessian Curves	Projective	12M	12M	10M	12M
Generalized Hessian Curves	Projective	12M+1D	6M+3S+1D	10M+1D	12M+1D
Huff Curves	Projective	12M	6M+5S	10M	11M
Generalized Huff Curves	Projective	11M+3D	6M+5S+3D	10M+3D	11M+3D
New Generalized Huff Curves	Projective	12M+4D	7M+5S+4D	11M+4D	Open Problem

where Add means addition. The unified addition formulae offer inherited countermeasure against SPA with comparable performance. In these models, SPA is avoided by employing the unified addition formulae or an algorithmic adaptation of it that behaves in similar fashion during the process of point addition and point doubling. Hence, for algorithmic flexibility, alternate models of elliptic curves with desirable properties are put together in this chapter for elliptic curve cryptographic protocols.

CHAPTER 4

MESSAGE TRANSMISSION AND NYBERG RUEPPEL TYPE DIGITAL SIGNATURE FOR GH-PKC

4.1 Introduction

In this chapter we propose ElGamal type encryption scheme based on the concepts of public key cryptosystem over cubic finite field extension proposed by Gong and Harn (GH). The proposed GH-ElGamal type encryption scheme is semantically secure and the semantic security is computationally equivalent to splitting $z \in \mathbb{F}_q$ into two numbers $(X, Y) \in \mathbb{F}_q^2$ such that $z = X + Y$ where $s_e \mapsto X$ and $s_{-e} \mapsto Y$. Moreover, $e \in \mathbb{Z}_\ell$ and $\mathbb{Z}_e ll$ is chosen large enough so that the brute force attack becomes infeasible. The number of such (X, Y) are equal to $\frac{q-1}{2}$. However the security of Deffie-Hellman type key exchange depends upon the difficulty of solving 3-LFSR-DLP, 3-LFSR-DHP and 3-LFSR-DDHP. The proposed encryption scheme is *ephemeral-static*, which is useful in situations like email where the recipient may not be on line. We adapt efficient double exponentiation algorithm for GH construction that leads us to propose GH-Nyberg-Rueppel-type digital signature algorithm (GH-NR-DSA) with message recovery based on the proposed scheme. We also give some countermeasures for GH-NR-DSA to resist two well known forgery attacks. Moreover, since digital signatures with message recovery is useful for the applications in which confidentiality, integrity, and non-repudiation is required.

The PKC based on intractability of the discrete logarithm problem (DLP) was first realized by Diffie and Hellman in 1976 [22]. Robust security, efficient group operations, and economical transmission are desirable properties for PKCs. Over the last few years, efforts were made to improve efficiency of PKCs in terms of compact representation and increased security such as LUC-PKC [71, 72], GH-PKC [30, 31, 32], XTR-PKC [47, 48, 16, 73, 68], and Torus-Based-PKC [64, 65]. In these systems, the group operations are performed in intermediate or prime subfields, whereas security of extension field is ensured. This gives rise to attain higher security with compact representation of operands. It is pertinent to note that for seminal Diffie-Hellman-PKC, both group operations and DLP are in prime field. Before discussing mathematical details first we define few terminologies which are used in the thesis.

The ***static public key*** is the one which is long lived and authenticated by certification authority where as the ***ephemeral public key*** is the one which is short lived and unauthenticated. The PKC could use any of the following combination of static, ephemeral public keys.

i) **Case I**: static-static public key, in which both sender and receiver use static public keys. In this case the shared key is computed based upon life time fixed keys.

ii) **Case II**: ephemeral-ephemeral public key, in which both sender and receiver use ephemeral keys. This case achieves perfect forward security but public keys are unauthenticated. Moreover, it can also mitigate DoS attacks to some extent as both sender and receiver have to compute expensive exponentiation. Moreover, both sender and receivers are to be on line to establish communication.

iii) **Case III**: ephemeral-static public key, in which sender uses ephemeral and receiver uses static key. In this case both sender and receivers are not required to be on line to establish communication.

The basic reason for reusing ephemeral key is to improve efficiency by reducing expensive exponentiations. On the other hand, to achieve better forward security the reuse of ephemeral key is to be avoided. In the proposed GH-ElGamal type encryption the security of key exchange depends upon the difficulty of solving 3-LFSR-DLP, 3-LFSR-DHP and 3-LFSR-DDHP, where as semantic security depends upon splitting $z \in \mathbb{F}_q$ into two numbers $(X, Y) \in \mathbb{F}_q^2$ such that $z = X + Y$. The number of such (X, Y) are equal to $\frac{q-1}{2}$.

Gong and Harn proposed GH-PKC [30] based on cubic finite field extension in 1998. In GH-PKC, the group operations are carried out in prime field $\mathbb{F}_p$, whereas DLP security lies in cubic extension field $\mathbb{F}_{p^3}$. They used elements of order dividing $p^2 + p + 1$ which does not lie in any proper subfields of $\mathbb{F}_{p^3}$ to construct DLP in $\mathbb{F}_{p^3}$ and employed third-order recurrence relation to compute n-th term in output sequence of a Linear Feedback Shift Register (LFSR). The output sequence of LFSR is computed over $\mathbb{F}_p$ where elements are represented by $\log p$ bits. Therefore, this system achieves a compressed representation by a factor of 2/3 to reduce bandwidth requirements.

In [30], Gong and Harn proposed the Diffie-Hellman public key distribution (GH-PKD) scheme and the RSA-type (GH-RSA-type) encryption scheme as an example of the applications of GH-PKC. Later, Gong, Harn and Wu obtained the construction of the GH-ElGamal-type digital signature algorithm (GH-DSA) in [31]. In GH-RSA-type encryption scheme, although the computation of n-th term is more feasible, there are more number of field multiplications, pre-computational overheads, and storage memory. Therefore an improvement in encryption protocol is proposed to make GH-PKD scheme more practical.

In [59], Nyberg and Rueppel proposed the first ElGamal signature scheme based

on DLP that gives key exchange, message recovery and digital signature algorithm in a single protocol. This was followed by several other proposals [60, 55, 1, 78].

This chapter is organized as follows. In section 4.2, we will review the necessary mathematical background to construct GH-PKC, and describe briefly GH-PKC protocols including GH-PKD scheme and GH-RSA-type encryption scheme. In section 4.3 we will discuss exponentiation algorithm for GH-PKC. In section 4.4 we will propose a novel *ephemeral-static* encryption scheme based on the concepts of GH-PKD scheme and then based on the proposed encryption scheme, the GH-Nyberg-Rueppel-type digital signature algorithm (GH-NR-DSA) is also proposed in section 4.5. We conclude the chapter in section 4.6.

4.2 GH-Public Key Cryptosystem

4.2.1 Preliminaries

Let q be a prime power and $f(x) = x^3 - ax^2 + bx - 1$ be a polynomial over $\mathbb{F}_q$. A third-order LFSR sequence $\underline{s} = \{s_n\} = \{s_n(a,b)\}$ over $\mathbb{F}_q$ is called the characteristic sequence generated by $f(x)$ if the elements of $\underline{s}$ satisfy

$$s_n = as_{n-1} - bs_{n-2} + s_{n-3}, \quad n \geq 3 \tag{4.1}$$

with the initial conditions $s_0 = 3, s_1 = a$, and $s_2 = a^2 - 2b$.

We assume that f is irreducible over $\mathbb{F}_q$ and α is a root of $f(x)$, then α, α^q and α^{q^2} are all three roots of $f(x)$ in $\mathbb{F}_{q^3}$. Therefore, by Newton's formula, $\underline{s}$ can be represented by $s_n = Tr(\alpha^n) = \alpha^n + \alpha^{nq} + \alpha^{nq^2}$, $n \in \mathbb{Z}$. It follows from this fact that n-th powers of the roots of $f(x)$ are the roots of the polynomial $f_n(x)$, i.e.,

$$\begin{aligned} f_n(x) &= x^3 - s_n(a,b)x^2 + s_{-n}(a,b)x - 1 \\ &= (x - \alpha^n)(x - \alpha^{nq})(x - \alpha^{nq^2}). \end{aligned} \tag{4.2}$$

Since $f(x)$ is irreducible, per(f), so-called the period of $f(x)$ in $\mathbb{F}_{q^3}$, is equal to the per$(\underline{s})$, which is a factor of $\ell = q^2 + q + 1$. If gcd(per$(f), n) = 1$, $f(x)$ and $f_n(x)$ have the same period, so $f_n(x)$ is irreducible over $\mathbb{F}_q$.

We now give the following facts that play important roles to construct GH-PKD scheme and our proposed encryption scheme. The details and the proofs can be found in [30].

Lemma 4.1. *Let $f(x) = x^3 - ax^2 + bx - 1$ be a polynomial over $\mathbb{F}_q$, and let $\underline{s}$ be its characteristic sequence. Then for all $e, r \in \mathbb{Z}^+$,*

$$(s_{er}, s_{-er}) = (s_e(s_r, s_{-r}), s_{-e}(s_r, s_{-r})) = (s_r(s_e, s_{-e}), s_{-r}(s_e, s_{-e})).$$

Remark 4.1. It follows from Lemma 4.1 that (s_{er}, s_{-er}) is computed when either $e \in \mathbb{Z}^+$ and (s_r, s_{-r}) or $r \in \mathbb{Z}^+$ and (s_e, s_{-e}) are known.

Lemma 4.2. *Let $f(x) = x^3 - ax^2 + bx - 1$ be an irreducible polynomial over $\mathbb{F}_q$ with per(f) dividing $\ell = q^2 + q + 1$, and $\underline{s}$ be its characteristic sequence. Let $\mathbb{Z}_\ell = \{0, 1, 2, \cdots, \ell - 1\}$ and $\mathbb{Z}_\ell^\times = \{x \in \mathbb{Z}_\ell \mid \gcd(x, \ell) = 1\}$. Let R_ℓ be the set containing all elements in $\mathbb{Z}_\ell^\times$ which are not conjugate each other modulo ℓ with respect to q, i.e., $R_\ell = \{e \in \mathbb{Z}_\ell^\times | \ e \not\equiv rq^i \mod \ell, \ r \in \mathbb{Z}, \ 0 \le i \le 2\}$. Then the following mathematical function ψ*

$$\begin{aligned} \psi \ : \ R_\ell &\longrightarrow \mathbb{F}_q \times \mathbb{F}_q \\ e &\longmapsto (s_e, s_{-e}) \end{aligned}$$

is a bijective map from R_ℓ to $\mathbb{F}_q \times \mathbb{F}_q$.

Remark 4.2. It follows from Lemma 4.2 that there will be different public keys corresponding to different private keys in GH-PKD scheme. The number of the space of private or public keys is $\phi(q^2 + q + 1)/3$, where ϕ is Euler function.

Lemma 4.3. *Let $\underline{s}$ be the characteristic sequence over $\mathbb{F}_q$ with the characteristic polynomial $f(x) = x^3 - ax^2 + bx - 1$. Then for any $u, v \in \mathbb{Z}^+$*

i) $s_{2u} = s_u^2 - 2s_{-u}$

ii) $s_u s_v - s_{u-v} s_{-v} = s_{u+v} - s_{u-2v}$.

Gong and Harn [30] showed that the elements of order dividing $\ell = q^2 + q + 1$ in $\mathbb{F}_{q^3}^*$ could be identified by $\{s_n, s_{-n}\}$ with a compression factor 2/3. They also obtained an efficient exponentiation algorithm to calculate a pair of the n-th terms s_n and s_{-n} by using Lemma 4.3, which needs $9 \log n$ modulo q multiplications on average. This algorithm is more efficient than Fiduccia's one that uses modulo polynomial in [26]. Gong, Harn and Wu [31, Algorithm 1] also proposed much more efficient algorithm utilizing the signed-digit representation that resists against Simple Power Analysis (SPA) attacks. Using this representation to calculate a pair of the n-th terms s_n and s_{-n} needs $4 \log n$ multiplications and $4 \log n$ squarings in $\mathbb{F}_q$ on average. This method is optimized where q is a prime or a prime power in [32].

Remark 4.3 (Parameter selection). In [29, 47] an efficient method to select a good prime p to construct the finite field $\mathbb{F}_q$ of characteristic p is given. Also it is described how to select a subgroup of order dividing $q^2 + q + 1$.

In [29], Giuliani and Gong define some complexity problems related to the kth-order LFSR. Inspired by their definitions, we give some analogous problems for third-order LFSR that the security of our proposed scheme will depend on these problems, later.

Definition 4.1. Given (s_1, s_{-1}) and (s_e, s_{-e}), the problem of finding e with $1 \le e \le q^2 + q + 1$ is called the third-order LFSR-Based Discrete Logarithm Problem (3-LFSR-DLP).

Definition 4.2. Given (s_1, s_{-1}), (s_e, s_{-e}) and (s_r, s_{-r}), the problem of determining (s_{er}, s_{-er}) is called the third-order LFSR-Based Diffie Hellman Problem (3-LFSR-DHP).

Definition 4.3. Given (s_1, s_{-1}), (s_e, s_{-e}), (s_r, s_{-r}), (s_{er}, s_{-er}) and (s_c, s_{-c}), where c is randomly chosen, the problem of determining the solution of 3-LFSR-DHP whether (s_{er}, s_{-er}) or (s_c, s_{-c}) is called the third-order LFSR-Based Decisional Diffie Hellman Problem (3-LFSR-DDHP).

Remark 4.4. It is essentially proven that kth-order LFSR-DLP is computationally equivalent to DLP over F_{q^k} in [74]. Giuliani and Gong prove the analogous for the Diffie Hellman and Decisional Diffie-Hellman problems in [29].

Semantic Security: Suppose that $m_1 \in \mathbb{F}_q$ and $(m_2) \in \mathbb{F}_q$ are two known messages from the attacker $\mathcal{E}$ and $\mathcal{A}$ encrypts message m_1 such that $c = m_1K = m_1(s_{tr} + s_{-tr})$ and sends the ciphertext c to attacker $\mathcal{E}$ who wants to determine whether c is encryption of m_1 or,m_2. To do this attacker E divides c with m_1 such that $\acute{K} = c/m_1$. Now if attacker $\mathcal{E}$ can find out either s_{tr} or s_{-tr} from $\acute{K}$ then he can form cubic equation $\bar{g}(x) = x^3 - s_{tr}x^2 + s_{-tr}x - 1$ over $\mathbb{F}_q$. Then he determines the irreducibility of $\bar{g}(x)$, if it is irreducible the attacker $\mathcal{E}$ concludes that c is encryption of m_1. But to determine s_{tr} or s_{-tr} from $\acute{K}$ is computationally equivalent to splitting $z \in \mathbb{F}_q$ into two numbers $(X, Y) \in \mathbb{F}_q^2$ such that $z = X + Y$ where $s_{tr} \mapsto X$ and $s_{-tr} \mapsto Y$. The number of such (X, Y) are equal to $\frac{q-1}{2}$. Moreover, the product $tr \in \mathbb{Z}_\ell$ and $\mathbb{Z}_ell$ is chosen large enough so that the brute force attack becomes infeasible.

Remark 4.5 (Semantic Security). Given $G =< \alpha >\subset \mathbb{F}_{q^3}$, (s_1, s_{-1}) and $z \in \mathbb{F}_q$. Finding (s_e, s_{-e}) such that $z = s_e + s_{-e}$ is computationally equivalent to computing $z = X + Y$, $(X, Y) \in \mathbb{F}_q^2$. Since there is no polynomial time algorithm to split z as a sum of two elements in $\mathbb{F}_q$, where the total number of unknown such as (X, Y) are equal to $\frac{q-1}{2}$ and $s_e \mapsto X$ and $s_{-e} \mapsto Y$. Moreover, $e \in \mathbb{Z}_\ell$ and $\mathbb{Z}_ell$ is chosen large enough so that the brute force attack becomes infeasible.

4.2.2 Motivation to Use LFSR Based Construction

The main reasons for developing cryptographic protocols based on LFSR Based construction are as follows:

(i) **Security:** In GH-PKC, the LFSR based construction provide security of extension field $\mathbb{F}_{q^3}$ that is DLP lies in $\mathbb{F}_{q^3}$ with faster exponentiation and compressed data for transmission by a factor of 2/3.

(ii) The cost of exponentiation is lesser than the generic square and multiply algorithm and elliptic curve $(E(\mathbb{F}_q))$ based exponentiation. The details are given in the table below:

(iii) **Transmission Size:** Due to compression factor of 2/3 in GH-PKC the overall data transmission size is also reduced to implement PKC protocols.

Table 4.1: Exponentiation Comparison

Process	**Multiplications over $\mathbb{F}_q$**	**Squaring over $\mathbb{F}_q$**
$\alpha^n \in \mathbb{F}_{q^3}, q = p^r, r \in \mathbb{Z}$	$(7r^{\log 3} + 10.8r - 10.8)\log n$	$3.8r^{log3}logn$
$s_n, s_{-n}, \alpha^n \in \mathbb{F}_{q^3}$	$4\log n$	$4\log n$
$nP, P \in E(\mathbb{F}_q)$	$4\lceil \log n \rceil + 12(weight(n))$	$4\lceil \log n \rceil + 4(weight(n))$

Algorithm 2 Compute $s_{\pm n}$ using signed-digit representation

Require: $a, b \in \mathbb{F}_q$ and $n = \sum_{j=0}^{t} n_j 2^{t-j}, n_j \in \{-1, 0, 1\}$ with $T_0 = n_0 = 1$, $T_i = n_i + 2T_{i-1}$ for $1 \le i \le t$, and so $T_t = n$.
Ensure: (s_{n-1}, s_n, s_{n+1})

$(s_{T_i-1}, s_{T_i}, s_{T_i+1}) \leftarrow (3, a, a^2 - 2b)$
$(s_{-T_i-1}, s_{-T_i}, s_{-T_i+1}) \leftarrow (b^2 - 2a, b, 3)$
if $n_t = 0$ **then**
    find $h < t$ such that $n_h \neq 0$ and $n_{h+1} = n_{h+2} = \cdots = n_t = 0$
else
    $h = t$
end if
if $h > 1$ **then**
    for $j \leftarrow 1$ ***to*** $h - 1$ **do**
        if $n_j = 1$ **then**
            $s_{T_i-1} \leftarrow s^2_{T_{i-1}} - 2s_{-T_{i-1}}$
            $s_{T_i} \leftarrow s_{T_{i-1}} s_{T_{i-1}+1} - a s_{-T_{i-1}} + s_{-(T_{i-1}-1)}$
            $s_{T_i+1} \leftarrow s^2_{T_{i-1}+1} - 2s_{-(T_{i-1}+1)}$
            $s_{-T_i-1} \leftarrow s^2_{T_{i-1}-1} - 2s_{-(T_{i-1}-1)}$
            $s_{-T_i} \leftarrow s_{T_{i-1}} s_{T_{i-1}-1} - b s_{-T_{i-1}} + s_{-(T_{i-1}+1)}$
            $s_{-T_i+1} \leftarrow s^2_{T_{i-1}} - 2s_{-T_{i-1}}$
        else
            $s_{T_i-1} \leftarrow s^2_{T_{i-1}-1} - 2s_{-(T_{i-1}-1)}$
            $s_{T_i} \leftarrow s_{T_{i-1}} s_{T_{i-1}-1} - b s_{-T_{i-1}} + s_{-(T_{i-1}+1)}$
            $s_{T_i+1} \leftarrow s^2_{T_{i-1}} - 2s_{-T_{i-1}}$
            $s_{-T_i-1} \leftarrow s^2_{T_{i-1}} - 2s_{-T_{i-1}}$
            $s_{-T_i} \leftarrow s_{T_{i-1}} s_{T_{i-1}+1} - a s_{-T_{i-1}} + s_{-(T_{i-1}-1)}$
            $s_{-T_i+1} \leftarrow s^2_{T_{i-1}+1} - 2s_{-(T_{i-1}+1)}$
        end if
    end for
end if
for $j = \max\{1, h\}$ ***to*** t **do**
    $s_{\pm T_i} \leftarrow s^2_{T_{i-1}} - 2s_{-T_{i-1}}$
end for
return $(s_{T_i-1}, s_{T_i}, s_{T_i+1})$ and $(s_{-T_i-1}, s_{-T_i}, s_{-T_i+1})$

Now briefly discuss GH-PKD scheme and GH-RSA-type encryption scheme. For more details, one can refer to [30] and [31].

4.2.3 GH-PKD Scheme

- *Public Parameters:* q is a prime power, and $f(x) = x^3 - ax^2 + bx - 1$ is an irreducible polynomial over $\mathbb{F}_q$ with $\text{per}(f) = \ell$ dividing $q^2 + q + 1$.
- $\mathcal{A}$ randomly selects e satisfying $0 < e < \ell$ such that $\gcd(e, \ell) = 1$ as her static private key. She then computes her static public key (s_e, s_{-e}) using public parameters, and sends (s_e, s_{-e}) to $\mathcal{B}$.
- $\mathcal{B}$ randomly selects r satisfying $0 < r < \ell$ such that $\gcd(r, \ell) = 1$ as his static private key. He then computes his static public key (s_r, s_{-r}) using public parameters, and sends (s_r, s_{-r}) to $\mathcal{A}$.
- $\mathcal{A}$ computes $(s_{er}, s_{-er}) = (s_e(s_r, s_{-r}), s_{-e}(s_r, s_{-r}))$.
- $\mathcal{B}$ computes $(s_{er}, s_{-er}) = (s_r(s_e, s_{-e}), s_{-r}(s_e, s_{-e}))$.
- Both $\mathcal{A}$ and $\mathcal{B}$ agree on common key (s_{er}, s_{-er}).

Security: The security of GH-PKD scheme depends on the difficulty of solving 3-LFSR-DLP and 3-LFSR-DHP.

Cost of Protocol: In GH-PKD protocol, both parties can compute common key in 8 trace based exponentiations

4.2.4 GH-RSA-Type Encryption Scheme

The RSA-type encryption scheme is based on third order characteristic sequence over $\mathbb{Z}_N$ which is an integer ring modulo N. Let $\mathcal{A}$ and $\mathcal{B}$ be two parties. $\mathcal{A}$ wants to send a message $M = (m_1, m_2)$ to $\mathcal{B}$, then both proceed as follows:

- *Public Parameters:* Let e and N, such that $N = pq$, p and q are primes, and $\gcd(e, p^\ell - 1) = 1$ for $\ell = 2, 3$.
- *Private Keys:* Let $d_0, d_1, \ldots, d_8$ be decryption keys. Each d_j with $0 \leq j \leq 8$ are computed and selected as a function of ciphertext $C = (c_1, c_2)$ such that $d_j e \equiv 1 \pmod{\delta_j}$ for $0 \leq j \leq 8$. Each δ_j with $0 \leq j \leq 8$ are also precomputed and selected as a function of C .
- *Encryption:* For a message $M = (m_1, m_2)$, where $0 < m_1, m_2 < N$, $\mathcal{A}$ computes ciphertext $C = (c_1, c_2)$ with $c_1 = s_e(m_1, m_2)$ and $c_2 = s_{-e}(m_1, m_2)$.
- *Decryption:* $\mathcal{B}$ selects a proper decryption key in the set $\{d_0, d_1, \ldots, d_8\}$, then he computes message $m_1 = s_{d_j}(c_1, c_2)$ and $m_2 = s_{-d_j}(c_1, c_2)$ using the selected key d_j for $0 \leq j \leq 8$.

Security Analysis: The security of GH-RSA-Type Encryption Scheme is based on the difficulty of factoring a large composite integer.

4.3 Improved Exponentiation Algorithms

Gong and Harn [30] showed that the elements of order ℓ dividing q^2+q+1 in $\mathbb{F}_{q^3}^*$ could be identified by $\{s_n, s_{-n}\}$ with a compression factor 2/3. They also obtained an efficient single exponentiation algorithm to calculate a pair of the n-th terms s_n and s_{-n} by using Lemma 4.3, which needs $9\log n$ modulo q multiplications on average. This algorithm is more efficient than Fiduccia's one that uses modulo polynomial in [26]. Gong, Harn and Wu [30, Algorithm 1] also proposed much more efficient algorithm utilizing the signed-digit representation that resists against Simple Power Analysis (SPA) attacks. Using this representation to calculate a pair of the n-th terms s_n and s_{-n} needs $4\log n$ multiplications and $4\log n$ squarings in $\mathbb{F}_q$ on average.

A double exponentiation algorithm to calculate s_{ak+bl} for XTR proposed by Lenstra with using matrices in [47]. Later, Stam and Lenstra [73] proposed more efficient double exponentiation algorithm for XTR without using matrices. Their algorithm is an adaptation of Montgomery's method [57] that computes Lucas construction. In this section, we propose efficient double exponentiation algorithm to calculate $s_{\pm(ak+bl)}$ for GH construction adapting the techniques used in [73, 57], while it is possible to use matrix methods. We also show how to use double exponentiation algorithm to speed up single exponentiation algorithm and compare it with the similar ones.

4.3.1 Improved GH Double Exponentiation

Let $a, b > 0$, $s_{\pm k}, s_{\pm l}, s_{\pm(k-l)}$ and $s_{\pm(k-2l)}$ be given. Then the double exponentiation $s_{\pm(ak+bl)}$ is efficiently computed in Algorithm 3. The outline of Algorithm 3 is as follows: Let $u=k, v=l, d=a$ and $e=b$. Then it follows from this fact that $du+ev=ak+bl$ and $s_{\pm u}$, $s_{\pm v}$, $s_{\pm(u-v)}$ and $s_{\pm(u-2v)}$ are known. Therefore, in the main part of Algorithm 3, d, e, u and v will be updated so that $du+ev=ak+bl$ holds with $d, e>0$ and $d+e$ decreases until $d=e$. Also, $s_{\pm u}$, $s_{\pm v}$, $s_{\pm(u-v)}$ and $s_{\pm(u-2v)}$ are updated according to the new values of u and v. If $d=e$, then $ak+bl=du+ev=d(u+v)$ so that $s_{\pm(ak+bl)}$ follows by computing $s_{\pm(u+v)}$ and next $s_{\pm d(u+v)}$ by using the single exponentiation algorithm. Table 4.2 gives the update rules for double exponentiation. Table 4.3 gives the cost of each update operation in $\mathbb{F}_q$. There are various ways in which d and e can be updated. The method that we mention here is an adaptation of [73, **?**] to GH construction.

Table 4.2: Computations and rules for double exponentiation

Name of Rule	Condition	d	e	$\pm u$	$\pm v$	$s_{\pm u}$	$s_{\pm v}$	$s_{\pm(u-v)}$	$s_{\pm(u-2v)}$
if $d>e$									
R_1	$d\le 4e$	e	$d-e$	$\pm(u+v)$	$\pm u$	$s_{\pm(u+v)}$	$s_{\pm u}$	$s_{\pm v}$	$s_{\pm(v-u)}$
R_2	d is even	$d/2$	e	$\pm 2u$	$\pm v$	$s_{\pm 2u}$	$s_{\pm v}$	$s_{\pm(2u-v)}$	$s_{\pm 2(u-v)}$
R_3	e is odd	$(d-e)/2$	e	$\pm 2u$	$\pm(u+v)$	$s_{\pm 2u}$	$s_{\pm(u+v)}$	$s_{\pm(u-v)}$	$s_{\mp 2v}$
R_4	e is even	$e/2$	d	$\pm 2v$	$\pm u$	$s_{\pm 2v}$	$s_{\pm u}$	$s_{\pm(2v-u)}$	$s_{\pm 2(v-u)}$
else									
R_{Subs}	$e>d$	e	d	$\pm v$	$\pm u$	$s_{\pm v}$	$s_{\pm u}$	$s_{\pm(v-u)}$	$s_{\pm(v-2u)}$

Table 4.3: Computational cost for each rules

Name of Rule	Condition	Cost
if $d > e$		
R_1	$d \leq 4e$	4 **M**
R_2	d is even	4 **M**+4 **S**
R_3	e is odd	4 **M**+4 **S**
R_4	e is even	4 **S**
else		
R_{Subs}	$e > d$	0

Algorithm 3 GH Double Exponentiation

Require: $a > 0, b > 0$, $s_{\pm k}$, $s_{\pm l}$, $s_{\pm(k-l)}$ and $s_{\pm(k-2l)}$.
Ensure: $(s_{ak+bl}, s_{-(ak+bl)})$

1: $f \leftarrow 0$, $d \leftarrow a$, $e \leftarrow b$, $u \leftarrow k$, $v \leftarrow l$
2: **while** d and e are both even **do**
3: $\quad d \leftarrow d/2$, $e \leftarrow e/2$, $f \leftarrow f + 1$
4: **end while**
5: **while** $d \neq e$ **do**
6: $\quad$ Apply the foremost applicable condition in Table 4.2
7: **end while**
8: Compute $(s_{u+v}, s_{-(u+v)})$ using Lemma 4.3(ii) with inputs $s_{\pm u}, s_{\pm v}, s_{\pm(u-v)}$ and $s_{\pm(u-2v)}$.
9: Compute $(s_{d(u+v)}, s_{-d(u+v)})$ applying [30, Algorithm 1] or alternatively Algorithm 4 described below with inputs d and $(s_{u+v}, s_{-(u+v)})$.
10: Compute $(s_{2^f d(u+v)}, s_{-2^f d(u+v)})$ using Lemma 4.3(i) with inputs 2 and $(s_{d(u+v)}, s_{-d(u+v)})$ f times.
11: **return** $(s_{2^f d(u+v)}, s_{-2^f d(u+v)})$

Remark 4.6. As given in [73], it is also possible to include the optional case, which d and e are both divisible by 3, into Algorithm 3.

Conjecture 4.4. *Given* $0 < a, b < \ell$, $s_{\pm k}$, $s_{\pm l}$, $s_{\pm(k-l)}$, *and* $s_{\pm(k-2l)}$, *according to the experimental results obtained by [73, Table 1], the computational cost of a pair of* s_{ak+bl} *and* $s_{-(ak+bl)}$ *can not exceed in about* $6\log(max(a, b))$ *multiplications in* $\mathbb{F}_q$ *using Algorithm 3. It may be possible to do further improvements by including more cases into Table 4.2.*

4.3.2 Improved GH Single Exponentiation

Let $s_{\pm 1}$ and n be given with $0 < n < \ell$. Then the single exponentiation $s_{\pm n}$ is efficiently computed in Algorithm 4 by just applying Algorithm 3 to $k = l = 1$ and any positive a, b with $a + b = n$. In order to speed up the algorithm, the best way to split up n in the sum of a and b by choosing b/a is close to the golden ratio $\frac{1+\sqrt{5}}{2}$, i.e., asymptotic ratio between two consecutive Fibonacci numbers. In

order to have $b/a \approx \frac{1+\sqrt{5}}{2}$, we can choose $b = z$, which is equal to the closest integer to $\frac{3-\sqrt{5}}{2}n$ and $a = n - b$.

Algorithm 4 GH Single Exponentiation

Require: $s_{\pm 1}$, n with $0 < n < \ell$, .
Ensure: (s_n, s_{-n})
1: $b \leftarrow z$, $a \leftarrow n - b$, $k \leftarrow 1$, $l \leftarrow 1$.
2: $s_{\pm k} \leftarrow s_{\pm 1}$,$s_{\pm l} \leftarrow s_{\pm 1}$, $s_{\pm(k-l)} \leftarrow s_0$, $s_{\pm(k-2l)} \leftarrow s_{\mp 1}$
3: Apply Algorithm 3 to $a, b, s_{\pm k}, s_{\pm l}, s_{\pm(k-l)}$ and $s_{\pm(k-2l)}$. **return** $(s_{ak+bl}, s_{-(ak+bl)})$

Conjecture 4.5. *Given an integer n with $0 < n < \ell$ and $s_{\pm 1}$, according to the analysis obtained by [73], the computational cost of a pair of s_n and s_{-n} can not exceed in about $5.2 \log n$ multiplications in $\mathbb{F}_q$, using Algorithm 4.*

Algorithm 4.3 is lesser computational costs than the generic exponentiation algorithm, elliptic curve exponentiation algorithm and the previously well known algorithm for GH construction. The detailed comparison is given in Table 1, where $w(n)$ is the Hamming weight of the binary representation of n that is equal to the number of non-zero digits.

Table 4.4: Comparison of Single Exponentiations

	Process	Computational Cost
Generic	$\alpha^n \in \mathbb{F}_{q^3}, q = p^r, r \in \mathbb{Z}^+$	$(7r^{\log 3} + 10.8r - 10.8) \log n$ **M** + $3.8r^{\log 3} \log n$ **S**
EC	$nP, P \in E(\mathbb{F}_p)$	$(4\lceil \log n \rceil + 12w(n))$ **M** + $(4\lceil \log n \rceil + 4w(n))$ **S**
GH	$(s_n, s_{-n}) \in \mathbb{F}_q^2$	$4 \log n$ **M** + $4 \log n$ **S**
XTR	$s_n \in \mathbb{F}_p$	$8 \log n$ **M**
Improved XTR	$s_n \in \mathbb{F}_p$	$5.2 \log n$ **M**
Improved GH	$(s_n, s_{-n}) \in \mathbb{F}_q^2$	$5.2 \log n$ **M**

4.4 Proposed Encryption Scheme

The proposed scheme is essentially an ephemeral-static public key scheme that is sender uses ephemeral key and computes encryption key using receiver's static key. The advantage of ephemeral-static encryption scheme is that both parties can communicate even when they are off line. Although the ephemeral-ephemeral encryption scheme provide better forward security but it requires both parties to be on line for communication. Moreover, ephemeral-ephemeral encryption scheme is slower than ephemeral-static scheme. The proposed encryption scheme is based on the concepts of GH-PKD scheme and semantic security depends upon splitting $z \in \mathbb{F}_q$ into two numbers $(X, Y) \in \mathbb{F}_q^2$ such that $z = X + Y$. The number of such (X, Y) are equal to $\frac{q-1}{2}$. It accrue the advantage of faster exponentiation algorithm [30, Algorithm 1] and 2/3 compression of operands. The Proposed

encryption scheme is faster than GH-RSA-Type encryption scheme with lesser storage memory. We also propose GH-Nyberg-Rueppel type digital signature algorithm (GH-NR-DSA) based on the proposed encryption scheme.

Remark 4.7. Let $f(x) = x^3 - ax^2 + bx - 1$ be a polynomial over $\mathbb{F}_q$, and let $\underline{s}$ be its characteristic sequence. It is well known that for some positive integer u, the s_u and s_{-u} are orthogonal in the variables (a, b).

Proposed GH-ElGamal type Encryption Scheme In this section, we will propose a novel public key encryption scheme which provides secure message transmission. It is essentially an *ephemeral-static* public key scheme where the initiator's public key is ephemeral and the responder's public key is static. This variant is especially useful for email where the recipient may not be on line and therefore do not have to use an ephemeral public key. The proposed encryption scheme is based on the concepts of GH-PKD scheme, which depends on the third-order LFSR sequence proposed by Gong and Harn [30]. The security of proposed encryption scheme depends on the difficulty of solving 3-LFSR-DLP, 3-LFSR-DHP and 3-LFSR-DDHP so that the intractability of 3-LFSR-DDHP provides the proposed scheme which is semantically secure.

Algorithm 3. Let q be a prime power, and let $f(x) = x^3 - ax^2 + bx - 1$ be an irreducible polynomial over $\mathbb{F}_q$ with $\text{per}(f) = \ell$ dividing $q^2 + q + 1$. Let $\mathcal{A}$ and $\mathcal{B}$ be two parties. $\mathcal{B}$ randomly selects a static private key $r \in \mathbb{Z}$ satisfying $0 < r < \ell$ such that $\gcd(r, \ell) = 1$, and computes his static public key $P_{\mathcal{B}} = (s_r, s_{-r}) \in \mathbb{F}_q^2$. $\mathcal{A}$ wants to send an encrypted message $M \in \mathbb{F}_q$ to $\mathcal{B}$, then both proceed as follows:

- ***Encryption:*** $\mathcal{A}$ encrypts a message $M \in \mathbb{F}_q$ as follows:
  - i) $\mathcal{A}$ randomly selects an ephemeral private key $t \in \mathbb{Z}$ satisfying $0 < t < \ell$ such that $\gcd(t, \ell) = 1$ and she then computes her ephemeral public key $(s_t, s_{-t}) \in \mathbb{F}_q^2$.
  - ii) $\mathcal{A}$ computes $(s_{tr}, s_{-tr}) = (s_t(s_r, s_{-r}), s_{-t}(s_r, s_{-r})) \in \mathbb{F}_q^2$ using the static public key $P_{\mathcal{B}} = (s_r, s_{-r})$ of $\mathcal{B}$.
  - iii) $\mathcal{A}$ computes $\bar{K} = (s_{tr} + s_{-tr})$ and $c = M\bar{K}$ and , sends the ciphertext $C = (s_t, s_{-t}, c)$ to $\mathcal{B}$.
- ***Decryption:*** $\mathcal{B}$ recovers the message $M \in \mathbb{F}_q$ as follows:
  - i) $\mathcal{B}$ computes $(s_{rt}, s_{-rt}) = (s_r(s_t, s_{-t}), s_{-r}(s_t, s_{-t}))$ using $\mathcal{A}$'s ephemeral public key (s_t, s_{-t}).
  - ii) $\mathcal{B}$ computes $\bar{K} = (s_{tr} + s_{-tr})$ and recovers message $M = c\bar{K}^{-1}$.

Remark 4.8 (Security Analysis). The security of key exchange in proposed encryption scheme depends on the difficulty of solving 3-LFSR-DLP, 3-LFSR-DHP and 3-LFSR-DDHP where as semantic security depends upon splitting $z \in \mathbb{F}_q$ in to two elements $(X, Y) \in \mathbb{F}_q^2$ such that $z = X + Y$, where $s_{tr} \mapsto X$ and $s_{-tr} \mapsto Y$. Moreover, $tr \in \mathbb{Z}_\ell$ and $\mathbb{Z}_e ll$ is chosen large enough so that the brute force attack becomes infeasible. The details are given in Remark 4.5.

Remark 4.9. $\mathcal{A}$ should validate the static public key of $\mathcal{B}$ so that the proposed scheme is invulnerable to small-subgroup attacks which may be effective for some discrete logarithm protocols [51]. It make sense that reusing ephemeral keys appears to be sound since the recent NIST SP 800-56A [79] standard mandates that all public keys be validated.

Table 4.5: Comparison of proposed encryption scheme with the similar ones

	ElGamal	**GH-RSA-Type**	**XTR-ElGamal**	**Proposed**
Encryption	$(2+2(9r^{\log 3}+13r-13)\log n)\mathbf{M}$ $+4r^{\log 3}\log n\ \mathbf{S}$	$10\ \mathbf{M}'$	$10.4\log n\ \mathbf{M}$	$(10.4\log n)\ \mathbf{M}$
Decryption	$(2+2(9r^{\log 3}+13r-13)\log n)\mathbf{M}$ $+4r^{\log 3}\log n\ \mathbf{S}$	$12\log N\ \mathbf{M}'$	$5.2\log n\ \mathbf{M}$	$(5.2\log n)\ \mathbf{M}$
Throughput	m_1	m_1, m_2	m_1	m_1
Comm. overhead in bits	$\|q\|$ with $q=p^r$	$2\|N\|$	$\|q\|$ with $q=p^2$	$2\|q\|$

Computational Cost: By improved singe exponentiation algorithm for GH construction, computing ephemeral key require $5.2\log n$ multiplications in $\mathbb{F}_q$ on average. Therefore, it requires about $(4+15.6\log n)$ field multiplications over $\mathbb{F}_q$ for both encryption and decryption of two field elements. On the other hand, if we apply our proposed scheme to XTR construction with two message elements, the total cost will be about $(4+31.2\log n)$ field multiplications over $\mathbb{F}_p$ because of the usage of two ephemeral keys. The detailed comparison is given in Table 5.6, where we enumerate the cost of field squarings, field multiplications and modulo N multiplications in terms of $\mathbf{S}$, $\mathbf{M}$ and $\mathbf{M}'$, respectively. The communication overhead is determined by the message length which is also included in Table 5.6.

4.5 GH-Nyberg-Rueppel-Type Digital Signature Algorithm

The Nyberg-Rueppel-type digital signature algorithm is one of the digital signature algorithm in which message recovery is also possible. Due to the faster and feasible encryption process of proposed encryption scheme, we propose GH-Nyberg-Rueppel-type digital signature algorithm (GH-NR-DSA) based on GH-ElGamal type encryption scheme so that key agreement, message recovery and digital signature protocol can be embedded together. Although, GH-DSA was already discussed in [30], as per our knowledge GH-NR-DSA is being proposed for the first time in this case. Since digital signatures with message recovery is useful for many applications in which message should be signed so that confidentiality, integrity and non-repudiation is ensured. There are two well known forgery attacks, namely the congruence equation attack and the homomorphism attack, applicable to Nyberg-Rueppel signature, the GH-NR-DSA has exhibits countermeasures for these attacks.

4.5.1 Algorithm 4 (GH-NR-DSA Based on GH-ElGamal Type Encryption Scheme)

Let q be a prime power, and let $f(x) = x^3 - ax^2 + bx - 1$ be an irreducible polynomial over $\mathbb{F}_q$ with $\text{per}(f) = \ell$ dividing $q^2 + q + 1$. Let $\mathcal{A}$ and $\mathcal{B}$ be two parties. $\mathcal{A}$ randomly selects a static private key $e \in \mathbb{Z}$ satisfying $0 < e < \ell$ such that $\gcd(e, \ell) = 1$, and computes her static public key $P_{\mathcal{A}} = (s_e, s_{-e}) \in \mathbb{F}_q^2$. $\mathcal{B}$ randomly selects a static private key $r \in \mathbb{Z}$ satisfying $0 < r < \ell$ such that $\gcd(r, \ell) = 1$, and computes his static public key $P_{\mathcal{B}} = (s_r, s_{-r}) \in \mathbb{F}_q^2$. Let H be a $\mathbb{Z}_\ell$-valued hash function. $\mathcal{A}$ wants to send a signed and encrypted message $M \in \mathbb{F}_q$ to $\mathcal{B}$ containing an agreed upon redundancy, then both proceed as follows:

(1) ***Signature***: $\mathcal{A}$ signs and encrypts the message M as follows:

(i) $\mathcal{A}$ randomly selects an ephemeral private key $t \in \mathbb{Z}$ satisfying $0 < t < \ell$ such that $\gcd(t, \ell) = 1$ and computes her ephemeral public key $(s_t, s_{-t}) \in \mathbb{F}_q^2$.

(ii) $\mathcal{A}$ computes hash value of M i.e. $h = H(M) \in \mathbb{Z}_\ell$. Then computes ciphertext C by encrypting the message M using proposed GH-ElGamal type encryption scheme in subsection 4.4 : $C = M(s_{tr}s_{-tr})$.

(iii) $\mathcal{A}$ computes $n = t - eh \mod \ell$.

(iv) $\mathcal{A}$ sends the signature $(n, (s_t, s_{-t}), C)$ to $\mathcal{B}$.

(2) ***Verification***: $\mathcal{B}$ recovers the message M and verifies $\mathcal{A}$'s signature as follows:

(i) $\mathcal{B}$ performs check $0 < n < \ell$, if not then failure.

(ii) $\mathcal{B}$ decrypts C to M using ephemeral public key (s_t, s_{-t}) and his static private key r : $M = C(s_{rt}s_{-rt})^{-1}$. If M does not contain the agreed upon redundancy, then failure.

(iii) $\mathcal{B}$ computes $h = H(M) \in \mathbb{Z}_\ell$ and $(s_{eh}, s_{-eh}) = (s_h(s_e, s_{-e}), s_{-h}(s_e, s_{-e}))$ using Algorithm 4, Lemma 4.1 and $\mathcal{A}$'s static public key (s_e, s_{-e}).

(iv) $\mathcal{B}$ computes $s_{\pm(eh-2)}$ using the characteristic sequence generated by $f(x)$ with $s_{\pm(eh-1)}$, $s_{\pm eh}$ and $s_{\pm(eh+1)}$ which are obtained in (iv).

(v) $\mathcal{B}$ computes $(s_{n+eh}, s_{-(n+eh)})$) using Algorithm 3, with $a = 1, b = n, (eh) \mapsto k, l \mapsto 1, s_{\pm(eh-2)}, s_{\pm(eh-1)}, s_{\pm eh}, s_{\pm 1}$.

(vii) $\mathcal{B}$ accepts if and only if $(s_{n+eh}, s_{-(n+eh)}) = (s_t, s_{-t})$.

4.5.2 Algorithm 5 (GH-NR-DSA Based on Generic Symmetric Encryption)

By keeping the system public parameters same as in Subsection 4.5.1, we have the following GH-NR-DSA based on the generic symmetric encryption.

(1) ***Signature***: $\mathcal{A}$ signs and encrypts the message M as follows:

(i) $\mathcal{A}$ randomly selects an ephemeral private key $t \in \mathbb{Z}$ satisfying $0 < t < \ell$ such that $\gcd(t, \ell) = 1$ and computes her ephemeral public key $(s_t, s_{-t}) \in \mathbb{F}_q^2$.

(ii) $\mathcal{A}$ computes symmetric key K based on (s_t, s_{-t}), $\mathcal{B}$'s static public key and encrypts message M to C using K and symmetric encryption scheme.

(iii) $\mathcal{A}$ computes hash value of C i.e. $h = H(C) \in \mathbb{Z}_\ell$.

(iv) $\mathcal{A}$ computes $n = t - eh \mod \ell$.

(v) $\mathcal{A}$ sends the signature (n, C) to $\mathcal{B}$.

(2) ***Verification***: $\mathcal{B}$ verifies $\mathcal{A}$'s signature and recovers the message M as follows:

(i) $\mathcal{B}$ performs check $0 < n < \ell$, if not then failure.

(ii) $\mathcal{B}$ computes $h = H(C) \in \mathbb{Z}_\ell$ and $(s_{eh}, s_{-eh}) = (s_h(s_e, s_{-e}), s_{-h}(s_e, s_{-e}),)$ using Algorithm 4, Lemma 4.1 and $\mathcal{A}$'s static public key (s_e, s_{-e}).

(iii) $\mathcal{B}$ computes $s_{\pm(eh-2)}$ using the characteristic sequence generated by $f(x)$ with $s_{\pm(eh-1)}$, $s_{\pm eh}$ and $s_{\pm(eh+1)}$ which are obtained in step 2.(ii).

(iv) $\mathcal{B}$ computes $(s_{n+eh}, s_{-(n+eh)})$ using Algorithm 3, with $a = 1, b = n, (eh) \mapsto k, l \mapsto 1, s_{\pm(eh-2)}, s_{\pm(eh-1)}, s_{\pm eh}, s_{\pm 1}$.

(v) $\mathcal{B}$ computes symmetric key K based on $(s_{n+eh}, s_{-(n+eh)})$, his static private key and decrypts C to M using symmetric encryption scheme. $\mathcal{B}$ accepts if and only if M contains agreed upon redundancy.

Security Analysis: It is well known from [56] that Nyberg-Rueppel signature scheme is vulnerable to two forgery attacks, which are congruence equation attack and homomorphism attack. GH-NR-DSA resistant to such attacks due to the following reasons.

(i) Congruence equation attack is avoided by using hash value of encrypted message and the agreed upon redundancy in the message.

(ii) The homomorphism attack is avoided by the fact that trace function is not an homomorphism, i.e., let $f(x)$ be an irreducible polynomial over $\mathbb{F}_q$ and $\alpha \in \mathbb{F}_{q^3}$ be any root of $f(x)$ of order ℓ dividing $q^2 + q + 1$. Then for any integer u and v,

$$\frac{Tr(\alpha^{u+v})}{Tr(\alpha^v)} = \frac{s_{u+v}}{s_v} \neq Tr(\alpha^u) = s_u,$$

where $\{s_n\}$ is called the third-order LFSR sequence over $\mathbb{F}_q$ generated by $f(x)$. It follows from this fact that LFSR based signature schemes resist this attack.

Remark 4.10. In Algorithm 4 and Algorithm 5, the general cryptographic hash functions published by NIST such as SHA family etc, can be used.

Table 4.6: Comparison of GH-NR-DSA with the similar DSAs

	GH-DSA	**XTR-NR-DSA**	**EC-DSA**	**GH-NR-DSA**
Double exponentiation	$(s_{c(h-dl)}, s_{-c(h-dl)})$	(s_{s+hk})	$(u_1 + u_2 d)P$	$(s_{n+eh}, s_{-(n+eh)})$
Cost	$16 \log \ell$ **M** $+ 16 \log \ell$ **S**	$6 \log \ell$ **M**	$7 \log \ell$ **M** $+ 3.7 \log \ell$ **S**	$10.4 \log \ell$ **M**
Throughput	m_1, m_2	m_1	m_1	m_1
Message recovery	No	Yes	No	Yes
Comm. overhead in bits	$2\|q\| + \|H\|$	$\|q\| + \|H\|$	$\|p\| + \|H\|$	$2\|q\| + \|H\|$

Computational Cost: In GH-DSA and GH-NR-DSA, there is a double exponentiation in $\mathbb{F}_q$, which the elements of order ℓ dividing $q^2 + q + 1$ in $\mathbb{F}_{q^3}$. In XTR-NR-DSA, there is also a double exponentiation in $\mathbb{F}_q$ with $q = p^2$, which the elements of order ℓ dividing $p^2 - p + 1$ in $\mathbb{F}_{p^6}$. For EC-DSA, $E(\mathbb{F}_p)$ is the group of points of an elliptic curve E over $\mathbb{F}_p$ and $P \in E(\mathbb{F}_p)$ is a point of prime order ℓ. GH-NR-DSA is efficient than GH-DSA and EC-DSA by considering the computational cost and the message recovery. The detailed comparison is given in Table 5.8, where we enumerate the cost of field multiplications and field squarings in terms of **M** and **S**, respectively. In Table 5.8, $h \in \mathbb{Z}_Q$ is the hash of encrypted message, $c(h - dl), s + hk, u_1 + u_2 d, n + eh \in \mathbb{Z}_Q$.

4.6 Conclusion

In this chapter we introduced GH-ElGamal type encryption scheme, which is semantically secure and semantic security depends upon splitting $z \in \mathbb{F}_q$ into two numbers $(X, Y) \in \mathbb{F}_q^2$ such that $z = X + Y$ where $s_e \mapsto X$ and $s_{-e} \mapsto Y$. Moreover, $e \in \mathbb{Z}_\ell$ and $\mathbb{Z}_e ll$ is chosen large enough so that the brute force attack becomes infeasible. Whereas the key exchange security depends on the difficulty of solving 3-LFSR-DLP and 3-LFSR-DHP. This scheme is faster than GH-RSA-type encryption scheme with lesser storage memory. Based upon the proposed encryption scheme, we also introduced GH-NR-DSA to embed key exchange, message transmission and digital signature algorithm in a single protocol to ensure confidentiality, integrity and non-repudiation.

CHAPTER 5

TRACE BASED PUBLIC KEY CRYPTOSYSTEMS

5.1 Introduction

The Discrete Log Problem (DLP), that is computing x, given $y = \alpha^x$ and $< \alpha >= G \subset \mathbb{F}_q^*$, based Public Key Cryptosystems (PKC) are being studied since late 1970's. Such development of PKC was possible because of the trapdoor function $f : \mathbb{Z}_\ell \to G =< \alpha > \subset \mathbb{F}_q^*$, $f(m) = \alpha^m$ is a group homomorphism. Due to this fact we have; Diffie-Hellman (DH) type key exchange, ElGamal type message encryption, and Nyberg Rueppel type digital signature protocols. The cryptosystems based on the trapdoor $f(m) = \alpha^m$ are well understood and complete. However, there is another trapdoor function $f : \mathbb{Z}_\ell \to G$, $f(m) \to Tr(\alpha^m)$, where $G =< \alpha > \subset \mathbb{F}_{q^k}^*$, $k \geq 2$, which needs more attention of the researchers from cryptographic protocols point of view. In the above mentioned case, even though f is a computable, but it is not clear how to produce protocols such as Diffie Hellman type key exchange, ElGamal type message encryption, and Nyberg Rueppel type digital signature algorithm, in general. It would be better, of course if we can find more efficient algorithm than repeated squaring and trace to compute $f(m) = Tr(\alpha^m)$ together with these protocols. In the literature we see some works for a more efficient algorithm to compute $f(m) = Tr(\alpha^m)$ and not wondering about the protocols. We also see some works dealing with an efficient algorithm to compute $Tr(\alpha^m)$ as well as discussing the cryptographic protocols. These works are presented by Smith, Lennon and Skinner (LUC-PKC) in 1994 [71, 72], L.Horn and G.Gong, (GH-PKC) in 1998 [30, 31], A.K.Lenstra and E.R.Verheul (XTR-PKC) in 2000 [47, 48, 16, 73] and K.Giuliani, G.Gong (GG-PKC) in 2003 [27, 28] and Koray Karabina (KK-PKC) in 2009 [41, 42].

In the literature there are also so called torus based cryptosystems introduced by K.Rubin and A.Silverberg in 2003 [64]. These systems depend on the parametric representation of the group $\mathbb{T}_k(\mathbb{F}_q) \cong G_{q,k} =< \alpha > \subset \mathbb{F}_{q^k}^*$ and $\mathbb{T}_k(\mathbb{F}_q)$ rather than the efficient algorithm to compute $Tr(\alpha^m)$, where $\mathbb{T}_k(\mathbb{F}_q)$ is torus consisting of the elements in $\mathbb{F}_{q^k}$ whose norm in 1 down to every intermediate subfield. Therefore, torus based cryptography is an area which does not fit to philosophy of this study. Therefore, we shall not discuss it here.

The aim of this chapter is to review the so called trace based cryptography studied

in the literature and bring to the attention the challenges faced in these systems. We will not only review these systems but also introduce cryptographic protocols for the ones they are not discussed in the literature.

This chapter is organized as follows: in section 5.2, we will review the necessary mathematical background and protocols for DLP based PKC. In section 5.3, we review the trace based systems including mathematical structure for efficient algorithm to compute $f(m) = Tr(\alpha^m)$ and cryptographic protocols such as Diffie Hellman key exchange, ElGamal type encryption scheme and Nyberg Rueppel type Digital Signature algorithm. Although, most of the facts discussed here are well known however, we have also added some new results such as ElGamal type encryption and Nyberg Rueppel type signature schemes for some cases which were not discussed in the literature. We also introduce efficient exponentiation for 5th degree extensions in subsection 5.3.10. Finally, we conclude the chapter in section 5.4.

5.2 Discrete Log Based PKC over Finite Fields

The general mathematical structure of the PKCs based on finite field and cryptographic protocols such as Key Exchange, Encryption scheme, and Nyberg Rueppel Digital signature algorithm are basic building blocks for the PKC. Also the PKC is considered practical if we have cryptographic protocols and efficient algorithm for computations involved in these protocols. Keeping this in focus, we discuss DLP based PKC over subgroup G of a finite field.

5.2.1 DLP Based PKC over G

The DLP based PKC over G were discussed in Chapter 2 the brief introduction is also given here Let,

$$\begin{aligned} &G =< \alpha > \subset \mathbb{F}_q^*, \quad q = p^r, \text{ i.e; prime power} \\ &G =< \alpha > \cong \mathbb{Z}_\ell, \quad ord(\alpha) = \ell \\ &f : \mathbb{Z}_\ell \to G; \text{ with } f(m) = \alpha^m; \end{aligned}$$

The f is a computable trapdoor, and well known repeated squaring algorithm computes f in polynomial time. Therefore by choosing the order G a large prime ℓ one can avoid polynomial time algorithms to solve DLP so that f becomes one-way function on $\mathbb{F}_q^*$. On the DLP based PKC, the main point is the trapdoor function:

$$\begin{aligned} &G =< \alpha > \subset \mathbb{F}_q^*, \quad ord(\alpha) = \ell, \\ &f : \mathbb{Z}_\ell \longrightarrow \mathbb{F}_q^*, \quad f(m) = \alpha^m. \end{aligned}$$

The trapdoor function f is a group homomorphism; $f(m+k) = f(m)f(k)$, and one should be careful to choose $< \alpha >= G \subset \mathbb{F}_q^*$ to avoid algorithms to attack

DLP on G and choose, if possible, the smallest size G such that DLP on G is computationally equivalent to DLP on $\mathbb{F}_q^*$.

5.3 Trace Based Public Key Cryptosystems over Finite Fields

In this section we will discuss LUK-PKC, GH-PKC, GG-PKC, and KK-PKC. Let α be an element over $\mathbb{F}_{q^k}$ with the characteristic polynomial;

$$g(x) = x^k - a_1 x^{k-1} + \cdots + (-1)^k a_k,$$

over $\mathbb{F}_q$ and let, Tr, $Norm$: $\mathbb{F}_{q^k} \to \mathbb{F}_q$, given by;

$$Tr(\beta) = \sum_{i=0}^{k-1} \beta^{q^i}, \quad Norm(\beta) = \prod_{i=0}^{k-1} \beta^{q^i}.$$

Note that,

$$g(x) = \prod_{i=0}^{k-1} (x - \alpha^{q^i}),$$

and $a_i = \sigma_i(\alpha, \alpha^q, \cdots, \alpha^{q^{k-1}})$, σ_i is the i-th elementary symmetric function in k variables. For any integer m, let $g_m(x)$ be the characteristic polynomial of α^m,

$$\begin{aligned} g_m(x) &= \prod_{i=0}^{k-1} (x - (\alpha^m)^{q^i}), \\ &= x^k - f_1(m)x^{k-1} + \cdots + (-1)^i f_i(m) x^{k-i} + \cdots + (-1)^k f_k(m). \end{aligned}$$

Note that; $g_1 = g$.

This gives us for $i = 1, 2, \cdots, k$,

$$f_i : \mathbb{Z}_n \longrightarrow \mathbb{F}_q^*, \quad f_i(m) = \sigma_i(\alpha^m, (\alpha^m)^q, \cdots, (\alpha^m)^{q^{k-1}}).$$

5.3.1 Relation of $f_i(m)$ with LFSR

Let, α be an element of finite field $\mathbb{F}_{q^k}$, with characteristic polynomial over $\mathbb{F}_q$:

$$g(x) = x^k + b_1 x^{k-1} + \cdots + b_k.$$

For a given initial state $\{s_0, s_1, \cdots, s_{k-1}\}$ one produces a unique periodic sequence, $\{s_m = -b_1 s_{m-1} - b_2 s_{m-2} - \cdots - b_k s_{m-k}, \quad m \geq k\}$, generated by $g(x)$. The general term s_n can be expressed uniquely in the form:

$$s_n = c_1 \alpha_1^n + c_2 \alpha_2^n + \ldots + c_k \alpha_k^n, \text{ for some } c_i \in \mathbb{F}_q,$$

and

$$g(x) = \prod_{i=1}^{k} (x - \alpha_i), \quad \alpha_i \in \mathbb{F}_{q^k}.$$

Remark 5.1. It is well known that, finding an efficient algorithm to compute $f_i(m) = \sigma_i(\alpha^m, (\alpha^m)^q, \cdots, (\alpha^m)^{q^{k-1}})$ is computationally equivalent to finding algorithm to compute s_n, with given specific initial conditions $s_0, \ldots, s_{k-1}$, where for $i = 1, 2, \cdots, k$; $\alpha_i = (\alpha^m)^{q^{i-1}}$.

Remark 5.2. We note that by means of Newton's formula having an efficient algorithm to compute $f_1(m) = Tr(\alpha^m)$ is computationally equivalent to finding such an algorithm for $f_i(m)$.

From now on our work will be restricted to the function $f_1(m) = Tr(\alpha^m)$ and for simplicity we denote f_1 as f. For $m = 1, 2, \cdots$ and $\alpha \in \mathbb{F}_{q^k}$, let,

$$g_m(x) = \prod_{i=0}^{k-1} (x - (\alpha^m)^{q^i}),$$
$$\widetilde{g}_m(x) = \prod_{i=0}^{k-1} (x - (\alpha^{-m})^{q^i}).$$

Here $\widetilde{g}_m(x) = x^k g_m(\frac{1}{x})$, is the reciprocal of $g_m(x)$.

Note that; for $f : \mathbb{Z}_\ell \to \mathbb{F}_q^*$, $f(m) = Tr(\alpha^m)$, computing $f^{-1}(\beta)$ is computationally equivalent to solving DLP on $G =< \alpha > \subset \mathbb{F}_{q^k}^*$. Since $f(m) = Tr(\alpha^m)$ is not a group homomorphism, it is not known, in general, how to obtain cryptographic protocols like the the ones in DLP case based on $\alpha \to \alpha^m$. Certainly, the PKC based on $f(m) = Tr(\alpha^m)$ will give more compressed system versus the system based on $\alpha \to \alpha^m$ in $\mathbb{F}_{q^k}^*$. With this development we define the trace based PKC as follows:

Definition 5.1. By a trace based cryptosystem we mean a Public Key Cryptosystem based on the one-way function,

$$f : \mathbb{Z}_\ell \to G =< \alpha > \subset \mathbb{F}_{q^k}^*,$$
$$f(m) = Tr(\alpha^m).$$

We mean a cryptosystem having an adaptability to cryptographic protocols such as, DH type key exchange, ElGamal type message encryption scheme, and Nyberg Rueppel type digital signature algorithm, based on f.

5.3.2 Main Challenges for Trace Based Cryptography

Let α be an element of $\mathbb{F}_{q^k}$ having characteristic polynomial $g(x) = x^k - a_1 x^{k-1} + \cdots + (-1)^k a_k$ over $\mathbb{F}_q$, **find conditions on α such that**:

(i) There is more efficient polynomial time algorithm than the algorithm that first computes α^m using square-and-multiply algorithm, then computes trace of α^m, for a given $m \in \mathbb{Z}$.

(ii) Introduce cryptographic protocols similar to protocols discussed above for DLP-based system.

(iii) Compare the security and implementation of these protocols with the existing ones.

Remark 5.3. The problem (i) arose also in paring based cryptosystems and there have been some works done there for some values of k and special conditions on α such as $ord(\alpha) = \ell | \Phi_k(q)$, where $\Phi_k(q)$ is kth cyclotomic polynomial evaluated at q.

By keeping in view the above challenges, we now discuss the related work done in the literature and in this respect following is the list of Public Key Cryptosystems.

(i) LUC-PKC : $k = 2, \quad q = prime\ power$, p-arbitrary;[71, 72].

(ii) GH-PKC : $k = 3, \quad q = prime\ power$, p-arbitrary;[30, 31].

(iii) GH-PKC Special Case : $k = 3, \quad q = p^2, \quad p$-arbitrary;[47, 48, 16].

(iv) GG PKC : $k = 5, \quad q = prime\ power$, p-arbitrary;[27].

(v) GG-PKC Special Case : $k = 5, \quad q = p^2, \quad p$-arbitrary; [27].

(vi) KK-PKC : $k = 4$, $p = 2$, $q = p^{2u+1}, u \geq 1$

(vii) KK-PKC : $k = 6$, $p = 3$, $q = p^{2u+1}, u \geq 1$

5.3.3 Motivations to use Trace Based Public Key Cryptography

The main reasons for developing cryptographic protocols based on Trace Based construction are as follows:

(i) **Security:** The trace based construction provide security of extension field $\mathbb{F}_{q^k}$ that is DLP lies in $\mathbb{F}_{q^k}$ with faster exponentiation and compressed data for transmission.

(ii) **Transmission Size:** Due to compression of operands the overall data transmission size is also reduced to implement PKC protocols as given in table below:

(iii) The cost of exponentiation is lesser than the generic square and multiply algorithm for finite field and elliptic curve ($E(\mathbb{F}_q)$) based exponentiation. The details are given in the table below:

Table 5.1: Trace Based Compression Factors for Cryptographic Protocols

Extension Field	**Compression Factor**	**PKC System**
$\alpha^n \in \mathbb{F}_{q^2}, q = p^r, r \in \mathbb{Z}$	1/2	LUC-PKC
$\alpha^n \in \mathbb{F}_{q^3}, q = p^2,$	1/3	GH-PKC and XTR-PKC
$\alpha^n \in \mathbb{F}_{q^5}, q = p^2,$	2/5	GG-PKC
$\alpha^n \in \mathbb{F}_{q^6}, q = 3^{2r+1}, r \in \mathbb{Z}$	1/6	KK-PKC
$\alpha^n \in \mathbb{F}_{q^4}, q = 2^{2r+1}, r \in \mathbb{Z}$	1/4	KK-PKC

Table 5.2: Exponentiation Comparison over $\mathbb{F}_{q^2}$

Process	**Multiplications over $\mathbb{F}_q$**	**Squaring over $\mathbb{F}_q$**
$\alpha^n \in \mathbb{F}_{q^2}, q = p^r, r \in \mathbb{Z}$	$weight(n) - 1$	$\lfloor \log n \rfloor$
$s_n, \alpha^n \in \mathbb{F}_{q^2}$	$0.75 \log n$	$0.17 \log n$
$nP, P \in E(\mathbb{F}_{q^2})$	$7 \log n$	$3.6 \log n$

Table 5.3: Exponentiation Comparison over $\mathbb{F}_{q^3}$

Process	**Multiplications over $\mathbb{F}_q$**	**Squaring over $\mathbb{F}_q$**
$\alpha^n \in \mathbb{F}_{q^3}, q = p^r, r \in \mathbb{Z}$	$(7r^{\log 3} + 10.8r - 10.8) \log n$	$3.8r^{log3}logn$
$s_n, \alpha^n \in \mathbb{F}_{q^3}$	$5.2 \log n$	-
$nP, P \in E(\mathbb{F}_q)$	$4 \lceil \log n \rceil + 12(weight(n))$	$4 \lceil \log n \rceil + 4(weight(n))$

Table 5.4: Exponentiation Comparison over $\mathbb{F}_{q^5}$

Process	**Multiplications over $\mathbb{F}_p$**	**Squaring over $\mathbb{F}_p$**
$\alpha^n \in \mathbb{F}_{q^5}, q = p^2,$	$45(weight(n) - 1)$	$30 \lfloor \log n \rfloor$
$s_n, \alpha^n \in \mathbb{F}_{q^5}, q = p^2$	$108.5 \log n$	$13 \log n$

5.3.4 Algorithm to compute mixed term $s_{u+v} = Tr(\alpha^u \alpha^v)$, given $Tr(\alpha), u, S_v = (s_{v-k+1} \cdots s_v)$, $s_j = Tr(\alpha^j)$ for $j \in \mathbb{Z}$, and v-unknown.

Before discussing PKC systems we give generalized algorithm to compute mixed term that is $s_{u+v} = Tr(\alpha^u \alpha^v)$ given $u, Tr(\alpha), S_v = (s_{v-k+1}, \cdots, s_v), v$-unknown. This algorithm will be required in Nyberg-Rueppel type digital signature system. Note that the algorithm to compute $Tr(\alpha^u)$ given $Tr(\alpha)$ and u will be discussed in respective PKC and here we assume such an algorithm already exists. In fact, one can always take repeated squaring and trace for such computations.

Let $G =< \alpha > \subset \mathbb{F}_{q^k}^*$, p-arbitrary, $q = p^r$, and $g(x)$ be the characteristic polynomial of α :

$$g(x) = \prod_{i=0}^{k-1} (x - \alpha^{q^i}) = x^k - a_1 x^{k-1} + a_2 x^{k-2} - \cdots + (-1)^{k-1} a_{k-1} + (-1)^k.$$

Let A be companion matrix associated with $g(x)$:

$$A = \begin{pmatrix} 0 & 0 & \cdots & 0 & (-1)^{k-1} \\ 1 & 0 & \cdots & 0 & (-1)^{k-2} a_{k-1} \\ 0 & 1 & \cdots & 0 & (-1)^{k-3} a_{k-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & a_1 \end{pmatrix}. \tag{5.1}$$

Using the algorithm for $Tr(\alpha^u)$, we form the state matrix M_u associated with u as::

$$M_u = \begin{pmatrix} s_{u-k+1} & s_{u-k+2} & \cdots & s_{u-1} & s_u \\ s_{u-k+2} & s_{u-k+3} & \cdots & s_u & s_{u+1} \\ s_{u-k+3} & s_{u-k+4} & \cdots & s_{u+1} & s_{u+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ s_u & s_{u+1} & \cdots & s_{u+k-2} & s_{u+k-1} \end{pmatrix}. \tag{5.2}$$

Let the vector $S_m = (s_{m-k+1}, \cdots, s_m)$ be given for any integer m then we have,

$$S_{m+1} = S_m A,$$

and therefore for any integer r,

$$S_{m+r} = S_m A^r.$$

This gives, $M_u = M_0 A^u$ therefore, $A^u = M_0^{-1} M_u$ and vector,

$$S_{u+v} = S_v (M_0^{-1} M_u) = (s_{u+v-k+1}, \cdots, s_{u+v}). \tag{5.3}$$

Algorithm 5 to compute $S_{u+v} = (s_{u+v-k+1}, \cdots, s_{u+v})$ for v-unknown

Require: $k \geq 2, S_v, u, Tr(\alpha)$.
Ensure: S_{u+v}.
1: Construct A as in equation (5.1),
2: Construct M_u as in equation (5.2),
3: Compute, M_0^{-1},
4: Compute, $A^u \leftarrow M_0^{-1} M_u$,
5: Compute, $C \leftarrow S_v(A^u)$.
6: **return** (C).

Remark 5.4. Following is the step by step computational complexities:

(i) The computational cost to construct M_u is the cost of algorithm to compute $S_u + O(k^3)$ field multiplications.

(ii) M_0 can be constructed form initial states with $O(k^3)$ field multiplications.

(iii) The complexity of computing M_0^{-1} and $M_0^{-1}M_u$ is also bounded by $O(k^3)$. Therefore, $s_{u+v} = Tr(\alpha^{u+v})$ can be computed in about $O(k^3)$ field multiplications.

5.3.5 LUC-PKC : Case $k = 2$, p-arbitrary, $q = p^r$, for any positive integer r

Let $G =< \alpha >\cong \mathbb{Z}_\ell \subset \mathbb{F}_{q^2}^*, ord(\alpha) = \ell | (q+1)$, then, the polynomial

$$g(x) = x^2 - ax + 1, \quad a = Tr(\alpha), \quad Norm(\alpha) = 1, \text{ and}$$
$$g_m(x) = (x - \alpha^m)(x - (\alpha^m)^q) = x^2 - V_m(a)x + 1, \ m \in \mathbb{Z}.$$
Note that, $\tilde{g}(x) = g(x)$ and $\tilde{g}_m(x) = g_m(x)$.

Notation: For any integer r, let $V_r : \mathbb{F}_q \to \mathbb{F}_q$ be a function given by $V_r(b) = Tr(\beta^r) = s_r$, where β is a root of $x^2 - bx + 1 = 0$. It is clear that V_r is well defined, namely it is independent than the choice of the root β of $x^2 - bx + 1 = 0$. Note that for any integer r

$$h_r(x) = \prod_{i=0}^{1} (x - \beta^{rq^i}) = x^2 - V_r(b)x + 1 = x^2 - s_r x + 1,$$

in particular, $s_0 = 2, s_1 = b$. Furthermore, for any integer m, we have

$$h_{mr}(x) = \prod_{i=0}^{1} (x - (\beta^{mr})^{q^i}) = x^2 - V_m(V_r(b))x + 1x^2 - s_{mr}x + 1$$

over $\mathbb{F}_q$. Which implies that, for any integers r and m we have $V_m(V_r(b)) = V_{mr}(b) = s_{mr}$.

The LFSR sequence $\{s_m\}$ with initial conditions $\{2, a\}$ associated to characteristic polynomial $g(x) = x^2 - ax + 1$, gives us the general term:

$$s_m = V_m(a) = Tr(\alpha^m).$$

Properties: Following properties for all integers m and n are proved in [71] and we list them;

(i) $V_m(a) = V_{-m}(a) = s_m$;

(ii) $V_m(s_n) = V_{mn}(a) = s_{mn}$;

(iii) $s_{2m} = s_m^2 - 2$;

(iv) $s_{m+n} = s_m s_n - s_{m-n}$.

These properties give us the following efficient algorithm to compute $s_m = Tr(\alpha^m)$.

Algorithm 6 to compute $s_m = Tr(\alpha^m)$

Require: $\alpha \in \mathbb{F}_{q^2}$, $ord(\alpha) = \ell|(q+1)$, $Tr(\alpha) = a \in \mathbb{F}_q$ and $m = \sum_{j=0}^{t-1} \epsilon_j 2^j \in \mathbb{Z}$ with $\epsilon_j = \{0, 1\}$, $\epsilon_{t-1} = 1$.
Ensure: (s_m, s_{m+1})
1: $(s_y, s_{y+1}) \leftarrow (2, a)$
2: **for** $j \leftarrow t-1$ *to* 0 **do**
3: **if** $\epsilon_j = 1$ **then**
4: $s_y \leftarrow s_y s_{y+1} - s_1$, $s_{y+1} \leftarrow s_{y+1}^2 - 2$.
5: **else**
6: $s_y \leftarrow s_y^2 - 2$, $s_{y+1} \leftarrow s_y s_{y+1} - s_1$.
7: **end if**
8: **end for**
9: **return** (s_y, s_{y+1})

Remark 5.5. For any integer m this algorithm actually computes (s_{m-1}, s_m) from the initial conditions $\{2, a\}$.

This algorithm is more efficient than first computing α^m and then $Tr(\alpha^m)$. In fact this algorithm does compute $Tr(\alpha^m)$ by using $0.75 \log m$ multiplications and $0.75 \log m$ squaring over $\mathbb{F}_p$.

5.3.5.1 LUC-DH Type Key Exchange

1. ***System Public Parameters:*** Let $G =< \alpha > \subset \mathbb{F}_{q^2}^*$, $ord(\alpha) = \ell|(q+1)$, and $g(x) = x^2 - ax + 1$, be the characteristic polynomial of α.

2. ***Private Keys:*** Random m, and r satisfying $1 < e, r < \mathbb{Z}_\ell$ are the private keys of **A** and **B** respectively.

3. ***Public Keys:*** Both A and B computes their public keys as follows:

   (i) **A** computes her public key $P_A = V_e(a) = s_e$, by running algorithm 6 with inputs $a = Tr(\alpha)$ and her private key e.

   (ii) **B** computes his public key $P_B = V_r(a) = s_r$, by running algorithm 6 with inputs $a = Tr(\alpha)$ and his private key r.

4. **Common key:** A and B agrees on the common key $K = P_{AB} = P_{BA} = s_{er}$ as follows:

   (i) **A** acquires **B**'s public key and constructs $g_r(x) = x^2 - P_B x + 1 = x^2 - s_r x + 1$. Then she computes $g_{er}(x) = x^2 - s_{er} x + 1$, by running algorithm 6 with inputs $P_B = s_r$ and her private key e such that,

$$K = P_{AB} = V_e(P_B) = V_e(s_r) = V_{er}(a) = s_{er}.$$

(ii) **B** does the similar things to compute,

$$K = P_{BA} = V_r(P_A) = V_r(s_e) = V_{re}(a) = s_{re}.$$

5.3.5.2 LUC-ElGamal Encryption Scheme

1. ***System Public Parameters:*** Let $G =< \alpha >\subset \mathbb{F}^*_{q^2}$, $ord(\alpha) = \ell|(q+1)$, and $g(x) = x^2 - ax + 1$, be the characteristic polynomial of α.
2. **B** runs algorithm 6 with inputs a and his static private key $1 < r < \mathbb{Z}_\ell$ to compute his static public key $P_B = s_r$, and publishes P_B.
3. Assumption: messages $M \in G$.
4. **A** sends message M as follows:
   (i) Chooses a random ephemeral private key $1 < t < \mathbb{Z}_\ell$ and runs algorithm 6 with inputs a and r to compute ephemeral public key s_t.
   (ii) Acquires **B**'s public key $P_B = s_r$ and computes encryption key $K = V_t(P_B) = V_t(s_r) = V_{rt}(a) = s_{rt}$, by running algorithm 6 with inputs $P_B = s_t$ and session key r.
   (iii) **A** encrypts M by computing $c = KM = s_{rt}M$ and sends ciphertext $C = (s_t, c)$.
5. ***Decryption:*** **B** decrypts C as follows:
   (i) Based upon ephemeral public key s_t and his private key r **B** computes encryption key $K = V_r(s_t) = V_{tr}(a) = s_{tr}$.
   (ii) Decrypts c to obtain M with encryption key K (computed in 5(i)), such that $M = K^{-1}c = s_{tr}^{-1}c$.

Remark 5.6. The LUC-ElGamal Encryption Scheme is semantically insecure. The attacker can identify the encryption of either m_1 or m_2 without solving DLP over $\mathbb{F}_{q^2}$ as discussed in chapter 4 Remark 4.5.

5.3.5.3 LUC-Nyberg Rueppel type Digital Signature Algorithm Based on LUC-ElGamal Encryption

Let, **A** wants to send signed message $M \in \mathbb{F}_q$ to **B** and **B** verifies it. To do this, they proceed as follows:

1. ***System Public Parameters:*** Let $G =< \alpha >\subset \mathbb{F}^*_{q^2}$, $ord(\alpha) = \ell|(q+1)$, and $g(x) = x^2 - ax + 1$, be the characteristic polynomial of α, and $H : G \to \mathbb{Z}_\ell$ valued hash function.
2. ***Public Keys:***

(i) **A** randomly selects $0 < e < \mathbb{Z}_\ell$ and computes public key $P_{\mathbf{A}} = s_e$ by running algorithm 6 with inputs a and her secret key e.

(ii) **B** randomly selects $0 < r < \mathbb{Z}_\ell$ and computes his public key $P_{\mathbf{B}} = s_r$ by running algorithm 6 with inputs a and his secret key r.

3. ***Private Keys:*** The private keys of **A** and **B** are e and r, respectively.

4. ***Signature***: **A** signs the message $M \in G$ as follows:

(i) Randomly selects private ephemeral key $0 < t < \mathbb{Z}_\ell$ and computes ephemeral public key s_t, by running algorithm 6 with inputs a, t and computes encryption key $V_t(s_r) = V_{tr}(a) = s_{tr}$ by running same algorithm with inputs s_r and ephemeral private key t.

(ii) computes the hash value of M i.e, $h = H(M) \mod \ell$ and computes ciphertext C by encrypting the message M.

(iii) Computes $n = t - he \mod \ell$.

(iv) **A** sends the signature (n, s_t, C) to **B**.

5. ***Verification***: **B** recovers message M and verifies **A**'s signature as follows:

(i) Computes computes encryption key s_{rt} by using Algorithm 6 with inputs r, s_t and decrypts C to M : $M = Cs_{rt}^{-1}$. If M contains required redundancy **B** proceeds.

(ii) Computes $h = H(M) \in \mathbb{Z}_\ell$,

(iii) By running algorithm 6 with inputs $P_A = s_e$ and h, computes $V_h(s_e) = V_{he}(a) = s_{he}$. Note that, as stated in remark 5.5, algorithm 6 computes both s_{he} as well as s_{he-1}.

(iv) Computes $s_{n+he} = \acute{s}_t$ by running algorithm 5 with inputs $k = 2, S_{he} = (s_{he-1}, s_{he}), n$.

(v) **B** accepts if $s_t = \acute{s}_t$.

The analysis of LUC-NR-DSA is given Table 5.5 below:

Table 5.5: LUC-NR-DSA Analysis

Process	**Cost**
Signature Generation	1 exponentiation $(0.75 \log q)\mathbf{M}+(0.75 \log q)\mathbf{S}$
Signature Verification	1 exponentiation $(1.5 \log q)\mathbf{M}+(1.5 \log q)\mathbf{S}$
Comm. Overhead	1 element over $\mathbb{F}_q$
Public Key size	1 element over $\mathbb{F}_q$

where **M** stands for multiplication and **S** stands for squaring.

5.3.6 GH-PKC : Case $k = 3$, p-arbitrary, $q = p^u$, $u \geq 1$

Let, $G =< \alpha >\cong \mathbb{Z}_\ell \subset \mathbb{F}_{q^3}^*$, $\alpha \in \mathbb{F}_{q^3}$, $ord(\alpha) = \ell | (q^2 + q + 1)$. Then, one can check that characteristic polynomial $g(x)$ of α and its reciprocal $\tilde{g}(x)$ have the following properties;

$$g(x) = \prod_{i=0}^{2} (x - \alpha^{q^i}) = x^3 - ax^2 + bx - 1, \text{ and}$$
$$\tilde{g}(x) = \prod_{i=0}^{2} (x - \alpha^{-q^i}) = x^3 - bx^2 + ax - 1,$$

namely $a = Tr(\alpha)$, $b = Tr(\alpha^{-1})$. In fact, for all integers m, let $g_m(x)$, $\tilde{g}_m(x)$ be the following polynomials;

$$g_m(x) = \prod_{i=0}^{2} (x - (\alpha^m)^{q^i}) = x^3 - s_m x^2 + s_{-m} x - 1, \text{ and}$$
$$\tilde{g}_m(x) = \prod_{i=0}^{2} (x - (\alpha^{-m})^{q^i}) = x^3 - s_{-m} x^2 + s_m x - 1,$$

where $s_m = Tr(\alpha^m)$ and $s_{-m} = Tr(\alpha^{-m})$.

Notation: For any integer r let $V_r : \mathbb{F}_q^2 \to \mathbb{F}_q$ be a function given by $V_r(c, d) = Tr(\beta^r) = s_r$, where β is a root of $x^3 - cx^2 + dx - 1 = 0$. It is clear that V_r is well defined, namely it does not depend on the choice of a root β of $h(x) = 0$. In particular, $s_0 = 3, s_1 = c$ and $s_2 = c^2 - 2d$. It is clear that:

$$h_r(x) = \prod_{i=0}^{2} (x - (\beta^r)^{q^i}) = x^3 - V_r(c, d)x^2 + V_{-r}(c, d)x - 1 = x^3 - s_r x^2 + s_{-r} x - 1,$$

and for any $m \in \mathbb{Z}$ we also have, $V_m(V_r(c, d), V_{-r}(c, d)) = V_{mr}(c, d) = s_{mr} = Tr(\beta^{mr})$.

The LFSR sequence $\{s_i\}$ with initial conditions $\{3, a, a^2 - 2b\}$ associated to $g(x) = x^3 - ax^2 + bx - 1$, gives us the general term $s_n = V_n(a, b) = Tr(\alpha^n)$,for any integer n.

Properties: For all integers m and n following properties are proved in [30] and we list here;

(i) $V_m(s_n, s_{-n}) = V_{mn}(a, b) = s_{mn}$,

(ii) $s_{2m} = s_m^2 - 2s_{-m}$,

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

The cryptographic protocols related to GH-PKC construction are given in Chapter 4

5.3.7 GH-PKC : Special Case $k = 3$, p-arbitrary, $q = p^2$

We keep the same notations above and now assume that $ord(\alpha) = \ell|(p^2 - p + 1)$. Note that $(q^2+q+1) = (p^2-p+1)(p^2+p+1)$. In the case $q = p^2$, the important point is that $Tr(\alpha^{-m}) = Tr(\alpha^m)^p = s_m^p$ for any integer m which was shown in [47]. Therefore for any integer m we get,

$$g_m(x) = \prod_{i=0}^{2} (x - (\alpha^m)^{q^i}) = x^3 - s_m x^2 + s_m^p x - 1,$$

where $s_m = Tr(\alpha^m)$.

The LFSR sequence $\{s_m\}$ associated to $g(x) = x^3 - ax^2 + bx - 1$, with initial conditions $\{3, a, a^2 - 2b\}$ gives us the general term:

$$\begin{aligned} s_m &= V_m(a,b) = Tr(\alpha^m), \text{ and} \\ s_{-m} &= V_{-m}(a,b) = V_m(a,b)^p = Tr(\alpha^m)^p = s_m^p. \end{aligned}$$

Properties: For all integers m and n following properties are consequences in [47] and they are the special case of GH-PKC;

(i) $V_{-m}(a,b) = V_m(a,b)^p = s_m^p$.

(ii) $V_m(V_n(a,b), V_n(a,b)^p) = V_{mn}(a,b) = s_{mn}$.

(iii) $s_{2m} = s_m^2 - 2s_m^p$.

(iv) $s_{m+n} = s_m s_n - s_{m-n} s_n^p + s_{m-2n}$.

(v) $s_m + s_m^p = Tr(Tr(\alpha^m))$ over $\mathbb{F}_p$.

Remark 5.7. Computing $Tr(Tr(\alpha^m))$ require negligible arithmetic operation that is one addition over $\mathbb{F}_{p^2}$.

Note that the advantage of special conditions on α and q is that $g(x)$ can be realized only by one element that is $Tr(\alpha)$. However, the same Algorithm 2 chapter 4, given in GH-PKC, can be used to compute $Tr(\alpha^m)$, given $Tr(\alpha)$ and m, for the special case as well. Now we discuss the cryptographic protocols for the special case.

5.3.7.1 GH-DH Type Key Exchange : Special Case $k = 3$, p-arbitrary, $q = p^2$

1. ***System Public parameters.*** Let, $G =< \alpha > \subset \mathbb{F}_{q^3}^*$, p-arbitrary, $q = p^2$, $ord(\alpha) = \ell|(p^2 - p + 1)$, and $g(x) = x^3 - ax^2 + bx - 1$, be the characteristic polynomial of α.

2. Both **A** and **B** choose random $1 < e < \mathbb{Z}_\ell$ and $1 < r < \mathbb{Z}_\ell$ respectively as their static private keys.

3. **A** computes her static public key $P_A = s_e \in \mathbb{F}_q$, by applying Algorithm 2 chapter 4 with inputs e and a, b.

4. Similarly, **B** computes his static public key $P_B = s_r \in \mathbb{F}_q$, by applying Algorithm 2 chapter 4 with inputs r and a, b.

5. Both **A** and **B** compute their common key as follows:

   (i) **A** acquires **B**'s public key P_B and constructs $g_r(x) = x^3 - s_r x^2 + s_r^p x - 1$. Then she computes common key by running Algorithm 2 chapter 4 with inputs $P_B = (s_r, s_r^p)$ and her private key e such that,

   $$K = P_{AB} = V_e(s_r, s_r^p) = s_{er}.$$

   (ii) **B** does the similar things using his private key r to compute common key:

   $$K = P_{BA} = V_r(s_e, s_e^p) = s_{re}.$$

5.3.7.2 Modified GH-ElGamal Type Encryption (MXTR-ElGamal): Special Case $k = 3$, p-arbitrary, $q = p^2$

1. ***System Public parameters.*** Let, $G =< \alpha > \subset \mathbb{F}_{q^3}^*$, p-arbitrary, $q = p^2$, $ord(\alpha) = \ell | (p^2 - p + 1)$, and $g(x) = x^3 - ax^2 + bx - 1$, be the characteristic polynomial of α.

2. **B**'s static public key $P_B = s_r$, private key : random $1 < r <\in \mathbb{Z}_\ell$.

3. Assumption: messages $M \in \mathbb{F}_p$.

4. ***Encryption:*** **A** encrypts message M and sends to **B** as follows:

   (i) Chooses a random ephemeral private key $1 < t < \mathbb{Z}_\ell$, computes ephemeral public key s_t, and encryption key $K = V_t(s_r, s_r^p) + V_t(s_r, s_r^p)^p = s_{rt} + s_{rt}^p \in \mathbb{F}_p$ by running Algorithm 2 chapter 4 with inputs a, b, t and s_r, s_r^p.

   (ii) Encrypts message M by computing $c = MK = M(s_{rt} + s_{rt}^p) \in \mathbb{F}_p$.

   (iii) **A** sends ciphertext $C = (s_t, c) \in (\mathbb{F}_q \times \mathbb{F}_p)$ to **B**.

5. ***Decryption:*** **B** decrypts C by computing encryption key K with the ephemeral public key s_t and his private key r : $K = V_r(s_t, s_t^p) + V_r(s_t, s_t^p)^p = s_{tr} + s_{tr}^p$, and, decrypts c to obtain M: $M = cK^{-1} = c(s_{tr} + s_{tr}^p)^{-1}$.

Remark 5.8. The semantic security of MXTR-ElGamal encryption scheme is computationally equivalent to splitting $z \in \mathbb{F}_p$ into two numbers $(X, Y) \in \mathbb{F}_p^2$ such that $z = X + Y$ where $s_{tr} \mapsto X$ and $s_{-tr} \mapsto Y$. Moreover, $tr \in \mathbb{Z}_\ell$ and $\mathbb{Z}_ell$ is chosen large enough so that the brute force attack becomes infeasible. The number of such (X, Y) are equal to $\frac{p-1}{2}$. More details are given in chapter 4 Remark 4.5.

Table 5.6: Comparison of GH ElGamal encryption scheme with the similar ones

	ElGamal	**GH-RSA-Type**	**MXTR-ElGamal**	**GH-ElGamal**
Encryption	$(2+2(9r^{\log 3}+13r-13)\log n)\mathbf{M}$ $+4r^{\log 3}\log n$ **S**	$10\log N\mathbf{M}$	$1+5.2\log n$ **M**	$(10.4\log n)$ **M**
Decryption	$(2+2(9r^{\log 3}+13r-13)\log n)\mathbf{M}$ $+4r^{\log 3}\log n$ **S**	$12\log N$ **M**	$5.2\log n$ **M**	$(10.4\log n)$ **M**
Throughput	m_1	m_1, m_2	m_1	m_1
Comm. overhead in bits	$\|q\|$ with $q=p^r$	$2\|N\|$	$\|q\|$ with $q=p^2$	$2\|q\|$

The comparison is given in the table below: where N is product of two distinct primes.

5.3.7.3 GH-Nyberg-Rueppel Type Signature : Special Case $k = 3$, p-arbitrary, $q = p^2$

In the special case $q = p^2$, the GH-NR type signature given above in subsection **??** namely, (n, s_r, s_{-r}, C) will be changed to (n, s_r, C) as we do not need to include s_{-r} in the signature because $s_{-r} = s_r^p$. So, we get shorter signature by applying special condition for $q = p^2$ above.

5.3.7.4 Lenstra and Verheul-Nyberg-Rueppel Type Signature Algorithm

We would like to recall a work done by Lenstra and Verheul "The XTR Public Key system" in 2000. In their paper they have constructed public key cryptosystem for the parameters $k = 3, q = p^2$. Their system is nothing but the GH-PKC special case we just discussed above. The main contribution of XTR public key system to the literature as we see, is nothing but the Nyberg Rueppel type signature scheme which was not discussed before in the literature. In the light of above discussion we now want to state their algorithm below.

Algorithm 7 to compute $Tr(\alpha^n\alpha^{mh})$ for the given: $Tr(\alpha) = s, (s_{m-1}, s_m, s_{m+1}), n, h \in \mathbb{Z}_\ell$, $m \in \mathbb{Z}$ is unknown.

Require: $n, h, S_m = (s_{m-1}, s_m, s_{m+1}), Tr(\alpha) = s$.
Ensure: s_{n+mh}.

1: Compute $e = n/h \pmod{\ell}$,
2: Use algorithm 2 chapter 4 to compute (s_{e-1}, s_e, s_{e+1}),
3: Based on s and (s_{e-1}, s_e, s_{e+1}) compute;

$$A = D^{-1}\begin{pmatrix} 2s^2-6s^p & 2s^{2p}+3s-s^{p+2} & s^{p+1}-9 \\ 2s^{2p}+3s-s^{p+2} & (s^2-2s^p)^{p+1}-9 & (2s^{2p}+3s-s^{p+2})^p \\ s^{p+1}-9 & (2s^{2p}+3s-s^{p+2})^p & (2s^2-6s^p)^p \end{pmatrix}\begin{pmatrix} s_{e-1} \\ s_e \\ s_{e+1} \end{pmatrix}$$

where, $D = s^{2p+2} + 18s^{p+1} - 4(s^{3p} + s^3) - 27$,
4: Compute $(s_{m-1}, s_m, s_{m+1})A = s_{e+m}$,
5: $C \leftarrow V_h(s_{e+m}, s^p_{e+m}) = s_{h(e+m)} = s_{n+mh}$.
6: **return** (C).

Remark 5.9. The $Tr(\alpha^n\alpha^{mh})$ can be computed in $8\log_2(n/h \pmod{\ell}) + 8\log_2(h) + 34$ multiplications in $\mathbb{F}_p$.

5.3.8 Simultaneous Double Exponentiation for Cubic Extensions

The double exponentiation algorithm for cubic extensions and 4th extension were introduced by Stam and Lenstra [73] and Kpray Karabinal [41]. Let $G =< \alpha >\subset \mathbb{F}_{q^3}$, $q = p^2$, $ord(\alpha) = \ell | (p^2 - p + 1)$ and $s_u = Tr(\alpha^u)$ for $u \in \mathbb{Z}$. Let $0 < a, b, n, m < \ell$ be integers then computing s_{bn+am}, given s_n, s_m, $S_{n,m} = [s_{n-m}, s_{n-2m}]$ and positive integers a, b. Let $u = n, v = m, d = b$ and $e = a$, then we can write $ud + ve = bn + am = constant$. The sum $d + e$ is decreased until $d = e$ in such a way that $ud + ve = constant$ and $d = e$. At the point $ud + vd = d(u + v) = constant$, single exponentiation is applied to compute $s_{d(u+v)}$. The update rules and related algorithm are given below.

Table 5.7: Stam and Lenstra's Rules for Double Exponentiation

Name of Rule	Condition	d	e	u	v	s_u	s_v	s_{u-v}	s_{u-2v}
if $d > e$									
R_1	$d \leq 4e$	e	$d-e$	$u+v$	u	s_{u+v}	s_u	s_v	$s_{(v-u)}$
R_2	$d \equiv 0 \pmod 2$	$d/2$	e	$2u$	v	s_{2u}	s_v	s_{2u-v}	$s_{2(u-v)}$
R_3	e odd	$(d-e)/2$	e	$2u$	$u+v$	s_{2u}	s_{u+v}	s_{u-v}	s_{-2v}
R_4	$e \equiv 0 \pmod 2$	d	$e/2$	$2v$	u	s_{2v}	s_u	s_{2v-u}	$s_{2(v-u)}$
else									
Sub	$e > d$	e	d	v	u	s_v	s_u	s_{v-u}	s_{2v-u}

Algorithm 8 Double Exponentiation for NR-GH-DSA

Require: $(a > 0, b > 0) \in \ell$, s_l, s_k,s_{k-l}, and s_{k-2l}.
Ensure: s_{bk+al}
1: $f_2 \leftarrow 1$, $d \leftarrow b$, $e \leftarrow a$, $u \leftarrow k$, $v \leftarrow l$
2: **while** d and e are both even **do**
3: $d \leftarrow d/2$, $e \leftarrow e/2$, $f_2 \leftarrow 2f_2$
4: **end while**
5: **while** $d \neq e$ **do**
6: Apply the foremost rule in Table 5.7
7: **end while**
8: compute $s_{d(u+v)}$ using algorithm [30, Algorithm 1] with inputs d and s_{u+v}.
9: **return** $(s_{f_2(d(u+v))})$

Remark 5.10. The Algorithm 3 can compute s_{bk+al} in $6\log(max(a,b))$ multiplications in $\mathbb{F}_p$ given s_l, $0 < a, b < \ell$, s_k,s_{k-l}, and s_{k-2l}. Note that Algorithm 3 can be used for single exponentiation for given s_1 and $u \in \ell$. In this case, single exponentiation takes $5.2\log u$ multiplications in $\mathbb{F}_p$.

Now we describe Lenstra and Verheul's Modified XTR Nyberg-Rueppel Type Digital Signature Scheme.

1. ***System Public parameters.*** Let, $G =< \alpha >\subset \mathbb{F}^*_{q^3}$, p-arbitrary, $q = p^2$, $ord(\alpha) = \ell|(p^2 - p + 1)$, and $g(x) = x^3 - ax^2 + bx - 1$, be the characteristic polynomial of α, and $H : G \to \mathbb{Z}_\ell$ valued hash function.
2. **A**'s public key $P_A = (s_m) := (s_{m-1}, s_m, s_{m+1})$, private key : random $1 < m <\in \mathbb{Z}_\ell$.
3. **B**'s public key $P_B = (s_e) := (s_{e-1}, s_e, s_{e+1})$, private key : random $1 < e <\in \mathbb{Z}_\ell$.
4. Assumption: message $M \in \mathbb{F}_p$ contains agreed upon redundancy.
5. ***Signature***: **A** signs the message M as follows:
   (i) Chooses a random ephemeral private key $1 < r < \mathbb{Z}_\ell$ and computes ephemeral public key s_r.
   (ii) Based on mask s_r and **B**'s static public key s_e determines symmetric key $K = s_{re} + s^p_{re}$.
   (iii) Computes $h = H(M) \mod \ell$ and encrypts message $M \longrightarrow C : C = MK$ using MXTR-ElGamal encryption scheme.
   (iv) Computes $n = (r - mh) \pmod \ell$.
   (v) **A**'s resulting signature on M is (n, s_r, C).

5. ***Verification***: **B** verifies **A**'s signature as follows:
   (i) Computes Checks whether $1 \leq n < \ell$. If not failure.

(ii) Computes encryption key $K = s_{er} + s_{er}^{p}$ and decrypts $C \mapsto M$. If M contains agreed upon redundancy then proceed.

(iii) Computes $h = H(M) \mod \ell$,

(iv) Running Algorithm 2 chapter 4 with inputs h and s_m and then Algorithm 8 above with inputs $n = a + b, (s_{hm-1}, s_h m, s_{hm+1}), s$, computes $s_{n+mh} = Tr(\alpha^n \alpha^{mh})$.

(v) If $s_{n+mh} = s_r$, **B** accepts.

The comparison of NR-DSA based on cubic extension is given in the table below:

Table 5.8: Comparison of GH-NR-DSA with the similar DSAs

	GH-DSA	**MXTR-NR-DSA**	**EC-DSA**	**GH-NR-DSA**
Double exponentiation	$(s_{c(h-dl)}, s_{-c(h-dl)})$	(s_{s+hk})	$(u_1 + u_2 d)P$	$(s_{n+eh}, s_{-(n+eh)})$
Cost	$16 \log Q$ **M** + $16 \log Q$ **S**	$6 \log Q$ **M**	$7 \log Q$ **M** + $3.7 \log Q$ **S**	$12 \log Q$ **M**
Throughput	m_1, m_2	m_1	m_1	m_1
Message recovery	No	Yes	No	Yes
Comm. overhead in bits	$2\|q\| + \|H\|$	$\|q\| + \|H\|$	$\|p\| + \|H\|$	$2\|q\| + \|H\|$

5.3.9 GG Public Key Cryptosystem

We will introduce two cases given by K. Giuliani and G. Gong in 2003 [27, 28] and fill the gap in this PKC by introducing GG-ElGamal type encryption scheme and Nyberg-Rueppel type digital signatures.
Case 1: $k = 5$, p-arbitrary, $q = p^u,\ u \geq 1 \in \mathbb{Z}$.
Case 2: $k = 5$, p-arbitrary, $q = p^2$.

5.3.9.1 GG PKC : Case $k = 5$, p-arbitrary, $q = p^u$

Let, $G =< \alpha > \cong \mathbb{Z}_\ell \subset \mathbb{F}_{q^5}^*$, $ord(\alpha) = \ell | (q^4 + q^3 + q^2 + q + 1)$, note that $\Phi_5(q) = (q^4 + q^3 + q^2 + q + 1)$ is the $5th$ cyclotomic polynomial evaluated at q then, let $g(x)$ be the characteristic polynomial of α:

$$g(x) = \prod_{i=0}^{4} (x - \alpha^{q^i}) = x^5 - ax^4 + bx^3 - cx^2 + dx - 1.$$

One can check that,

$$a = Tr(\alpha),\ d = Tr(\alpha^{-1}), b = \sum_{0 \leq i < j \leq 4} \alpha^{q^i + q^j},\ c = \sum_{0 \leq i < j \leq 4} \alpha^{-q^i - q^j}.$$

For all integers m, let $g_m(x)$ be the characteristic polynomial of α^m:

$$g_m(x) = \prod_{i=0}^{4} (x - (\alpha^m)^{q^i}) = x^5 - a_m x^4 + b_m x^3 - c_m x^2 + d_m x - 1.$$

One can check that,

$$a_m = Tr(\alpha^m),\ d_m = a_{-m} = Tr(\alpha^{-m}),$$
$$b_m = \sum_{0\le i<j\le 4} \alpha^{mq^i+mq^j},\ c_m = b_{-m} = \sum_{0\le i<j\le 4} \alpha^{-mq^i-mq^j}.$$

So let, $V_m(a,b,c,d) = a_m = s_m$, and $\widehat{V}_m(a,b,c,d) = b_m = \hat{s}_m$. In particular, $s_0 = 5, s_1 = a, s_2 = a^2 - 2b, s_3 = a^3 - 3ab + 3c, s_4 = s_2^2 - 2b^2 - 4d + 4c$. Then we have,

$$g_m(x) = x^5 - s_m x^4 + \hat{s}_m x^3 - \hat{s}_{-m} x^2 + s_{-m} x - 1.$$

Now if we replace α above by $\beta = \alpha^r$ for any integer r, then we have the characteristic polynomial for β^m in the form,

$$h_m(x) = \prod_{i=0}^{4} (x - (\beta^m)^{q^i}) = \prod_{i=0}^{4} (x - (\alpha^{rm})^{q^i}) = g_{rm}(x).$$

Therefore,

$$V_m(s_r, \hat{s}_r, \hat{s}_{-r}, s_{-r}) = V_{mr}(a,b,c,d) = s_{mr},$$
$$\widehat{V}_m(s_r, \hat{s}_r, \hat{s}_{-r}, s_{-r}) = \widehat{V}_{mr}(a,b,c,d) = \hat{s}_{mr}.$$

Properties: For all integers m and n following are the properties:

(i) $s_{2n} = s_n^2 - 2\hat{s}_n$, and
$\hat{s}_{2n} = \hat{s}_n^2 + 2s_{-n} - 2s_n\hat{s}_{-n}$.

(ii) $s_{3n} = s_n^3 - 3s_n\hat{s}_n + 3\hat{s}_{-n}$.

(iii) $\hat{s}_{3n} = \hat{s}_n^3 - 3s_n^3\hat{s}_n - 3s_n\hat{s}_n\hat{s}_{-n} + 3s_n^2 s_n + 3\hat{s}_{-2n} - 3s_n$.

(iv) $s_{n+m} = s_n s_m - s_{n-m}\hat{s}_m + s_{n-2m}\hat{s}_{-m} - s_{n-3m}s_{-m} + s_{n-4m}$.

(v) $\hat{s}_n\hat{s}_m - s_{-m}\hat{s}_{n-m} + 3\hat{s}_{n+m} = s_n s_m s_{n+m} s_{n-2m} s_{n-m} + s_{2n-3m} - s_{n+2m}s_n - s_{2n+m}s_m + s_{n+m}^2$.

(vi) $s_{n+2m} = s_{n+m}\hat{s}_m - s_n\hat{s}_m + s_{n-m}\hat{s}_{-m} - s_{n-2m}s_{-m} + s_{n-3m}$.

Using above properties the algorithm introduced by K. Giuliani and G. Gong in [27, 28] to compute nth terms of sequences $\{s_u, \hat{s}_u\}$ is as follows:

Algorithm 9 to compute $s_n = Tr(\alpha^n)$ and $\hat{s}_n = \sum_{0 \le i < j \le 4} \alpha^{nq^i + nq^j}$ using triple-add-subtract

Require: $n \in \mathbb{Z}\ell : n = \sum_{j=0}^{l} n_j 3^j \in \mathbb{Z}^+, n_j \in \{-1, 0, 1\}$ with $n_l \neq 0$, and initial conditions $(s_{-1}, 5, s_1, s_2, s_3), (\hat{s}_{-1}, 5, \hat{s}_1, \hat{s}_2, \hat{s}_3)$.

Ensure: $(s_{n-2}, s_{n-1}, s_n, s_{n+1}, s_{n+2}), (\hat{s}_{n-2}, \hat{s}_{n-1}, \hat{s}_n, \hat{s}_{n+1}, \hat{s}_{n+2})$

1: $u \leftarrow 1, S \leftarrow (s_{-1}, 5, s_1, s_2, s_3), \hat{S} \leftarrow (\hat{s}_{-1}, 5, \hat{s}_1, \hat{s}_2, \hat{s}_3)$
2: **for** $i \leftarrow 0$ ***to*** l **do**
3: **if** $n_i = -1$ **then**
4: $S \leftarrow (s_{3u-3}, s_{3u-2}, s_{3u-1}, s_{3u}, s_{3u+1})$,
5: $\hat{S} \leftarrow (\hat{s}_{3u-3}, \hat{s}_{3u-2}, \hat{s}_{3u-1}, \hat{s}_{3u}, \hat{s}_{3u+1})$.
6: **else if** $n_i = 0$ **then**
7: $S \leftarrow (s_{3u-2}, s_{3u-1}, s_{3u}, s_{3u+1}, s_{3u+2})$,
8: $\hat{S} \leftarrow (\hat{s}_{3u-2}, \hat{s}_{3u-1}, \hat{s}_{3u}, \hat{s}_{3u+1}, \hat{s}_{3u+2})$.
9: **else**
10: $S \leftarrow (s_{3u-1}, s_{3u}, s_{3u+1}, s_{3u+2}, s_{3u+3})$,
11: $\hat{S} \leftarrow (\hat{s}_{3u-1}, \hat{s}_{3u}, \hat{s}_{3u+1}, \hat{s}_{3u+2}, \hat{s}_{3u+3})$.
12: **end if**
13: $u = 3u + n_i$
14: **end for**
15: **return** $((s_{n-2}, s_{n-1}, s_n, s_{n+1}, s_{n+2}), (\hat{s}_{n-2}, \hat{s}_{n-1}, \hat{s}_n, \hat{s}_{n+1}, \hat{s}_{n+2})) = (S_n, \hat{S}_n)$.

The average cost of computations is approximately $108.5 \log n$ multiplications, $13 \log n$ scalar multiplications and $280.1 \log n$ additions.

For this case it is not difficult to give the cryptographic protocols to have a trace based cryptosystem. However, it is not interesting from the implementation point of view and therefore we leave this and discuss the next special case together with cryptographic protocols.

5.3.9.2 GG PKC : Special Case, $k = 5$, p-arbitrary, $q = p^2$

K. Giuliani and G. Gong introduced special case for their cryptosystem in 2004 [27] . Let $G =< \alpha >\subset \mathbb{F}_{q^5}^*$, p-arbitrary, $q = p^2$, $ord(\alpha) = \ell | (p^4 - p^3 + p^2 - p + 1) | (q^4 + q^3 + q^2 + q + 1)$. We keep the notations above and note that in this special case for any integer m we have,

$$
\begin{aligned}
&Tr(\alpha^{-m}) = Tr(\alpha^m)^p = s_m^p, \\
&\sum_{0 \le i < j \le 4} \alpha^{-mq^i - mq^j} = Tr(\alpha^{-m(p+1)} + \alpha^{-m(p^2+1)}) = Tr(\alpha^{m(p+1)} + \alpha^{m(p^2+1)})^p \\
&\qquad = \hat{s}_m^p.
\end{aligned}
$$

Thus, for any integer m the the characteristic polynomial $g_m(x)$ of α^m is

$$g_m(x) = \prod_{i=0}^{4}(x - \alpha^{mq^i}) = x^5 - s_m x^4 + \hat{s}_m x^3 - \hat{s}_m^p x^2 + s_m^p x - 1.$$

Properties: For all integers m and n following properties are proved by K. Giuliani and G. Gong in [27, 28] and we list here. Let $\alpha \in \mathbb{F}_{q^5}$, having order $\ell|(p^4 - p^3 + p^2 - p + 1)$, and characteristic polynomial $g(x) = x^5 - ax^4 + bx^3 - cx^2 + dx - 1$, then,

(i) $s_m = Tr(\alpha^m)$, $s_{-m} = s_m^p$ and,
$\hat{s}_m = Tr(\alpha^{m(p+1)} + \alpha^{m(p^2+1)})$, $\hat{s}_{-m} = \hat{s}_m^p$.

(ii) $V_m(s_n, \hat{s}_n, \hat{s}_n^p, s_n^p) = s_{mn}$, and
$\widehat{V}_m(s_n, \hat{s}_n, \hat{s}_n^p, s_n^p) = \hat{s}_{mn}$.

(iii) $s_{2n} = s_n^2 - 2\hat{s}_n$, and
$\hat{s}_{2n} = \hat{s}_n^2 + 2s_{-n} - 2s_n\hat{s}_n^p$.

(iv) $s_{3n} = s_n^3 - 3s_n\hat{s}_n + 3\hat{s}_n^p$.

(v) $\hat{s}_{3n} = \hat{s}_n^3 - 3s_n^3\hat{s}_n - 3s_n\hat{s}_n\hat{s}_n^p + 3s_n^2 s_n + 3\hat{s}_{2n}^p - 3s_n$.

(vi) $s_{n+m} = s_n s_m - s_{n-m}\hat{s}_m + s_{n-2m}\hat{s}_m^p - s_{n-3m}s_m^p + s_{n-4m}$.

(vii) $\hat{s}_n\hat{s}_m - s_{-m}\hat{s}_{n-m} + 3\hat{s}_{n+m} = s_n s_m s_{n+m} s_{n-2m} s_{n-m} + s_{2n-3m} - s_{n+2m}s_n - s_{2n+m}s_m + s_{n+m}^2$.

(viii) $s_{n+2m} = s_{n+m}\hat{s}_m - s_n\hat{s}_m + s_{n-m}\hat{s}_m^p - s_{n-2m}s_m^p + s_{n-3m}$.

Remark 5.11. Algorithm 9 can also compute *nth* term just by replacing $s_{-1} = s_1^p$ and $\hat{s}_{-1} = \hat{s}_1^p$ and accordingly in subsequent computations. Actually GG Algorithm computes for a given m and $(s_1, \hat{s}_1)$; the terms
$(s_{m-4}, s_{m-3}, s_{m-2}, s_{m-1}, s_{m+})$ and $(\hat{s}_{m-4}, \hat{s}_{m-3}, \hat{s}_{m-2}, \hat{s}_{m-1}, \hat{s}_m)$.

With this setup, now we discuss the cryptographic protocols and add ElGamal type encryption scheme and Nyberg Rueppel type digital signature scheme to GG-PKC.

5.3.9.3 GG-DH Type Key Exchange : Special Case

1. **System Public Parameters:** Let, $G =< \alpha > \subset \mathbb{F}_{q^5}^*$, p-arbitrary, $q = p^2$, $ord(\alpha) = \ell|(p^4 - p^3 + p^2 - p + 1)$, and $g(x) = x^5 - ax^4 + bx^3 - cx^2 + dx - 1$, be the characteristic polynomial of α.

2. **A** computes her static public key $P_A = (s_e, \hat{s}_e)$, by running Algorithm 9 with inputs a, b, c, d and her static private key, a random $1 < e < \mathbb{Z}_\ell$.

3. Similarly, **B** computes his static public key $P_B = (s_r, \hat{s}_r)$, by running Algorithm 9 with inputs a, b, c, d and his static private key, a random $1 < r < \mathbb{Z}_\ell$.

4. Both A and B compute common key $K = P_{AB} = P_{BA} = (s_{er}, \hat{s}_{er})$ as follows:

   (i) **A** acquires **B**'s public key P_B and constructs

   $$g_t(x) = x^5 - s_r x^4 + \hat{s}_r x^3 - \hat{s}_r^p x^2 + s_r^p x - 1.$$

   Then she computes common key by running Algorithm 9 with inputs $P_B = (s_r, \hat{s}_r, \hat{s}_r^p, s_r^p)$ and her private key e such that,

   $$K = P_{AB} = (V_e(s_r, \hat{s}_r, \hat{s}_r^p, s_r^p), \widehat{V}_e(s_r, \hat{s}_r, \hat{s}_r^p, s_r^p)) = (s_{er}, \hat{s}_{er}).$$

   (ii) **B** does the similar things using his private key r to compute the common key:

   $$K = P_{BA} = (V_r(s_e, \hat{s}_e, \hat{s}_e^p, s_e^p), \widehat{V}_r(s_e, \hat{s}_e, \hat{s}_e^p, s_e^p)) = (s_{re}, \hat{s}_{re}).$$

5.3.9.4 GG-ElGamal Type Encryption Scheme : Special Case

For the sake of completeness of GG-PKC, the ElGamal type encryption scheme is being introduced for the first time . Let **A** wants to send a message M to **B**, both proceed as follows:

1. ***System Public Parameters:*** Let, $G =< \alpha > \subset \mathbb{F}_{q^5}^*$, p-arbitrary, $q = p^2$, $ord(\alpha) = \ell | (p^4 - p^3 + p^2 - p + 1)$, and $g(x) = x^5 - ax^4 + bx^3 - cx^2 + dx - 1$, be the characteristic polynomial of α.

2. ***Public Keys:*** Following are the public keys of **A** and **B**:

   (i) **A** selects random $0 < e < \mathbb{Z}_\ell$, and computes her static public key $P_A = (s_e, \hat{s}_e)$ by running Algorithm 9 with inputs a, b, c, d and e.

   (ii) Similarly, **B** selects random $0 < r < \mathbb{Z}_\ell$, and computes static public key $P_B = (s_r, \hat{s}_r)$ by running Algorithm 9 with inputs a, b, c, d and r.

3. ***Private Keys:*** e and r are the static private keys of **A** and **B** respectively.

4. ***Encryption:*** **A** encrypts message M and sends to **B** as follows:

   (i) **A** randomly selects ephemeral private key $0 < t < \mathbb{Z}_\ell$ and computes static public key $(s_t, \hat{s}_t)$ by running Algorithm 9 with inputs a, b, c, d and t.

   (ii) **A** acquires **B**'s public key $P_B = (s_r, \hat{s}_r)$ and computes encryption key $K = (s_{tr} + \hat{s}_{tr}) = (V_t(s_r, \hat{s}_r, \hat{s}_r^p, s_r^p) + \widehat{V}_t(s_r, \hat{s}_r, \hat{s}_r^p, s_r^p))$ by running Algorithm 9 with inputs $(s_r, \hat{s}_r, \hat{s}_r^p, s_r^p)$ and t.

   (iii) **A** computes, $C = MK$ and sends the ciphertext $C = ((s_t, \hat{s}_t), C)$ to **B**.

5. ***Decryption:*** **B** recovers the message M as follows:

   (i) Using ephemeral public key $(s_t, \hat{s}_t)$ and his static private key r, computes encryption key $K = (s_{rt}+\hat{s}_{rt}) = (V_r(s_t, \hat{s}_t, \hat{s}_t^p, s_t^p)+\widehat{V}_r(s_t, \hat{s}_t, \hat{s}_t^p, s_t^p)$ as above.

   (ii) Computes, M such that $M = CK^{-1}$.

Remark 5.12. The semantic security of GG-ElGamal type encryption is computationally equivalent to splitting $z \in \mathbb{F}_q$ into two numbers $(X, Y) \in \mathbb{F}_q^2$ such that $z = X + Y$ where $s_{tr} \mapsto X$ and $s_{-tr} \mapsto Y$. Moreover, $tr \in \mathbb{Z}_\ell$ and $\mathbb{Z}_ell$ is chosen large enough so that the brute force attack becomes infeasible. The number of such (X, Y) are equal to $\frac{q-1}{2}$ as discussed in chapter 4, Remark 4.5.

The analysis of GG-ElGamal type encryption is given in the Table 5.9 below.

Table 5.9: GG-ElGamal type Encryption scheme Analysis

Process	**Cost**
Encryption	$\approx 120 \log p\mathbf{M}$
Decryption	$\approx 120 \log p\mathbf{M}$
Comm. Overhead	2 elements over $\mathbb{F}_q$
Public Key size	2 elements over $\mathbb{F}_q$

5.3.9.5 GG-NR Type Digital Signature Algorithm : Special Case

After introduction of GG-ElGamal type encryption scheme now we introduce Nyber-Rueppel type digital signature scheme based on GG-ElGamal type encryption scheme. Let **A** wants to send signed message $M = (m_1, m_2)$ containing agreed upon redundancy to **B** and **B** verifies it. To do this, **A** and **B** proceed as follows:

1. ***System Public Parameters:*** Let, $G =< \alpha > \subset \mathbb{F}_{q^5}^*$, p-arbitrary, $q = p^2$, $ord(\alpha) = \ell | (p^4 - p^3 + p^2 - p + 1)$, and $g(x) = x^5 - ax^4 + bx^3 - cx^2 + dx - 1$, be the characteristic polynomial of α and $H : G \to \mathbb{Z}_\ell$ valued hash function.

2. ***Public Keys:***

   (i) **A** chooses random $0 < e < \mathbb{Z}_\ell$, and computes public key $P_{\mathbf{A}} = (s_e, \hat{s}_e)$ by running Algorithm 9 with inputs a, b, c, d, and e.

   (ii) **B** chooses random $0 < r < \mathbb{Z}_\ell$, and computes public key $P_{\mathbf{B}} = (s_r, \hat{s}_r)$ by running Algorithm 9 with inputs a, b, c, d, and r.

3. ***Private Keys:*** The private keys of **A** and **B** are e and r, respectively.

4. ***Signature***: **A** signs the message M as follows:

(i) **A** chooses random ephemeral private key $0 < t < \mathbb{Z}_\ell$, and computes ephemeral public key $(V_t(a,b,c,d), \widehat{V}_t(a,b,c,d)) = (s_t, \hat{s}_t)$, and encryption key $K = (V_t(s_r, \hat{s}_r, \hat{s}_r^p, s_r^p) + \widehat{V}_t(s_r, \hat{s}_r, \hat{s}_r^p, s_r^p)) = (s_{tr} + \hat{s}_{tr})$.

(ii) Computes hash value of M : $h = H(M) \mod \ell$.

(iii) **A** obtains ciphertext C by encrypting message M : $C = MK = M(s_{tr} + \hat{s}_{tr})$.

(iv) **A** computes $n = t - he \mod \ell$.

(v) **A** sends the signature $(n, (s_t, \hat{s}_t), C)$ to **B**.

5. ***Verification***: **B** recovers message M and verifies **A**'s signature as follows:

(i) Checks $0 \leq n < \ell$, if not failure.

(ii) Computes encryption key $K = (V_r(s_t, \hat{s}_t, \hat{s}_t^p, s_t^p) + \widehat{V}_r(s_t, \hat{s}_t, \hat{s}_t^p, s_t^p)) = (s_{rt} + \hat{s}_{rt})$ and decrypts C to M : $M = CK^{-1} = C(s_{rt} + \hat{s}_{rt})^{-1}$. If M does not contain agreed upon redundancy then failure.

(iii) Computes $h = H(M) \mod \ell$, and $(s_{he}, \hat{s}_{he}) = (V_h(s_e, \hat{s}_e, \hat{s}_e^p, s_e^p), \hat{V}_h(s_e, \hat{s}_e, \hat{s}_e^p, s_e^p))$ from h and **A**'s public key $(s_e, \hat{s}_e)$. Note that Algorithm 9 computes vector $S_{he} = (s_{he-4}, \cdots, s_{he})$.

(iv) Computes (s_{n+he}) by running Algorithm 5 with inputs $k = 5$, vector S_{he}, and n.

(v) **B** checks whether $(s_{n+he}, \hat{s}_{n+he})$ is equal to $(s_t, \hat{s}_t)$, if yes accepts.

The analysis of GG-NR-DSA is given in the Table 5.10 below.

Table 5.10: GG-NR-DSA Analysis

Process	**Cost**
Signature Generation	$\approx 120 \log p$ **M**
Signature Verification	$\approx 240 \log p$ **M**
Comm. Overhead	3 elements over $\mathbb{F}_q$
Public Key size	2 elements over $\mathbb{F}_q$

5.3.10 Speeding Up GG-PKC

Here we introduce double exponentiation algorithm adopted from [41]. The double exponentiation can be transformed to single exponentiation. This method speeds up the double and single exponentiations.

Lemma 5.1. *For all u, v, let*

$$s_{u+2v} = s_{u+v}s_v - s_u\hat{s}_v + s_{u-v}\hat{s}_v^p - s_{u-2v}s_v^p + s_{u-3v}, \tag{5.4}$$

and

$$s_{u+v} = s_u s_v - s_{u-v}\hat{s}_v + s_{u-2v}\hat{s}_v^p - s_{u-3v}s_v^p + s_{u-4v}. \tag{5.5}$$

(i) $s_{3u-v} = s_{2u+v}^p - s_{u+v}^p s_u^p + s_v^p \hat{s}_u^p - s_{u-v}\hat{s}_u + s_{2u-v}s_u$,

(ii) $s_{3u-2v} = s_{2(u-v)}s_u - s_{u-2v}\hat{s}_u + s_{2v}^p \hat{s}_u^p - s_{u+2v}^p s_u^p + s_{2(u+v)}^p$,

(iii) $s_{u+3v} = s_{u+2v}s_v - s_{u+v}\hat{s}_v + s_u \hat{s}_v^p - s_{u-v}s_v^p + s_{u-2v}$,

(iv) $s_{u+4v} = s_{u+3v}s_v - s_{u+2v}\hat{s}_v + s_{u+v}\hat{s}_v^p - s_u s_v^p + s_{u-v}$,

(v) $s_{2u-v} = s_{u-v}s_u - s_v^p \hat{s}_u + s_{u+v}^p \hat{s}_u^p - s_{2u+v}^p s_u^p + s_{3u+v}^p$,

(vi) $s_{2u-3v} = s_{2(u+v)} - s_{2u+v}s_v + s_{2u}\hat{s}_v - s_{2u-v}\hat{s}_v^p + s_{2(u-v)}s_v^p$,

(vii) $s_{3u+v} = s_{2u-v}^p - s_{u-v}^p s_u^p + s_v \hat{s}_u^p - s_{u+v}\hat{s}_u + s_{(2u+v)}s_u$,

(viii) $s_{2u+v} = s_{u+v}s_u - s_v\hat{s}_u + s_{u-v}^p \hat{s}_u^p - s_{2u-v}^p s_u^p + s_{3u-v}^p$.

Table 5.11: Substitution values for u, v and derived formulas

Substitution for u	**Substitution for v**	**Equation**	**Formulas**
$-v$	$-u$	5.4	s_{3u-v}, (Lemma 5.1(i))
$u-2v$	u	5.4	s_{3u-2v}, (Lemma 5.1(ii))
$u+v$	v	5.4	s_{u+3v}, (Lemma 5.1(iii))
$u+2v$	v	5.4	s_{u+4v}, (Lemma 5.1(iv))
$-v$	u	5.4	s_{2u-v}, (Lemma 5.1(v))
$2u+v$	v	5.5	s_{2u-3v}, (Lemma 5.1(vi))
v	$-u$	5.4	s_{3u+v}, (Lemma 5.1(vii))
v	u		s_{2u+v}, (Lemma 5.1(viii))

Corollary 5.2. *Give s_u, s_v, s_{u-v}, s_{u-2v}, s_{u-3v}, s_{u-4v}, $\hat{s}_u$, $\hat{s}_v$, $\hat{s}_{u-v}$ and $\hat{s}_{u+v}$ following holds;*

(i) Computing $s_{u+2v} = s_{u+v}s_v - s_u\hat{s}_v + s_{u-v}\hat{s}_v^p - s_{u-2v}s_v^p + s_{u-3v}$, takes $4M$ over $\mathbb{F}_{p^2}$.

(ii) Computing $s_{3u-v} = s_{2u}s_{u-v} - s_{u+v}\hat{s}_{u-v} + s_{2v}\hat{s}_{u-v}^p - s_{u-3v}^p s_{u-v}^p + s_{2(u-2v)}^p$, takes $4M + 3S$ over $\mathbb{F}_{p^2}$.

(iii) Computing $s_{3u-2v} = s_{2(u-v)}s_u - s_{u-2v}\hat{s}_u + s_{2v}^p \hat{s}_u^p - s_{u+2v}^p s_u^p + s_{2(u+v)}^p$, takes $4M + 3S$ over $\mathbb{F}_{p^2}$.

(iv) Computing $s_{u+3v} = s_{u+2v}s_v - s_{u+v}\hat{s}_v + s_u\hat{s}_v^p - s_{u-v}s_v^p + s_{u-2v}$, takes $4M$ over $\mathbb{F}_{p^2}$.

(v) Computing $s_{u+4v} = s_{u+3v}s_v - s_{u+2v}\hat{s}_v + s_{u+v}\hat{s}_v^p - s_u s_v^p + s_{u-v}$, takes $4M$ over $\mathbb{F}_{p^2}$.

(vi) Computing $s_{2u-v} = s_{u-v}s_u - s_v^p \hat{s}_u + s_{u+v}^p \hat{s}_u^p - s_{2u+v}^p s_u^p + s_{3u+v}^p$, takes $7M + 1S + 1I$ over $\mathbb{F}_{p^2}$.

(vii) Computing $s_{2u-3v} = s_{u-2v}s_{u-v} - s_v^p \hat{s}_{u-v} + s_u^p \hat{s}_{u-v}^p - s_{2u-v}^p s_{u-v}^p + s_{3u-2v}^p$, takes $4M$ over $\mathbb{F}_{p^2}$.

(viii) *Computing* $s_{3u+v} = s_{2u}s_{u+v} - s_{u-v}\hat{s}_{u+v} + s_{2v}^p\hat{s}_{u+v}^p - s_{u+3v}^p s_{u+v}^p + s_{2(u+2v)}$, *takes* $4M + 3S$ *over* $\mathbb{F}_{p^2}$.

(ix) *Computing* $s_{2u+v} = s_{u+v}s_u - s_v\hat{s}_u + s_{u-v}^p\hat{s}_u^p - s_{2u-v}^p s_u^p + s_{3u-v}^p$, *takes* $4M$ *over* $\mathbb{F}_{p^2}$.

Fifth-degree Double Exponentiation:

The double exponentiation was introduced by P.L. Montgomery, Stam and Lenstra, and Koray Karabina in [57, 73, 42] for second-degree, third-degree and fourth-degree recursive relation, respectively. Here we adopt double exponentiation techniques for fifth-degree recursive relation related to the irreducible polynomial $f(x) = x^5 - a_1x^4 + a_2x^3 - a_2^px^2 + a_1^px - 1$ over $\mathbb{F}_{p^2}$. Let s_{bk+al} represent double exponentiation, where $0 < a, b, k, l < \ell$ are positive integers. Let $u = k$, $v = l$, $d = b$, and $e = a$, from which one can compute $bk + al = ud + ve = \chi$. The idea is to vary $d, e, u,$ and v in such a way that $ud + ve = \chi$ holds, and $(d + e)$ decreases until $d = e$. During this variation the values of $u, v, s_{u+v}, s_{u-v}, s_{u-2v}$, and s_{u-3v} are updated according to new values of u and v given in Table 5.12. When $d = e$ that is $ud + ev = d(u + v) = \chi$, single exponentiation Algorithm 9 is applied to compute $s_{d(u+v)}$.

Table 5.12: The Rules for Double Exponentiation for $\mathbb{F}_{q^5}, q = p^2$

Rule	**Condition**	**d**	**e**	u	v	s_{u+v}	s_{u-v}	s_{u-2v}	s_{u-3v}
if $d > e$									
R_1	$d \leq 4e$	e	$d-e$	$u+v$	u	s_{2u+v}	s_v	s_{u-v}^p	$s_{(2u-v)}^p$
R_2	$d \equiv 0 \pmod 2$	$d/2$	e	$2u$	v	s_{2u+v}	s_{2u-v}	$s_{2(u-v)}$	s_{2u-3v}
R_3	e *odd*	$(d-e)/2$	e	$2u$	$u+v$	s_{3u+v}	s_{u-v}	s_{2v}^p	s_{u+3v}^p
R_4	$e \equiv 0 \pmod 2$	$e/2$	d	$2v$	u	s_{u+2v}	s_{u-2v}^p	$s_{2(u-v)}^p$	s_{3u-2v}^p
else									
Sub	$e > d$	e	d	v	u	s_v	s_u	s_{v-u}	s_{2v-u}

Table 5.13: Analysis of Rules for Double Exponentiation for $\mathbb{F}_{q^5}, q = p^2$

Rule	**Condition**	**Cost**	**Reduction Factor for** $(d+e)$	***Average Usage Exp [42]***
R_1	$d \leq 4e$	$4M + 1S$	$\geq 5/4, < 2$	0.61
R_2	$d \equiv 0 \pmod 2$	$8M + 1S$	2	0.175
R_3	e *odd*	$8M + 1S$	$\geq 5/3, < 2$	0.129
R_4	$e \equiv 0 \pmod 2$	$8M + 1S$	$> 1, < 10/9$	0.085

Note that the average usage of rules R_1, R_2, R_3, and R_4 does not depend upon the group order ℓ.

Algorithm 10 Double Exponentiation for Fifth Degree Recursive Relation

Require: $(a > 0, b > 0) \in \ell$, s_l, s_k, s_{k+l}, s_{k-l}, s_{k-2l}, s_{k-3l}, $\hat{s}_u$, $\hat{s}_v$, $\hat{s}_{u+v}$, and $\hat{s}_{u-v}$.
Ensure: s_{bk+al}

```
f_2 ← 1, d ← b, e ← a, u ← k, v ← l
while d and e are both even do
    d ← d/2, e ← e/2, f_2 ← 2f_2
end while
while d ≠ e do
    apply the foremost rule in Table 5.12
end while
Compute s_{d(u+v)} using [30, Algorithm 1] with inputs d and s_{u+v}.
Compute s_{f_2(d(u+v))} using relation s_{2u} = s_u^2 − 2ŝ_u.
return (s_{f_2(d(u+v))})
```

Analysis: The complexity analysis of the Algorithm 10 is similar to the one given in [42]. If rule R_4 is never required then it is clear from Table 5.13 that of second while loop never exceeds

$$max(4.8\log_{5/4}(\acute{a}+\acute{b})M, 8.8\log_2(\acute{a}+\acute{b})M, 8.8\log_{5/3}(\acute{a}+\acute{b})M) \approx 14.4\log_2(\acute{a},\acute{b}),$$

where $\acute{a} = a/gcd(a,b)$, and $\acute{b} = b/gcd(a,b)$. Without loss of generality, we assumed that 1 squaring is equal to 0.8 multiplications. Now, let rule R_4 is applied for $i > 0$ times successively, with $(d_1, e_1) \leftarrow (d, e)$. This implies $d_1 > 4e_1$, $d_1 \not\equiv 0 \pmod 2$, $d_1 \not\equiv e \pmod 2$, and $e_1 \equiv \pmod 2$. Let (d_2, e_2) be the new value of (d, e) after $i - th$ applications of rule R_4, then $d_2 = d_1$ and $e_1 = 2^i e_2$. At this stage let R_2 becomes applicable and thereafter R_3, possibly. Let R_2 is applied $j > 0$ times and then rule R_1 (i.e. $d \leq 4e$) qualifies or $j \leq i$ and new value of (d, e) becomes (d_3, e_3). This implies $e_2 = e_3$ and $d_3 \leq d_2/2^i$. If $d_3 \leq 4e_3$ then,

$$\frac{(d_1+e_1)}{(d_3+e_3)} \geq \frac{5e_1}{5e_3} = 2^i.$$

If $j = i$ then,

$$\frac{(d_1+e_1)}{(d_3+e_3)} \geq \frac{d_2+e_2 2^i}{d_2/2^j+e_2} = 2^i.$$

In both cases, the $(d+e)$ value is reduced at least by a factor of 2^i at the cost of $8.8(i+j)M \leq 17.6iM$. The experimental results in Table 5.13 shows that the usage of rule R_4 is only 8.5% as compared with 61% of R_1. Therefore, most of the computational cost in second while loop is contributed due to execution of R_1 which never exceeds $14.4\log_2(\acute{a}+\acute{b})M$ and upper bound for the total cost is $14.4\log(a+b)M$.

Remark 5.13. Note that the experiments show that number of iterations for second while loop in Algorithm 10 are $1.45\log(a+b)$ as given in [42] and the number of multiplications required to compute $s_{d(u+v)}$ is small as compared with multiplications for second while loop.

New Fifth-degree Single Exponentiation: The double exponentiation Algorithm 10 can be used to speed up single exponentiation with some pre-computations [42]. Let $0 < w \leq \ell$ be integer, also let $w = \bar{b}\lambda + \bar{b}$ for some $\lambda \in \mathbb{Z}$. Use λ and initial conditions $s_{-1}, s_0, s_1, s_2, s_3$ and $\hat{s}_{-1}, \hat{s}_0, \hat{s}_1, \hat{s}_2, \hat{s}_3$ in Algorithm 9 to compute $s_{\lambda-3}, s_{\lambda-2}, s_{\lambda-1}, s_\lambda, s_{\lambda+1}$ and $\hat{s}_{\lambda-3}, \hat{s}_{\lambda-2}, \hat{s}_{\lambda-1}, \hat{s}_\lambda, \hat{s}_{\lambda+1}$ as pre-computations. After this let $k = \lambda$, $l = 1$, $\bar{a}, \bar{b}$ as given above, and $s_{\lambda+i}, \hat{s}_{\lambda+i}$ for $i = -3, -2, -1, 0, 1$ be given then following algorithm computes s_w.

Algorithm 11 Single Exponentiation for Fifth Degree Recursive Relation

Require: $(\bar{a} > 0, \bar{b} > 0) \in \ell$, $s_{\lambda+i}, \hat{s}_{\lambda+i}$ for $i = -3, -2, -1, 0, 1$.
Ensure: $s_{\bar{b}k+\bar{a}l} = s_w$
1: $f_2 \leftarrow 1$, $d \leftarrow \bar{b}$, $e \leftarrow \bar{a}$, $u \leftarrow k$, $v \leftarrow l = 1$
2: **if** (**then**$\bar{a} \neq 0$ and $\bar{b} \neq 0$)
3: Compute $s_{\bar{b}k+\bar{a}l} = s_w$ using Algorithm 10 with inputs $\bar{a}, \bar{b}, s_{\lambda+i}, \hat{s}_{\lambda+i}$ for $i = -3, -2, -1, 0, 1$.
4: **end if**
5: **return** $(s_{\bar{b}k+\bar{a}l} = s_w)$

Conjecture 5.3. *The single exponentiation that is computing s_w with Algorithm 11, given $0 < w < \ell$ $s_{\lambda+i}, \hat{s}_{\lambda+i}$ for $i = -3, -2, -1, 0, 1$, takes about $36\log(\bar{a} + \bar{b})$ multiplications over $\mathbb{F}_p$ where as Algorithm 9 takes $59\log(w)$ multiplications over F_p*

5.3.11 Koray Karabina PKC : Case $k = 4$, $p = 2$, $q = p^{2u+1}, u \geq 1$

K. Karabina introduced an efficient algorithm to compute $Tr(\alpha^m)$ in [41] for characteristic 2-case when q is odd power of 2. However, he has not discussed some of the cryptographic protocols attached to his algorithm $\alpha \to Tr(\alpha^m)$. We now discuss his algorithm and related protocols for the sake of completeness.

Let $G =< \alpha >\subset \mathbb{F}_{q^4}^*$, $t = \sqrt{2q}$, $p = 2$, $q = p^{2u+1}, u \geq 1$, $ord(\alpha) = \ell | (q - t + 1) | (q^2 + 1)$. Note that, $\Phi_4(q) = q^2 + 1 = (q + t + 1)(q - t + 1)$, is the 4th cyclotomic polynomial evaluated at q. Let $g(x)$ be the characteristic polynomial of α:

$$g(x) = \prod_{i=0}^{3} (x - \alpha^{q^i}) = x^4 - ax^3 + bx^2 - cx + 1.$$

One can check that $a = Tr(\alpha)$, $b = Tr(\alpha^{q+1}) = Tr(\alpha)^t$, and $c = Tr(\alpha)$. So $g(x)$ is completely determined by only one variable that is $Tr(\alpha)$. For any integer m, similarly $g_m(x)$ becomes;

$$g_m(x) = \prod_{i=0}^{3} (x - \alpha^{mq^i}) = x^4 - Tr(\alpha^m)x^3 + (Tr(\alpha^m)^{t)})x^2 - Tr(\alpha^m)x + 1.$$

Let $V_n : \mathbb{F}_q \to \mathbb{F}_q$ be the function defined by $V_n(a) = Tr(\beta^n)$, where β is the root of $h(x) = x^4 - ax^3 + a^t x^2 - ax + 1 = 0$. It is clear that V_n is independent than

the choice of the root β of $h(x) = 0$. Now replacing α by $\beta = \alpha^m$ above we get for any integer r the characteristic polynomial $h_r(x)$ of β^r;

$$\begin{aligned} h_r(x) &= \prod_{i=0}^{3} (x - \beta^{rq^i}), \\ &= x^4 - Tr(\alpha^{mr})x^3 + (Tr(\alpha^{mr}))^t x^2 - Tr(\alpha^m r)x + 1 = g_{mr}(x). \end{aligned}$$

This in particular gives, $V_m(V_r(a)) = V_{mr}(a) = s_{mr} = Tr(\alpha^{mr})$.

Properties: For all integers m and n we have following properties:

(i) $s_n = Tr(\alpha^n) = V_n(a)$.

(ii) $s_n = s_{-n}$.

(iii) $s_{2u} = s_u^2$.

(iv) $s_m s_n = s_{m+n} + s_{m-n} + s_{m+n(t-1)} + s_{n+m(t-1)}$.

(v) $s_{m+n} = s_m s_n - s_{m-n} s_{nt} + s_{m-2n} s_n - s_{m-3n}$.

K. Karabina's algorithm to compute $Tr(\alpha^n)$, given $Tr(\alpha)$, and n, is as follows:

Algorithm 12 Compute $s_n = Tr(\alpha^n)$, given $s = Tr(\alpha) \in \mathbb{F}_q$, n, $q = 2^{2r+1}$

Require: $n \in \mathbb{Z}_\ell : n = \sum_{i=0}^{l-1} n_i 2^i, n_i \in \{0, 1\}$ with $n_{l-1} \neq 0$, and initial conditions $(s_1, 0, s_1, s_1^2)$.
Ensure: $(s_{n-1}, s_n, s_{n+1}, s_{n+2})$.
1: $S_u \leftarrow [s_{u-2}, s_{u-1}, 0, s_{u+1}] = (s_1, 0, s_1, s_1^2)$.
2: $m_1 \leftarrow 1/s_1^{t+1}$ and $m_2 \leftarrow 1/c_1$.
3: **for** $r \leftarrow l - 2$ ***to*** 0 **do**
4: $\quad s_{2u-1} \leftarrow m_1((s_{u+1} + s + s_{u-1} + s_{u-2})^2 + (s_u + s_{u-1})^2(s^t + s^2))$.
5: $\quad s_{2u} \leftarrow s^2$
6: $\quad s_{2u+1} \leftarrow s_{2u-1} + m_2((s_{u+1} + s_{u-1})^2 + s_u^2 s^t)$
7: $\quad$ **if** $n_r = 1$ **then**
8: $\quad\quad s_{2u+2} \leftarrow s_{u+1}^2, (s_{3u-3}, s_{3u-2}, s_{3u-1}, s_{3u}, s_{3u+1})$,
9: $\quad\quad S_u \leftarrow [s_{2u-1}, s_{2u}, s_{2u+1}, s_{2u+2}]$.
10: $\quad$ **else**
11: $\quad\quad s_{2u-2} \leftarrow s_{u-1}^2$,
12: $\quad\quad S_u \leftarrow [s_{2u-2}, s_{2u-1}, s_{2u}, s_{2u+1}]$.
13: $\quad$ **end if**
14: **end for**
15: **return** (S_u).

The above algorithm computes $Tr(\alpha^n)$ given $Tr(\alpha), n$ in $(1I + 1M)$ as precomputation and $(4M + 4S)(l - 1)$ operations for main loop, where I, M, and S stand for inversion, multiplication and squaring respectively.

Simultaneous Double Exponentiation: Let $0 < a, b, m, n > \ell$ be integers the double exponentiation for this case is given below in Algorithm 13. The variation rules and corresponding effects on variables are given in Table 5.14 below:

Table 5.14: The Rules for Factor-4 Double Exponentiation

Rule	**Condition**	d	e	u	v	s_v	s_{2u-v}	$s_{u-2v}, s_{u-v}, s_{u+v}$
if $d > e$								
R_1	$d \leq 4e$	$d-e$	e	u	$u+v$	s_{u+v}	s_{u-v}	s_{u+2v}, s_v, s_{2u+v}
R_2	$d \equiv e \pmod 2$	$(d-e)/2$	e	$2u$	$u+v$	s_{u+v}	s_{3u-v}	s_v^2, s_{u-v}, s_{3u+v}
R_3	$d \equiv 0 \pmod 2$	$d/2$	e	$2u$	v	s_v	s_{4u-v}	$s_{u-v}^2, s_{2u-v}, s_{2u+v}$
R_4	$e \equiv 0 \pmod 2$	d	$e/2$	u	$2v$	s_v^2	s_{u-v}^2	$s_{u-4v}, s_{u-2v}, s_{u+2v}$
else								
Sub	$e > d$	e	d	v	u	s_v	s_{u-2v}	$s_{2u-v}, s_{u-v}, s_{u+v}$

Algorithm 13 Double Exponentiation for Factor 4, for NR-KK-DSA

Require: $(a > 0, b > 0) \in \mathbb{F}_q$, s_m, s_n,s_{n-m}, and s_{n-2m}.
Ensure: s_{an+bm}
1: $f_2 \leftarrow 1$, $d \leftarrow a$, $e \leftarrow b$, $u \leftarrow n$, $v \leftarrow m$
2: **while** d and e are both even **do**
3: $d \leftarrow d/2$, $e \leftarrow e/2$, $f_2 \leftarrow 2f_2$
4: **end while**
5: **while** $d \neq e$ **do**
6: Execute the first applicable rule in Table 5.14
7: **end while**
8: compute $s_{d(u+v)}$ using Algorithm 12 with inputs d and s_{u+v}.
9: **return** $(s_{(d(u+v))}^{f_2})$

Remark 5.14. The Algorithm 13 can compute s_{an+ml} in $\approx 6.37 \log(max(a, b))$ multiplications in $\mathbb{F}_p$ given s_m, $0 < a, b < \ell$, s_n,s_{n-m}, and s_{n-2m}.

Now we look at the protocols to validate this trace based PKC.

5.3.11.1 KK-DH Type Key Exchange : Case $k = 4$, $p = 2$, $q = p^{2u+1}, u \geq 1$

1. ***System Public Parameters:*** Let $G =< \alpha > \subset \mathbb{F}_{q^4}^*$, $p = 2$, $q = p^{2u+1}, u \geq 1$, $t = \sqrt{2q}$, $ord(\alpha) = \ell | (q - t + 1)$, and $g(x) = x^4 - ax^3 + bx^2 - cx + 1$, be the characteristic polynomial of α.

2. **A** computes her public key $P_A = V_e(a) = s_e$ by running Algorithm 12 with inputs a and her secret key e, and **B** computes his public key $P_B = V_r(a) = s_r$ by running Algorithm 12 with inputs a and his secret key r.

3. ***Private Keys:*** Random $1 < r < \mathbb{Z}_\ell$ and $1 < r < \mathbb{Z}_\ell$ are secret keys of **A** and **B** respectively.

4. **Common key:** Both **A** and **B** compute their common key $K = P_{AB} = P_{BA} = s_{er}$ as follows:

(i) **A** acquires **B**'s public key and runs Algorithm 12 with input s_r and her private key e to obtain common key: $K = P_{AB} = V_e(s_r) = V_{er}(a) = s_{er}$.

(ii) **B** does the similar things by using his private key r to compute common key: $K = P_{BA} = V_r(s_e) = V_{re}(a) = s_{re}$.

5.3.11.2 KK-ElGamal Type Encryption Scheme : Case $k = 4$, $p = 2$, $q = p^{2u+1}, u \geq 1$

1. ***System Public Parameters:*** Let $G =< \alpha >\subset \mathbb{F}_{q^4}^*$, $p = 2$, $q = p^{2u+1}, u \geq 1$, $t = \sqrt{2q}$, $ord(\alpha) = \ell | (q - t + 1)$, and $g(x) = x^4 - ax^3 + bx^2 - cx + 1$, be the characteristic polynomial of α.
2. **B**'s public key $P_B = V_r(a) = s_r$, private key: $1 < r < \mathbb{Z}_\ell$.
3. Assumption: messages M are in G.
4. **A** sends message M as follows:
   (i) Chooses a random $1 < e < \mathbb{Z}_\ell$ and computes encryption key $K = s_{er} + s_{er}^t$ such that $V_e(P_B) = V_e(s_r) = s_{er}$, mask s_e, by running Algorithm 12 and then computes $c = KM = s_{er}M$.
   (ii) **A** sends ciphertext $C = (s_e, c)$ to **B**.
5. ***Decryption:*** **B** decrypts C as follows:
   (i) Based on mask s_e and his private key r computes encryption key $K = s_{er} + s_{er}^t : V_r(s_e) = s_{re}$ by running Algorithm 12 with inputs s_e and his private key r.
   (ii) Then decrypts message by computing, $M = K^{-1}c = (s_{re} + s_{re}^t)^{-1}c$.

Remark 5.15. The semantic security of KK-ElGamal type encryption is computationally equivalent to splitting $z \in \mathbb{F}_q$ into two numbers $(X, Y) \in \mathbb{F}_q^2$ such that $z = X + Y$ where $s_{er} \mapsto X$ and $s_{-er} \mapsto Y$. Moreover, $er \in \mathbb{Z}_\ell$ and $\mathbb{Z}_e ll$ is chosen large enough so that the brute force attack becomes infeasible. The number of such (X, Y) are equal to $\frac{q-1}{2}$ as discussed in chapter 4, Remark 4.5.

5.3.11.3 KK-Nyberg Rueppel Type Digital Signature Algorithm Based on Generic Symmetric encryption : Case $k = 4$, $p = 2$, $q = p^{2u+1}, u \geq 1$

This scheme was introduced by Koray Karabina in [41]. Let **A** wants to send signed message M to **B** and **B** verifies it. To do this, **A** and **B** do the following:

1. ***System Public Parameters:*** Let $G =< \alpha >\subset \mathbb{F}_{q^4}^*$, $p = 2$, $q = p^{2u+1}, u \geq 1$, $t = \sqrt{2q}$, $ord(\alpha) = \ell | (q - t + 1)$, $g(x) = x^4 - ax^3 + bx^2 - cx + 1$, be the characteristic polynomial of α and $H : G \to \mathbb{Z}_\ell$-valued hash function.

2. ***Public Keys:***

   (i) **A** randomly selects $1 < m < \mathbb{Z}_\ell$ and computes public key $P_A = V_m(a) := S_m = \{s_{m-2}, s_{m-1}, s_m, s_{m+1}\}$ by running Algorithm 12 with inputs a and her private key m.

   (ii) **B** also randomly selects $1 < r < \mathbb{Z}_\ell$ and computes public key $P_B = V_r(a) := S_r = \{s_{r-2}, s_{r-1}, s_r, s_{r+1}\}$ by running Algorithm 12 with inputs a and his private key r.

3. ***Private Keys:*** The private keys of **A** and **B** are m and r, respectively.

4. ***Signature***: **A** signs the message M as follows:

   (i) **A** randomly selects $1 < d < \mathbb{Z}_\ell$, and computes mask $V_d(a) = s_d$, and extracts session key $K = Ext(s_d)$ from s_d.

   (ii) **A** obtains ciphertext C by encrypting the message M with K, using generic symmetric encryption and computes the hash value of C, that is $h = H(C) \mod \ell$.

   (iii) **A** computes $n = d + mh \mod \ell$.

   (iv) **A** sends the signature (n, C) to **B**.

5. ***Verification***: **B** verifies **A**'s signature and recovers message as follows:

   (i) Computes $h = H(C) \mod \ell$ and replaces h by $-h$.

   (ii) Computes s_{hm+n} from $S_m = \{s_{m-2}, s_{m-1}, s_m, s_{m+1}\}$ and a using double exponentiation Algorithm 13.

   (iii) Computes session key $\acute{K} = Ext(s_{n+hm})$ from s_{n+mh} and computes $\acute{C}$ using $\acute{K}$ and M.

   (iv) **B** accepts if and only if $\acute{C} = C$.

The analysis of KK-NR-DSA based on generic symmetric encryption scheme is given in the Table 5.15 below.

Table 5.15: Analysis of KK-NR-DSA based on Generic Symmetric Encryption

Process	**Cost**
Signature Generation	$\approx 6.37 \log p$ **M**
Signature Verification	$\approx 6.37 \log p$ **M**
Comm. Overhead	2 elements over $\mathbb{F}_q$
Public Key size	4 elements over $\mathbb{F}_q$

5.3.11.4 KK-Nyberg Rueppel Type Digital Signature Algorithm Based on KK-ElGamal type encryption : Case $k = 4$, $p = 2$, $q = p^{2u+1}, u \geq 1$

Let **A** wants to send signed encrypted message M containing agreed upon redundancy to **B** and **B** verifies and recovers the message . To do this, **A** and **B** do the following:

1. ***System Public Parameters:*** Let $G =< \alpha >\subset \mathbb{F}^*_{q^4}$, $p = 2$, $q = p^{2u+1}, u \geq 1$, $t = \sqrt{2q}$, $ord(\alpha) = \ell | (q - t + 1)$, $g(x) = x^4 - ax^3 + bx^2 - cx + 1$, be the characteristic polynomial of α and $H : G \to \mathbb{Z}_\ell$-valued hash function.
2. ***Public Keys:***
   (i) **A** randomly selects $1 < m < \mathbb{Z}_\ell$ and computes public key $P_A = V_m(a) = s_m$ by running Algorithm 12 with inputs a and her private key m.
   (ii) **B** also randomly selects $1 < r < \mathbb{Z}_\ell$ and computes public key $P_B = V_r(a) = s_r$ by running Algorithm 12 with inputs a and his private key r.
3. ***Private Keys:*** The private keys of **A** and **B** are m and r, respectively.
4. ***Signature***: **A** signs the message M as follows:
   (i) **A** randomly selects ephemeral private key $1 < d < \mathbb{Z}_\ell$ and computes ephemeral public key $V_d(a) = s_d$, and computes encryption key $K = s_{dr} + s^t_{dr} : V_d(s_r) = s_{dr}$.
   (ii) **A** computes $h = H(M) \mod \ell$, obtains ciphertext $C = MK$.
   (iii) **A** computes $n = d - mh \pmod \ell$.
   (iv) **A** sends the signature (n, s_d, C) to **B**.
5. ***Verification***: **B** recovers message M and verifies **A**'s signature as follows:
   (i) Checks $0 \leq n < \ell$, if not failure.
   (ii) Computes encryption key $K = s_{rd} + s^t_{rd} : V_r(s_d) = s_{rd}$, and decrypts C to $M : M = CK^{-1}$. If M does not contain agreed upon redundancy then failure.
   (iii) Computes $h = H(M) \mod \ell$ and computes $S_{hm} := \{s_{hm-2}, s_{hm-1}, s_h m, s_{hm+1}\}$ using Algorithm 12.
   (iv) Computes s_{hm+n} from $S_m = \{s_{m-2}, s_{m-1}, s_m, s_{m+1}\}$ and a using double exponentiation Algorithm 13.
   (v) Accepts if $s_{n+hm} = s_d$.

The analysis of KK-NR-DSA based on KK-ElGamal encryption scheme is given in the Table 5.16 below.

Remark 5.16. The KK-NR-DSA based on KK-ElGamal type encryption scheme is resistant to both forgery attacks discussed in Remark **??**.

Table 5.16: Analysis of KK-NR-DSA based on KK-ElGamal Encryption

Process	**Cost**
Signature Generation	$\approx 6.37 \log p$ **M**
Signature Verification	$\approx 12.74 \log p$ **M**
Comm. Overhead	2 elements over $\mathbb{F}_q$
Public Key size	1 element over $\mathbb{F}_q$

5.3.12 KK-PKC : Case $k = 6$, $p = 3$, $q = p^u, u \geq 1$, $u = odd$,$t = \sqrt{3q}$

Let $\alpha \in \mathbb{F}_{q^6}$ with $ord(\alpha) = \ell | (q \pm t + 1)$. Note that $\ell | \Phi_6(q) = (q^2 - q + 1) = (q + t + 1)(q - t + 1)$. Here $\Phi_6(q)$ is the 6th cyclotomic polynomial evaluated at q. It is shown in [41] that the characteristic polynomial $g(x)$ of α is nothing but in the form,

$$\begin{aligned} g(x) &= \prod_{i=0}^{6} (x - \alpha^{q^i}), \\ &= x^6 - (Tr(\alpha))x^5 + (Tr(\alpha)^t + Tr(\alpha))x^4 - (Tr(\alpha^2) + 2Tr(\alpha) + 2)x^3 \\ &- (Tr(\alpha)^t + Tr(\alpha))x^2 + (Tr(\alpha))x - 1. \end{aligned}$$

Note that it is also shown in [41] that $Tr(\alpha^2) = Tr(\alpha)^2 + Tr(\alpha) + Tr(\alpha)^t$, and therefore $g(x)$ becomes;

$$\begin{aligned} g(x) &= x^6 - (Tr(\alpha))x^5 + (Tr(\alpha)^t + Tr(\alpha))x^4 - (Tr(\alpha)^2 + Tr(\alpha)^t + 2)x^3 \\ &- (Tr(\alpha)^t + Tr(\alpha))x^2 + (Tr(\alpha))x - 1. \end{aligned}$$

Hence $g(x)$ is completely determined by $Tr(\alpha)$. For any integer m, let,

$$g_m(x) = \prod_{i=0}^{6} (x - (\alpha^m)^{q^i}),$$

be the characteristic polynomial of α^m, then it can be checked that

$$\begin{aligned} g_m(x) &= x^6 - (Tr(\alpha^m))x^5 + (Tr(\alpha^m)^t + Tr(\alpha^m))x^4 \\ &- (Tr(\alpha^m)^2 + Tr(\alpha^m)^t + 2)x^3 - (Tr(\alpha^m)^t + Tr(\alpha^m))x^2 \\ &+ (Tr(\alpha^m))x - 1. \end{aligned}$$

Now let us introduce $V_r : \mathbb{F}_q \to \mathbb{F}_q$ as follows, for any b; $V_r(b) = Tr(\beta)$, where β is a root of,

$$h(x) = x^6 - bx^5 + (b + b^t)x^4 - (b^2 + b^t + 2)x^3 + (b^t + b)x^2 - bx + 1 = 0.$$

It is clear that V_r is well defined and for any integer m,

$$V_m(V_r(b)) = V_{mr}(b) = s_{mr}.$$

Properties: It is shown in [41] that for all integers m, n we have the following properties;

(i) $s_{-n} = s_n$.

(ii) $s_{2n} = s_n^2 + s_n + s_n^t$.

(iii) $s_n s_m = s_{n+m} + s_{n-m} + s_{n+m(t-1)} + s_{n+m(t-2)} + s_{m+n(t-2)}$.

(iv) $s_{n+m} = s_n s_m - (s_{n-m} + s_{n-3m})(s_v^t + s_v) + s_{n-2m}(s_v^2 + s_v^t + 2) + s_{n-4m} s_m - s_{n-5m}$.

Based upon above properties K. Karabina in [41] also gave algorithm to compute $s_n = Tr(\alpha^n)$, given n and $Tr(\alpha)$ which is as follows:

Algorithm 14 Compute $s_n = Tr(\alpha^n)$, given $s = Tr(\alpha)$, n, $q = 3^u$, $u = odd$

Require: $n \in \mathbb{Z}_\ell : n = \sum_{j=0}^{l-1} n_j 2^j, n_j \in \{0,1\}$ with $n_{l-1} \neq 0$, and initial conditions $(s_1, 0, s_1, s_2, s_1^3, s_4)$.

Ensure: $(s_{n-2}, s_{n-1}, s_n, s_{n+1}, s_{n+2}, s_{n+3})$.

1: $C_2 \leftarrow s_1^2$ and $C_t \leftarrow s_1^{t+1}$, $C_{1,t} \leftarrow s_1 + s_1^t$, $C_{1,t,2} \leftarrow C_2 + s_1^t + 2$,

2: $s_2 \leftarrow C_{1,t,2} + s_1 + 1$, $s_4 \leftarrow s_2^2 + s_2 + s_2^t$.

3: $S_u \leftarrow (s_{u-2}, s_{u-1}, s_u, s_{u+1}, s_{u+2}, s_{u+3}) = (s_1, 0, s_1, s_2, s_1^3, s_4)$.

4: $M \leftarrow (s_1^t(s_1^3 + C^t + 2(s_1^t + 1)) + 2(s_1^3 + s_1 + 1))^{-1}$

5: **for** $i \leftarrow l - 2$ ***to*** 0 **do**

6: $s_{2u-4} \leftarrow s_{u-2}^2 + s_{u-2} + s_{u-2}^t$, $s_{2u-2} \leftarrow s_{u-1}^2 + s_{u-1} + s_{u-1}^t$, $s_{2u} \leftarrow s_u^2 + s_u + s_u^t$, $s_{2u+2} \leftarrow s_{u+1}^2 + s_{u+1} + s_{u+1}^t$, $s_{2u+4} \leftarrow s_{u+2}^2 + s_{u+2} + s_{u+2}^t$, $s_{2u+6} \leftarrow s_{u+3}^2 + s_{u+3} + s_{u+3}^t$,

7: $Y_1 \leftarrow s_1(s_{2u+2} + s_{2u-2}) + s_{2u} C_{1,t,2}$,

8: $Y_2 \leftarrow s_1(s_{2u+4} + s_{2u}) + s_{2u+2} C_{1,t,2}$,

9: $Y_4 \leftarrow C_{1,t}(s_{2u} + s_{2u-2}) + s_{2u+2} + s_{2u-4}$,

10: $Y_5 \leftarrow C_{1,t}(s_{2u+2} + s_{2u}) + s_{2u+4} + s_{2u-2}$,

11: $Y_6 \leftarrow C_{1,t}(s_{2u+4} + s_{2u+2}) + s_{2u+6} + s_{2u}$.

12: $s_{2u-1} \leftarrow M((s_1^3 + 2(C_2 + s_1) + C_t)Y_1 + C_2 Y_2 + (2(C_2 + s_1^t) + s_1 + 1)Y_4 + (2(C_2 + C_t) + s_1)Y_5 + 2s_1 Y_6)$

13: $s_{2u+1} \leftarrow M(2C_2(Y_1 + Y_2) + s_1(Y_4 + Y_6) + (2s_1^t + C_t + s_1 + 1)Y_5)$

14: $s_{2u+3} \leftarrow M(C_2 Y_1 + (s_1^3 + 2(C_2 + s_1) + C_t)Y_2 + 2s_1 Y_4 + (2(C_2 + C_t) + s_1)Y_5 + (2(C_2 + s_1^t) + s_1 + 1)Y_6)$

15: **if** $n_i = 1$ **then**

16: $S_u \leftarrow (s_{u-1}, s_u, s_{u+1}, s_{u+2}, s_{u+3}, s_{u+4})$

17: **else**

18: $S_u \leftarrow (s_{u-2}, s_{u-1}, s_u, s_{u+1}, s_{u+2}, s_{u+3})$

19: **end if**

20: **end for**

21: **return** (S_u).

The above algorithm computes $Tr(\alpha^n)$ given, $Tr(\alpha)$ and n in $(1I + 2M + 2S)$ as precomputation and $(53A + 6F + 23M + 6S)(l - 1)$ for the main loop, where A, M, S, F, I stands for addition, multiplication, squaring, exponentiation by the power of the finite field characteristic and inversion respectively. In literature we do not see cryptographic protocols for this case as well. We add the cryptographic protocols to this system and comment it is a valid PKC.

5.3.12.1 KK-DH Type Key Exchange : Case $k = 6$, $p = 3$, $q = p^u$, $u = odd$

1. ***System Public Parameters:*** Let $G =< \alpha >\subset \mathbb{F}_{q^6}^*$, $p = 3$, $q = p^u$, $u = odd$, $t = \sqrt{3q}$, $ord(\alpha) = \ell|(q - t + 1)$, and $g(x) = x^6 - ax^5 + bx^4 - cx^3 + dx^2 - ex + 1$, be the characteristic polynomial of α.

2. ***Public Keys:*** **A**'s public key $P_A = V_m(a) = s_m$, and **B**'s public key $P_B = V_r(a) = s_r$.

3. ***Private Keys:*** The private keys of **A** and **B** are random $1 < m < \mathbb{Z}_\ell$ and $1 < r < \mathbb{Z}_\ell$ respectively.

4. **Common key:** Both A and B computes common key $K = P_{AB} = P_{BA} = s_{mr}$.

(i) **A** acquires **B**'s public key and runs Algorithm 14 with input s_r and her private key m to obtain common key: $K = P_{AB} = V_m(s_r) = s_{mr}$.

(ii) **B** does the similar things to compute common key using his private key r: $K = P_{BA} = V_r(s_m) = s_{mr}$.

It is not difficult to give the KK-ElGamal Type encryption scheme and KK-Nyberg Rueppel Type DSA for this case also. It is step by step similar to the one discussed in subsections 5.3.11.2 and 5.3.11.4 respectively, so we leave it to the reader for verification.

Remark 5.17. It is clear that the security of trace based Public Key Cryptosystems so far discussed depends on the trapdoor function $\alpha \to Tr(\alpha^m)$ which is equivalent to DLP on the group $G =< \alpha >\subset \mathbb{F}_{q^k}$. Therefore for the maximum security one should choose α such that $\mathbb{F}_{q^k} = \mathbb{F}_q(\alpha)$, namely characteristic polynomial of α is equal to its minimal polynomial, and DLP on G is computationally equivalent to DLP on $\mathbb{F}_{q^k}^*$. Note that the semantic security is computationally equivalent to splitting $z \in \mathbb{F}_q$ into two numbers $(X, Y) \in \mathbb{F}_q^2$ such that $z = X + Y$ where $s_e \mapsto X$ and $s_{-e} \mapsto Y$. Moreover, $e \in \mathbb{Z}_\ell$ and $\mathbb{Z}_ell$ is chosen large enough so that the brute force attack becomes infeasible. The number of such (X, Y) are equal to $\frac{q-1}{2}$.

5.4 Conclusion

For a given $\alpha \in \mathbb{F}_{q^k}$, one can compute for any integer m the $Tr(\alpha^m)$ by repeated squaring and then trace mapping in a polynomial time. We have searched the literature and found that for some special conditions on α, q, and k there are more efficient algorithms to compute $Tr(\alpha^m)$ from the above classical one. We also found out that in some of these cases there are cryptographic protocols based on these algorithms which work more efficiently and securely. In this chapter we discussed most of these cases and introduced cryptographic protocols to the

ones they were not discussed. We also ensured that the semantic security of the encryption schemes is not compromised either by modifying existing ones or introducing new ones, except LUC-PKC. We also introduced double exponentiation and subsequently single exponentiation for fifth degree recursive relations. Conjecturally, proposed exponentiation is at least 25% faster than previous methods for fifth degree extensions.

CHAPTER 6

SUMMARY

Following is the summary of work done in this thesis:

(a) The alternative models of elliptic curves are surveyed by pinning down group operations, and performance in various coordinate systems. A summary was written in tabular form including the costs of addition, doubling, mixed addition and unified addition on various models of elliptic curves. It was noted that the unified addition formulae offer inherited countermeasure against Simple Power Analysis(SPA) with comparable performance. In these models, SPA is avoided by employing the unified addition formulae or an algorithmic adaptation of it that behaves in similar fashion during the process of point addition and point doubling. Hence, for algorithmic flexibility, alternate models of elliptic curves with desirable properties are put together in this chapter that can be adopted cryptographic protocols.

(b) The GH-ElGamal type encryption scheme is introduced, which is semantically secure and semantic security depends upon splitting $z \in \mathbb{F}_q$ into two numbers $(X, Y) \in \mathbb{F}_q$ such that $z = X + Y$. The number of such (X, Y) are equal to $\frac{q-1}{2}$. Whereas the key exchange security depends on the difficulty of solving 3-LFSR-DLP and 3-LFSR-DHP. The GH-ElGamal encryption scheme is faster than GH-RSA-type encryption scheme with lesser storage memory. Based upon the proposed encryption scheme, we also introduced GH-NR-DSA to embed key exchange, message transmission and digital signature algorithm in a single protocol to ensure confidentiality, integrity and non-repudiation.

(c) In addition to GH-PKC we introduced modified XTR-ElGamal(MXTR-ElGamal) type encryption scheme for cubic extensions. In this modification one can transmit encrypted message over $\mathbb{F}_p$ instead of $\mathbb{F}_{q=p^2}$. The semantic security is also ensured in this case, which depends upon semantic security depends upon splitting $z \in \mathbb{F}_p$ into two numbers $(X, Y) \in \mathbb{F}_p$ such that $z = X + Y$. The number of such (X, Y) are equal to $\frac{p-1}{2}$. Based upon MXTR-ElGamal encryption scheme the XTR-NR-DSA is also introduced. The GG-ElGamal type encryption scheme and GG-NR-DSA are also added to the literature. We also adopted efficient double exponentiation and subsequently single exponentiation for $\mathbb{F}_{q^5}$ case.

Problem : Theoretical computational cost of 2nd while loop in double exponentiation algorithm may be worked out.

REFERENCES

[1] M. Abe, T. Okamoto, "*A signature scheme with message recovery as secure as discrete logarithm*", in Advances in Cryptology Asiacrypt'99, Lect. Notes in Comp. Sci. 1716, Springer-Verlag, pp. 378389, 1999.

[2] L. Adleman, K.Manders and G.Miller, *"On taking roots in finite fields"*, Proc. 18th IEEE Symposium on Foundation of Computer Science (FOCS), pp. 175-177, 1977.

[3] D. Agrawal, B. Archambeault, J.R. Rao, P. Rohatgi, *"The EM Side-Channel(s)"*. Cryptographic Hardware and Embedded Systems - CHES 2002, Lecture Notes in Computer Science Vol. 2523. Springer-Verlag, pp. 29-45, 2003.

[4] C. Arenea, T. Lange, M. Naehrig, C. Ritzenthaler, *"Faster Computation of the Tate Pairing"*, Journal of Number Theory 131(5), pp. 842-857, 2011.

[5] M. Ashraf and B. B. Kırlar, *"Efficient Message Transmission for GH-Public Key Cryptosystem"*, Institute of Applied Mathematics, number 2013-08, Preprint, 2013.

[6] M. Ashraf and B. B. Kırlar, *"On the Alternate Models of Elliptic Curves"*, IJISS, Vol.1, No 2(2012).

[7] D. Bernstein and T. Lange, *"Faster addition and doubling on elliptic curves"*. Progress in Cryptology - Africacrypt 2007, Lecture Notes in Computer Science Vol. 4833, Springer, pp. 29-50, 2007.

[8] D. Bernstein and T. Lange, *"Explicit Formulas Database"*, Available at http://www.hyperelliptic.org/EFD

[9] D. Bernstein and T. Lange, *"Inverted Edwards coordinates"*. Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 17th International Symposium - AAECC-17, Lecture Notes in Computer Science Vol. 4851, Springer, pp. 20-27, 2007.

[10] D. Bernstein, D. Kohel and T. Lange, *"Twisted Hessian curves"*. Available at http://www.hyperelliptic.org/EFD/g1p/auto-twistedhessian.html.

[11] D. Bernstein, T. Lange and R. R. Farashahi, *"Binary Edwards Curves"*. Cryptographic Hardware and Embedded Systems - CHES 2008, Lecture Notes in Computer Science Vol. 5154, Springer, pp. 244-265, 2008.

[12] D. Bernstein, P. Birkner, M. Joye, T. Lange and C. Peters, *"Twisted Edwards curves"*, Progress in Cryptology - Africacrypt 2008, Lecture Notes in Computer Science Vol. 5023, Springer, pp. 389-405, 2008.

[13] E. Biham and A. Shamir, *"Differential fault analysis of secret key cryptosystems"*. Advances in Cryptology - Crypto '97, Lecture Notes in Computer Science Vol. 1294, Springer-Verlag, pp. 513-525, 1997.

[14] O. Billet and M. Joye, *"The Jacobi model of an elliptic curve and side-channel analysis"*, AAECC 2003, Lecture Notes in Computer Science Vol. 2643, Springer-Verlag, pp. 34-42, 2003.

[15] I. F. Blake, G. Seroussi and N. P. Smart, *"Advances in Elliptic Curve Cryptography"*, London Mathematical Society Lecture Note Series 317, Cambridge University, 2005.

[16] W. Bosma, J. Hutton, E. R. Verheul, *"Looking beyond XTR"*, in Advances in Cryptology - Asiacrypt 2002, Lect. Notes in Comp. Sci. 2501, Springer, Berlin, pp. 46-63, 2002.

[17] E. Brier and M. Joye, *"Weierstrass elliptic curves and side-channel attacks"*. Public Key Cryptography - PKC 2002, Lecture Notes in Computer Science Vol. 2274, Springer, pp. 335-345, 2002.

[18] G.H. Cho, N. Koo, E.Ha and S. Kwon, *"New cube root algorithm based on third order linear recurrence relation in finite field"*, eprint, available from http://eprint.iacr.org/2013/024.pdf, 2013.

[19] M. Cipolla, *"Un metodo per la risolutione della congruenza di secondo grado"*, Rendiconto dell' Accademia Scienze Fisiche e Matematiche, Napoli, Ser3, Vol.IX, pp. 154-163, 1903.

[20] A. A. Ciss and D. Sow, *"On a New Generalization of Huff Curves"*. Available at http://eprint.iacr.org/2011/580.pdf.

[21] J. Devigne and M. Joye, *"Binary Huff Curves"*. Topics in Cryptology - CT-RSA 2011, Lecture Notes in Computer Science Vol. 6558, Springer, pp. 340-355, 2011.

[22] W. Diffie, M.E. Hellman, *"New directions in cryptography"*, IEEE Trans. Inform. Theory 22, pp. 644-654, 1976.

[23] H. Edwards, *A normal form for elliptic curves*. Bulletin of the American Mathematical Society 44(3) , pp. 393-422, 2007.

[24] R. R. Farashahi and M. Joye, *"'Efficient Arithmetic on Hessian Curves"*. Public Key Cryptography - PKC 2010, Lecture Notes in Computer Science Vol. 6056, Springer, pp. 243-260, 2010.

[25] R. Feng, M. Nie and H. Wu, *"'Twisted Jacobi Intersections Curves"'*. Available at http://eprint.iacr.org/2009/597.pdf

[26] C. M. Fiduccia, *"An efficient formula for linear recurrences"*, SIAM J. Comput. 14, pp. 106-112, 1985.

[27] K. Giuliani, G. Gong, *"Efficient Key Agreement and Signature Schemes Using Compact Representations in GF(p10)"*, Proceedings of IEEE International Symposium on Information Theory - ISIT 2004 (Chicago, IL, 2004), pp. 13-13

[28] K. Giuliani, G. Gong, *"Analogues to the Gong-Harn and XTR Cryptosystems"*, Combinatorics and Optimization Research Report CORR 2003-34, University of Waterloo Canada.

[29] K. J. Giuliani, G. Gong, *"New LFSR-Based Cryptosystems and the Trace Discrete Log Problem (Trace-DLP)"*, Sequences and Their Applications - SETA 2004, Lect. Notes in Comp. Sci. 3486, pp 298-312, 2005.

[30] G. Gong, L. Harn, *"Public-key cryptosystems based on cubic finite field extensions"*, IEEE Trans. Inform. Theory 45, pp. 2601-2605, 1999.

[31] G. Gong, L. Harn, and H. Wu, *"The GH Public-key cryptosystem"*, SAC 2001, Lect. Notes in Comp. Sci. 2259, pp. 284-300, 2001.

[32] G. Gong, A. Hassan, H. Wu and A. Youssef, *"An Efficient Algorithm for Exponentiation in DH Key Exchange and DSA in Cubic Extension Fields"*, CORR 2002-27.

[33] D. M. Gordon,*"Discrete logarithm in GF(p) using the number field sieve"*, SIAM J. Discrete Math., 6(1):124-138, 1993.

[34] H. Gu, D. Gu and W. Xie, *"Efficient Pairing Computation on Elliptic Curves in Hessian form"*. Information Security and Cryptology - ICISC 2010, Lecture Notes in Computer Science Vol. 6829, Springer, pp. 169-176, 2011.

[35] T. S. Gustavsen and K. Ranestad, *"A Simple Point Counting Algorithm for Hessian Elliptic Curves in Characteristic Three"*. Appl. Algebra Eng. Commun. Comput. 17(2), pp. 141Ã¯Â¿Â½150, 2006.

[36] H. Hisil, K. Koon-Ho Wong, G. Carter and E. Dawson, *"Twisted Edwards Curves Revisited"*. Advances in Cryptology - Asiacrypt 2008, Lecture Notes in Computer Science Vol. 5350, Springer-Verlag, pp. 326-343, 2008.

[37] H. Hisil, K. Koon-Ho Wong, G. Carter and E. Dawson, *"Jacobi Quartic Curves Revisited"*. ACISP 2009, pp. 452-468, 2009.

[38] G. Huff, *"Diophantine problems in geometry and elliptic ternary forms"*. Duke Math. J., 15, pp. 443-453, 1948.

[39] M. Joye and J. Quisquater, *"Hessian elliptic curves and side-channel attacks"*. Cryptographic Hardware and Embedded Systems - CHES 2001, Lecture Notes in Computer Science Vol. 2162, Springer, pp. 402-410, 2001.

[40] M.Joye, M. Tibbouchi and D. Vergnaud, *"Huff's Model for Elliptic Curves"*. Algorithmic Number Theory - ANTS-IX, Lecture Notes in Computer Science Vol. 6197, Springer, pp. 234-250, 2010.

[41] K. Karabina, *"Factor-4 and 6 compression of cyclotomic subgroups of* $\mathbb{F}_{2^{4m}}$ *and* $\mathbb{F}_{3^{6m}}$*"*, J.Math Cryptol. 4, 1-42 (2010)

[42] K. Karabina, *"Double Exponentiation in Factor-4 groups and it applications"*, eprint.iacr.org/2009/475.pdf.

[43] P. C. Kocher, *"Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems"*. Advances in Cryptology - Crypto '96, Lecture Notes in Computer Science Vol. 1109, Springer, pp. 104-113, 1996.

[44] P. Kocher, J. Jaffe and B. Jun, *"Differential Power Analysis"*. Advances in Cryptology - Crypto'99, Lecture Notes in Computer Science Vol. 1666, Springer-Verlag, pp. 388-397, 1999.

[45] S. Lang, *"Algebra"*, Springer, 2005.

[46] D.H. Lehmer, *"Computer technology applied to the theory of numbers"*, Studies in Number Theory, Englewood Cliffs, NJ: Pretice-Hall, pp. 117-151, 1969.

[47] A. K. Lenstra, E. R. Verheul, *"The XTR public key system, in Advances in Cryptology"*, Crypto 2000, Lect. Notes in Comp. Sci. 1880, Springer, Berlin, pp. 1-19, 2000.

[48] A. K. Lenstra, E. R. Verheul, *"An overview of the XTR public key system"*, in Publickey cryptography and computational number theory (Warsaw, 2000), de Gruyter, Berlin, pp. 151-180, 2001.

[49] P. Liardet and N. Smart, *"Preventing SPA/DPA in ECC systems using the Jacobi form"*. Cryptographic Hardware and Embedded Systems - CHES 2001, Lecture Notes in Computer Science Vol. 2162, Springer-Verlag, pp. 391-401, 2001.

[50] R. Lidl and H. Niederreiter. *"Finite Fields"*, Addison-Wesley Publishing Company, Reading, MA, 1983.

[51] C. Lim, P. Lee, *"A key recovery attack on discrete log-based schemes using a prime order subgroup"*, in Advances in Cryptology Crypto'97, Lect. Notes in Comp. Sci. 1294, 249-263, 1997.

[52] A. Mahalanobis, *"The discrete logarithm problem in the group of non-singular circulant matrices"*, Group Cryptology 2 (2010), 83-39.

[53] A. Mahalanobis, *"The ElGamal Cryptosystem over Circulant Matrices"*, eprint.iacr.org/2011/572 (2011).

[54] A. Menezes and Y.H. Wu, *"The discrete logarithm problem in* $GL(n, q)$*"*, Ars comb. 47(1997),23-32.

[55] A. Miyaji, *"A message recovery signature scheme equivalent to dsa giving message recovery"*, in Advances in Cryptology Asiacrypt'96, Lect. Notes in Comp. Sci. 1163, Springer-Verlag, pp. 114, 1996.

[56] A. Miyaji, *"Weakness in message recovery signature schemes based on discrete logarithm problems 1"*, IEICE Japan Tech. Rep., ISEC95-7, 1995.

[57] P. L. Montgomery, *"Evaluating recurrences of form* $X_{m+n} = f(X_m; X_n; X_m..n)$ *via Lucas chains"*, Revised (1992) version from ftp.cwi.nl: /pub/pmontgom/Lucas.ps.gz, 1983.

[58] S. Muller, *"On the computations of square roots in finite fields"*, Design Codes and Cryptography, Vol. 31, ppl 301-312, 2004.

[59] K. Nyberg, R. A. Rueppel, *"A new signature scheme based on the DSA giving message recovery"*, 1st ACM Conference on Computer and Communications Security, Nov 35, Fairfax, Virginia, 1993.

[60] K. Nyberg, R. A. Rueppel, *"Message recovery for signature schemes based on the discrete logarithm problem"*, Designs, Codes and Cryptography, Volume 7, Issue 1 to 2, pp. 61-81, 1996.

[61] J. Pollard, *"Monte Carlo methods for index computation* (mod p)*"*, Mathematics of Computation, 32(143): 918-924, 1978.

[62] J. Pollard, *"Kangaroos, monopoly and discrete logarithms"*, Journal of Cryptology, 13(4):437-447, 2000.

[63] R.Rivest, A.Shamir, and L. Adleman, *"A method for obtaining digital signature and public key cryptosystems"*, Comm. of the ACM, 21(2):120-126, 1978.

[64] K. Rubin and A. Silverberg, *"Torus-based cryptography"*. Advances in Cryptology - CRYPTO 2003, Lecture Notes in Computer Science, 2729:349-365, 2003.

[65] K. Rubin and A. Silverberg, *"Compression in Finite Fields and Torus-Based Cryptography"*, SIAM J. Comput. 37(5), pp. 1401-1428, 2008.

[66] D. Shanks, *"Five number-theoretic algorithms"*, Proc. 2nd Manitoba Conf. Number. Math., Manitoba, Canada, pp. 51-70, 1972.

[67] D. Shanks, *"Class number, a theory of factorization, and genera"*, In 1969 Number Theory Institute (Proc. Sympos. Pure Math., Vol. XX, State Univ. New York, Stony Brook, N.Y., 1969), pages 415-440, Providence, R.I., 1971. Amer. Math. Soc.

[68] M. Shirase, D. Han, Y. Hibin, H. Kim, and T. Takagi, *"A more compact representation of XTR cryptosystem"*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E91-A, pp. 2843-2850, 2008.

[69] J.H. silverman, *"Rings with low multiplicative complexity"*, Finite Fields and their appl. 6(2000), 175-191.

[70] N. Smart and E. J. Westwood, *"Point Multiplication on Ordinary Elliptic Curves over Fields of Characteristic Three"*. Appl. Algebra Eng. Commun. Comput. 13(6), pp. 485-497, 2003.

[71] P. J. Smith, M. J. J. Lennon, *"LUC: A New Public Key System"*, in Proceedings of the IFIP TC11 Ninth International Conference on Information Security IFIP/Sec'1993, pp. 103-117, 1993.

[72] P. Smith, C. Skinner, *"A public-key cryptosystem and a digital signature system based on the Lucas function analogue to discrete logarithms"*, in Advances in Cryptology - Asiacrypt 1994, Lect. Notes in Comp. Sci. 917, Springer, Berlin, pp. 357-364, 1995.

[73] M. Stam and A. Lenstra, *"Speeding up XTR"*, Advances in Cryptology - Asiacrypt 2001, Lect. Notes in Comp. Sci. 2248, pp. 125-143, 2001.

[74] C. H. Tan, X. Yi, C. K. Siew, *"On the n-th Order Shift Register Based Discrete Logarithm"*. IEICE Trans. Fundamentals. E86-A, pp. 1213-1216, 2003.

[75] A. Tonelli, *"Bemerking uber Auflosung quadratischer congruenzen"*, Gottinger Nachrichten, pp. 344-346, 1891.

[76] H. Wang, K. Wang, L. Zhang and B. Li, *"Pairing Computation on Elliptic Curves of Jacobi Quartic Form"*, Chinese Journal of Electronics 20(4), pp. 655-661, 2011.

[77] H. Wu and R. Feng, *"Elliptic curves in Huff's model"*. Available at http://eprint.iacr.org/2010/390.pdf, 2010.

[78] C. Y. Yeun, *"Digital signature with message recovery and authenticated encryption (signcryption) a comparison"*, in IMA - Cryptography and Coding'99, Lect. Notes in Comp. Sci. 1746, pp. 307312, 1999.

[79] SP 800-56A Special Publication 800-56A, *"Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"*, National Institute of Standards and Technology, March 2007.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Ashraf, Muhammad
Nationality: Pakistani
Date and Place of Birth:01-3-1973, Hafizabad, Pakistan
Marital Status: Married
Phone: 00905433249442
Fax:r

EDUCATION

Degree	Institution	Year of Graduation
M.S.	National University of Science and Technology	2008
B.E.	National University of Science and Technology	1996
High School	Government College Gujranwala	1990

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
Feb 1997 to Feb 1998	College of Aeronautical Engineering	Assist. Prof. of Elect.

PUBLICATIONS

International Conference Publications

(i) *Exploiting silence for cipher text only cryptanalysis of stream cipher digitized voice*,. International Conference on Security and Management LasVegas Nevada USA 25 - 28 June 2007.

(ii) *Efficient Implementation of 2D-DWT by purging Read After Write dependency for high speed applications*. International Conference on Emerging Technologies 2007, Islamabad, Pakistan 13-14 November 2007.

(iii) M. Ashraf, B.B. Kirlar, *Compressed Data Public Key Cryptosystems with DLP over Extension Fields*, ISCTurkey 2012.

(iv) M. Ashraf, B.B. Kirlar, *Alternate Models of Elliptic Curves: A Survey*, IJISS, Vol 1 of 2, 2013.

(iv) M. Ashraf, B.B. Kirlar, *Efficient Message Transmission for GH Public Key Cryptosystem*, ICACM, Oct, 2013.

(iv) *An Overview of Trace Based Public Key Cryptosystems over Finite Fields*, ICACM, Oct, 2013.