Addition Chains in Application to Elliptic Curve Cryptosystems

Raveen Ravinesh Goundar

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Graduate School of Science

Kochi University, Japan

March 2008

Copyright © 2008 by author
All rights reserved.

To my family

Contents

List of Tables

List of Algorithms

Preface

In todays world, information security is of the greatest importance in which communication over open networks and storage of data in digital form plays a key role in daily life. The science of cryptography provides efficient tools to secure information. In [16], cryptography is defined as "*the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity identification, and data origin authentication*".

Due to their tamper resistance and mobility, cryptosystems are often implemented on constraint memory devices such as smart cards. In such cases modern cryptography that deals with elliptic curve cryptosystems is considered to be most appropriate. This was independently discovered by Neal Koblitz [25] and Victor Miller [34] in 1985. The major advantage of elliptic curve cryptosystem is that a small key of size 160-bits can provide comparable security level with other cryptographic standards such as RSA of 1024-bits. However, the basic operations involved are more complicated on elliptic curves, as a result it is essential to implement efficient operations required by elliptic curve cryptosystems. The basic operations involves scalar and multi-scalar multiplication, which are the main subject of this thesis.

The aim of this thesis is to show the importance of addition chains in applications to elliptic curve cryptosystems. This context is well portrayed in this thesis as a new strategy for addition-subtraction chain has been proposed and exploited for scalar and multi-scalar multiplication. In fact, this thesis discusses a novel approach for multi-exponentiation and elliptic curve multi scalar multiplication utilizing the

proposed strategy for addition-subtraction chain.

The contents in chapter 1 is a motivator for the thesis which includes an overview of the history of the cryptography, beginning with the first rumblings of subject in ancient Egypt four millennia ago and ending with a modern day needs and challenges. Chapters $2, 3$ and 4 includes the preliminary work of the thesis. Chapters $5, 6, 7$ and 8 introduces authors original work which includes four major results, respectively. Chapter 9 concludes this thesis including suggestions for future work. The following paragraphs gives an overview on chapters $2 - 8$.

Chapter 2 begins with a brief history on elliptic curves, following some concepts on arithmetics of elliptic curves. This includes definition, group law, etc. Later sections includes some cryptographic aspects of elliptic curves.

Chapter 3 reviews some algorithms for computation of scalar and multi-scalar multiplication. These includes the conventional binary method, Shamir method and Interleave method.

Chapter 4 deals with addition chains and exponentiation methods. It includes definitions of different types of addition chains and discusses some conventional exponentiation techniques.

Chapter 5 proposes a new strategy to find doubling-free short addition-subtraction chain for an arbitrary integer by utilizing a precise golden ratio. It is termed as the golden ratio addition-subtraction chain method or GRASC method. It also includes the discussion on SPA attack resistance provided by the proposed method.

Chapter 6 exploits the result of chapter 5 to elliptic curve cryptosystems by proposing an efficient and secure elliptic curve scalar multiplication over odd prime fields. In fact, an explicit algorithm for GRASC method has been proposed. It is termed as golden ratio addition chain method or GRAC method. Later GRAC method is utilized to propose an elliptic curve scalar multiplication.

Chapter 7 offers a novel method for the computation of simultaneous multi-exponents in abelian groups. More specifically, given an abelian group G and two elements a,b in G, the algorithm aims at computing expressions such as $a^e b^f$ with as few as possible group operations. Such problems are important in public key meth-

ods and specifically in elliptic curve based methods, where addition is an expensive operation. This chapter offers an algorithm that is based on an addition chain, in fact it generalizes a method that was proposed in chapter 6.

Chapter 8 proposes a novel algorithm for simultaneous multi-scalar multiplication, by employing addition chains. More specifically, the result of chapter 7 has been modified to suit the general case and has been exploited for elliptic curve cryptosystems.

Acknowledgements The author sincerely thank his supervisors, Professor Shiota and Professor Toyonaga for facilitating discussions, critical comments, motivation and for their kind support throughout his doctor course. The author gratefully acknowledge the Japanese Government for offering scholarship to pursue a Ph.D program.

Raveen R. Goundar
Kochi University, Japan
March 2008

Chapter 1

Introduction to Cryptography

Cryptography has a long and a fascinating history since it was initially being practiced by the ancient Egyptians four millennia ago. An encyclopedic history of cryptography could be cited from [18] or for a brief overview one may refer to [36]. Historically, cryptography deals with methods of transmitting information in a confidential manner such that a third party (adversary) cannot read the information, even if the transmission is done through an insecure channel such as a public telephone line.

In todays point of view, definition of cryptography deals with designing of algorithms, protocols and systems for secure transfer of information. The following gives the cryptographic goals: (1) *confidentiality*, a service that provides information content only to the authorized recipients. (2) *data integrity*, a service which addresses the unauthorized alteration of data. To assure data integrity, one must have the ability to detect data manipulation by unauthorized parties. Data manipulation includes insertion, deletion, and substitution of the transferred information. (3) *authentication*, a service related to identification. This function both applies to entities and information itself. Two parties entering into a communication should identify each other. (4) *non-repudiation*, a service which prevents an entity from denying previous commitments or actions. In order to satisfy privacy, one needs to convert the information (message, plaintext) to some unintelligent language (cryptogram,

ciphertext, codeword) by using some secret data (cryptographic key), which is so called encryption operation. The other party receiving the ciphertext takes it back to a plaintext only by correct secret key, which is so called the decryption operation. Algorithms, protocols and systems satisfying cryptographic encryption-decryption operations are called *cryptosystems*. There are two types of cryptosystems to achieve secure transmission; secret-key cryptosystems and public-key cryptosystems.

The most oldest and by far the fastest type of cryptosystems is the secret-key cryptosystems also known as the symmetric-key cryptosystems. This involves sharing key between two communicating people where the secret key is used both for the encryption and decryption process. This is why it is called symmetric-key cryptosystems. The current standardized method of this type is the AES (Advanced Encryption Standard) symmetric-key cryptosystem. Almost all method of this type involves bit manipulation between the message (binary form) and the secret key. Decryption is achieve by reversing these manipulation hence the process is fast. The major drawback of symmetric-key cryptosystem involves sharing of key between two people beforehand in a secure way, and that key management is more tricky in a large network.

The most striking development in the history of cryptography came in 1976 when Diffie and Hellman published their article "New directions in cryptography" [12]. This article introduced the revolutionary concept of public-key cryptosystem and also provided a new and ingenious method for key exchange, the security of which is based on the intractability of the discrete logarithm problem. Hence, adversary will not be able to decrypt the encrypted message even it is known publicly. In fact, having the encryption public has many advantages such as enabling protocols for authentication, signature, etc. Even though public-key cryptosystem is slow compared to the symmetric-key cryptosystem, it is used as a complement of symmetric-key cryptosystem, either for signatures or authentication or for key exchange due to short messages being transmitted.

It remains a major problem to find suitable public-key cryptosystems. Many cryptosystems have been proposed, and many have been broken. The two types of

methods survived are RSA and Diffie-Hellman. The RSA variants cryptosystem are used widely. These are based on the asymmetrical fact that it is very efficient to create large prime numbers but intractable to factor it into two non trivial primes. Today's computer technologies are able to factor 700-bit numbers. The existence of subexponential algorithm explains the necessity of using 768 or 1024-bit prime numbers, in other words very large keys.

The case of Diffie-Hellman considers asymmetric cryptosystems based on the computational infeasibility of the discrete logarithm problem (DLP). That is, finding the exponent with respect to some pre-defined element (base) of an abelian group is computationally hard.

In the late eighties, Neal Koblitz and Victor Miller independently proposed a public-key cryptosystem using additive points on an elliptic curve. This emerged a new research area called *elliptic curve cryptography*. They exploited the discrete logarithm problem on a general elliptic curve that has no subexponential time solution to propose an *elliptic curve cryptosystem*. Such type of cryptosystem provides the same level of security with smaller key sizes compared to RSA requiring large key size. Hence, elliptic curve cryptosystems can be used in environments such as smart cards since ciphers need less memory and smaller processor requirements. It was not until the late 1990's when elliptic curve systems started receiving commercial acceptance due to accredited standards organization specified elliptic curve protocols, and private companies included these protocols in their security products. An abundance of research has been published on the security and efficient implementation of elliptic curve cryptography. The computations involved for the basic operations on elliptic curve are more complicated, therefore it is crucial to implement efficient operations required by the cryptosystems. The efficiency of most elliptic curve cryptosystems depends on the computations of scalar or multi-scalar multiplication operations. This is one of the active research area in elliptic curve cryptography.

As we begin this new millennium, the effects of cryptography on our daily lives will only increase since we are in the midst of a revolution in information process-

ing and telecommunications. To ever increasing depths, our lives are impacted on the daily basis by interactions that requires sending of digital messages through cyberspace. This may involve the electronic transfer of digital dollars, the sending of personal electronic messages, or the sending of military secrets. One common requirement to all these types of message-sending is the need to keep these messages secret, and ensure that nobody tampers with the message. Hence, the importance of cryptography to our information-based society will only deepen in future time. It is essential that we are equipped with the knowledge to understand and deal more effectively with the new reality. Due to the improvements and inventions of new technologies that may cause danger to break secret codes, there is a need for a continued research in cryptography to compensate for information security with this changing modern world.

Chapter 2

Elliptic Curve Cryptography

Over the past few decades, elliptic curves have been playing an increasingly important role both in number theory and related fields such as cryptography. In mid eighties, Neal Koblitz [25] and Victor Miller [34], independently discovered the use of elliptic curves in cryptosystems and hence elliptic curve techniques were developed for factorization and primality testing. We abbreviate the term elliptic curve cryptosystem in what follows to ECC for the sake of convenience.

In the 1980's and 1990's, elliptic curve played an important role in the proof's of Fermat's Last Theorem. In 1992, Koyama et. al [28] proposed an analogue of RSA by using a special class of elliptic curves over the ring $\mathbb{Z}/n\mathbb{Z}$ where $n \in \mathbb{N}$ is composite. In 1993 Demytko [11] presented another analogue of RSA, in this case where there is minimal restriction on the types of elliptic curves to be used. In 1997, Vanstone and Zuccherato [46] developed a new cryptosystem based on elliptic curves over $\mathbb{Z}/n\mathbb{Z}$ in which the message is contained in the exponent instead of the group element. The security of all of the above RSA analogues is based upon the presumed difficulty of factoring. In 1997, Anshel and Goldfeld [1] presented an explicit construction of a pseudorandom number generator arising from an elliptic curve, which can be effectively computed at low computational cost. They introduced a new intractable problem, that leads to a new class of one-way functions based on the arithmetic theory of zeta functions, and against which there is currently no known attack.

There are several fine texts on arithmetic and cryptographic aspects of elliptic curves in the literature that could be consulted, such as [5, 24, 25, 48, 19, 42, 43] .

The main purpose of this chapter is to introduce the basic concepts of elliptic curves over prime fields. This has importance in the remaining chapters. The detail work on arithmetics on elliptic curves could be found in [32, 36, 48]. First, we formally define the notion of elliptic curves.

2.1 Elliptic Curve Arithmetic

We shall consider finite fields $\mathbb{F}_p$ in order to define elliptic curves for cryptographic purposes.

Definition 2.1.1. *Let p be a prime number, and let $\mathbb{F}_p$ denote the field of integers modulo p with characteristic not equal to 2 or 3. An elliptic curve E defined over $\mathbb{F}_p$ is given by an equation of the form*

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

where $a, b \in \mathbb{F}_p$ satisfy $4a^3 + 27b^2 \neq 0$.

The set of solutions $(x, y) \in \mathbb{F}_p$ to the equation (2.1) together with a point $\mathcal{O}$, called the point of infinity, is denoted by $E(\mathbb{F}_p)$, called the set of $\mathbb{F}_p$-rational points on E. The value $\triangle(E) = -16(4a^3 + 27b^2)$ is called the discriminant of the elliptic curve E.

If the points on the curve are represented using affine coordinates, as $P = (x, y)$, both the point addition and point doubling involve an expensive field inversion (to compute the slope of the chord of the tangent). To avoid these inversion, several projective systems of coordinates have been proposed in literature [3]. The major ones include the, affine coordinates system ($\mathcal{A}$), projective coordinates system ($\mathcal{P}$), Jacobian coordinates system ($\mathcal{J}$), Chudnovsky Jacobian coordinates system ($\mathcal{J}^C$) and modified Jacobian coordinates system ($\mathcal{J}^m$).

In further chapters, mixed coordinates system are considered since it has lower computational cost compared to other coordinate systems as proposed by Cohen et. al [9] . The notations $[i], [s]$ and $[m]$ denotes the cost of one inversion, one squaring and one multiplication, respectively. The cost of field additions is negligible. Generally, it is assumed $[s] = 0.8[m]$ for curves over odd prime field [14].

2.1.1 Group Law

Suppose P and Q are points on an elliptic curve E. Then following states the definition of point inversion, point addition and point doubling.

Definition 2.1.2. ***Point Inversion*** *Negative of a point represents the point inversion, which means the reflection of the point about the x-axis.*

Definition 2.1.3. ***Point Addition (ECADD)*** *We define $P + Q$. If $P \neq \mathcal{O}$ and $P \neq \pm Q$, then there must be the third point R on E, uniquely determined as the intersection point of the line through P and Q. Taking the reflection of R about the x-axis gives the result $P + Q$. Hence, $P + Q = -R$.*

Definition 2.1.4. ***Point Doubling (ECDBL)*** *We define $2P$. We assume that $P = Q$ and $P \neq -Q$. Then to form $P + Q = 2P$, we take the tangent line at P, which gives rise to a third point $R = (x_3, y_3)$, uniquely determined as the intersection point of E with that tangent line. Then the reflection $-R$ of R about the x-axis is what we define as $P + P = 2P = -R$. Thus, $-R$ is the other point of intersection of E with the line $x = x_3$, which is also the intersection of the line containing R and $\mathcal{O}$ with E.*

Let E be an elliptic curve defined by $y^2 = x^3 + Ax + B$. Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be points on E with $P_1, P_2 \neq \mathcal{O}$. Define $P_1 + P_2 = (x_3, y_3)$ as follows:

1. If $x_1 \neq x_2$, then $x_3 = m^2 - x_1 - x_2$, $y_3 = m(x_1 - x_3) - y_1$, where $m = \frac{y_2 - y_1}{x_2 - x_1}$.

2. $x_1 = x_2$ but $y_1 \neq y_2$, then $P_1 + P_2 = \mathcal{O}$.

3. If $P_1 = P_2$ and $y_1 \neq 0$, then $x_3 = m^2 - 2x_1$, $y_3 = m(x_1 - x_3) - y_1$, where $m = \frac{3x_1^2 + A}{2y_1}$.

4. If $P_1 = P_2$ and $y_1 = 0$, then $P_1 + P_2 = \mathcal{O}$.

5. $P + \mathcal{O} = P$ for all points P on E.

Note that when P_1 and P_2 have coordinates in a field $\mathbb{F}_p$ that contains A and B, then $P_1 + P_2$ also has coordinates in $\mathbb{F}_p$. Therefore $E(\mathbb{F}_p)$ is closed under the above addition points.

Theorem 2.1.1. *The addition points on an elliptic curve E satisfies the following properties:*

1. *commutativity, since $P_1 + P_2 = P_2 + P_1$ for all P_1, P_2 on E.*

2. *existence of identity, since $P + \mathcal{O} = P$ for all points P on E.*

3. *existence of inverses, since given P on E, there exist P' on E with $P + P' = \mathcal{O}$. This point P' is usually denoted as $-P$.*

4. *associativity, since $(P_1 + P_2) + P_3 = P_1 + (P_2 + P_3)$ for all P_1, P_2, P_3 on E.*

In other words, the points on E form an additive abelian group with $\mathcal{O}$ as the identity element.

If P is a point on an elliptic curve and k is a positive integer, the kP denotes $P + P + \cdots + P$ (with k summands). If $k < 0$, then $kP = (-P) + (-P) + \cdots + (-P)$, with $|k|$ summands. To commute kP for a large integer k, it is inefficient to add P to itself repeatedly. It is faster to involve doublings. Note that the size of the points are continually reduced mod p since we are considering a finite field $\mathbb{F}_p$. Also note that the associative property allows computations irrespective of the order of the combined summands.

On the other hand, if we are working over a large finite field and are given points P and kP, it is very difficult to determine the value of the k. This is called an elliptic curve discrete logarithm problem (ECDLP) and is the basis for elliptic curve cryptosystems. That is, the security of elliptic curve cryptosystems depends upon the intractability of the following problem.

Definition 2.1.5. *If E is an elliptic curve over a field $\mathbb{F}_p$, then the elliptic curve discrete logarithm problem to base $Q \in E(\mathbb{F}_p)$ is the problem of finding an $x \in \mathbb{Z}$ (if one exists) such that $P = xQ$ for a given $P \in E(\mathbb{F}_p)$.*

Currently, the DLP in elliptic curve groups is several orders of magnitude and is more difficult then the DLP in multiplicative group of finite field (of similar size). Explicitly this means that for a suitably chosen elliptic curve E over $\mathbb{F}_p$, the DLP for the group of $E(\mathbb{F}_p)$ appears to be (given our current state of knowledge) of complexity exponential in the size $\lceil log_2 p \rceil$ of the field elements, whereas there exist subexponential algorithms in $\lceil log_2 p \rceil$ for the DLP in $\mathbb{F}_p^*$.

The lack of subexponential attacks on ECCs gives us the advantage of allowing smaller message units, as well as smaller amounts of processing time and electrical power. Moreover, a consensus is that a suitably chosen elliptic curve in a given ECC over a finite field of size approximately 160 bits ensures security equivalent to RSA with modulus of 1024 bits. There are two major drawbacks of ECC. Firstly, it involves the deep amount of mathematics which allows only a small group of qualified people to deal with the implementation of ECCs. Secondly, short span of time has been contributed for its researches up todate. For example, the appearance of MOV attack [31], showing that the DLP can be solved on supersingular curves is worrying to some, since other classes of elliptic curves have similarly fallen victim. If elliptic curves are to be used, then it is considered wise to use key sizes of 300 bits for even the most modest security requirements, and 500 bits for more sensitive communication. Such safety margins are recommended since the real threat to the presumed intractability of both discrete logs and factoring is unexpected with the new mathematical discoveries.

It is not certain what algorithm will be discovered in the near future. There are relatively very few people who have worked on the problem of discrete logs and integer factorization, which implies that the area has just not been examined exhaustively. Lastly, there is a ever-present threat of quantum computers, even though they are currently only a theoretical possibility.

2.2 Elliptic Curve Cryptosystems

This section reviews three cryptographic schemes which exploits the intractability of the ECDLP. Although they are originally designed for the group $(\mathbb{F}_p)^*$ and therefore exploit that the DLP is a complex mathematical problem, they can be adjusted to work with elliptic curves.

Throughout this section, it is assumed that the elliptic curves used in the cryptosystems are chosen in accordance with [33, 17] and that the order of the public points as well as the chosen parameters are 160-bit numbers.

2.2.1 Diffie-Hellman Key Exchange

Alice and Bob wants to agree on a common secret key that they can use for exchanging data via a symmetric encryption scheme such as DES or AES. One way to establish a secret key is the following method, due to Diffie and Hellman [12].

1. Alice and Bob agree on an elliptic curve E over a finite field $\mathbb{F}_p$ such that the discrete logarithm problem is hard in $E(\mathbb{F}_p)$. They also agree on a point $P \in E(\mathbb{F}_p)$ such that the subgroup generated by P has large order.

2. Alice chooses a secret integer a, computes $P_a = aP$, and sends P_a to Bob.

3. Bob chooses a secret integer b, computes $P_b = bP$, and sends P_b to Alice.

4. Alice computes $aP_b = abP$.

5. Bob computes $bP_a = baP$.

6. Alice and Bob use some publicly agreed on method to extract a key from abP. For example, either they could use the last 256 bits of the x-coordinate of abP as the key or they could compute the hash function at the x-coordinate.

The only information eavesdropper Eves has is the curve E, the finite field $\mathbb{F}_p$, and the points P, aP and bP. She therefore needs to compute abP using the given information, this is known as the *Diffie-Hellman problem*.

If Eve can solve discrete logs in $E(\mathbb{F}_p)$, then she can use P and aP to find a. Then she can compute $a(bP)$. However, it is not known whether there is some way to compute abP without first solving a discrete log problem.

This protocol can be used to establish a secure tunnel between two parties, e.g the SSH protocol uses this technique to exchange the secret key required for a symmetric scheme.

2.2.2 ElGamal Cryptosystem

The ElGamal Cryptosystem [13] is an extension of Diffie-Hellman key exchange protocol and its purpose is to encrypt and decrypt messages. It is described as follows.

Suppose Alice wants to send a message to Bob. First, Bob has to establish his public key. He chooses an elliptic curve E over a finite field $\mathbb{F}_p$ such that the discrete log problem is hard for $E(\mathbb{F}_p)$. He also chooses a point P on E. He chooses a secret integer b and computes $B = bP$. The elliptic curve E, the finite field $\mathbb{F}_p$, and the points P and B are Bob's public key. To send the message to Bob, Alice does the following:

1. Download Bob's public key.
2. Expresses her message as a point $M \in E(\mathbb{F}_p)$.
3. Chooses a secret random integer k and computes $M_1 = kP$.
4. Computes $M_2 = M + kB$.

5. Sends M_1, M_2 to Bob.

Bob decrypts by calculating $M = M_2 - bM_1$. Since

$$M_2 - bM_1 = (M + kB) - b(kP) = M + k(bP) - bkP = M\,.$$

The eavesdropper Eve knows Bob's public information P and B and points M_1 and M_2. If she can solve the ECDLP then she can retrieve b from P and B, which she can then use to decrypt the messages as $M_2 - bM_1$.

It is important for Alice to choose different random k for each encryption. Suppose Alice uses the same k for both M and M' then Eve can detect easily that $M = M'$. Hence, she can compute $M_2' - M_2 = M' - M$. Suppose M is revealed to the public day later. Then Eve can calculate $M' = M - M_2 + M_2'$. Therefore, knowledge of one plaintext M can help Eve to deduce another plaintext M' in such situation.

2.2.3 ElGamal Digital Signature

Suppose Alice wants to send a signed electronic message m to Bob. The naive way is to digitize the signature and append it to the message. In such situation, Eve may tamper with the signature by attaching a different message. Therefore, Alice must insert the signature in the message such that it can not be used again. However signature validation and authentication are required.

These are shown in ElGamal digital signature algorithm, which is classically based on multiplicative group of discrete logs. In fact, it applies to any finite group, hence we will consider it for elliptic curves.

First, Alice establishes a public key. She chooses an elliptic curve E over a finite field $\mathbb{F}_p$ such that the discrete log problem is hard for $E(\mathbb{F}_p)$. She also chooses a point $A \in E(\mathbb{F}_p)$ such that the order N of A is a large prime. Alice chooses a secret a and computes $B = aA$. Finally, she chooses a function $f : E(\mathbb{F}_p) \to \mathbb{Z}$ where $f(x, y) = x$ such that $0 \leq x \leq p$. Alice makes the following information public. That is $E, \mathbb{F}_p, f, A$ and B. She keeps a private.

To sign a message, Alice does the following.

1. Represents the document as an integer m. In the case, $m > N$, a larger curve should be chosen.
2. Chooses a random integer k with $\gcd(k, N) = 1$ and computes $R = kA$.
3. Computes $s \equiv k^{-1}(m - af(R)) \ (mod\ N)$.

Resulting the signed message as (m, R, s), where m, s are integers and R is a point on E.

To verify the message, Bob does the following.

1. Downloads the Alice's public information.
2. Computes $V_1 = f(R)B + sR$ and $V_2 = mA$.
3. If $V_1 = V_2$, he declares the signature valid.

The signature is valid if $V_1 = V_2$ since

$$V_1 = f(R)B + sR = f(R)aA + skA = f(R)aA + (m - af(R))A = mA = V_2\,.$$

The fact that $sk \equiv (m - af(R)) \ (mod\ N)$ implies $sk \equiv (m - af(R)) + zN$ for some integer z. Therefore,

$$skA = (m - af(R))A + zNA = (m - af(R))A + \infty = (m - af(R))A\,.$$

2.2.4 Elliptic Curve Digital Signature Algorithm

Elliptic curve digital signature algorithm (ECDSA) is variant on ElGamal signature scheme with some modifications. It is Digital Signature Standard [33, 17] used by the U.S. government. The ECDSA is based on the intractability of ECDLP. We describe its algorithm as follows.

Suppose Alice wants to sign a message m, which is an integer. Alice chooses an elliptic curve over a finite field $\mathbb{F}_p$ such that $\#E(\mathbb{F}_p) = fr$, where r is a large prime and f is a small integer. She chooses a base point G in $E(\mathbb{F}_p)$ of order r.

Finally, Alice chooses a secret integer a and computes $Q = aG$. The following information is made public by Alice. That is $\mathbb{F}_p, E, r, G, Q$. The following steps are involve for signing message m by Alice.

1. Choose a random integer k with $1 \leq k < r$ and computes $R = kG = (x, y)$.

2. Compute $s = k^{-1}(m + ax)\ (mod\ r)$.

Resulting the signed message as (m, R, s) .

The following steps are involve for verifying signature m by Bob.

1. Compute $u_1 = s^{-1}m\ (mod\ r)$ and $u_2 = s^{-1}x\ (mod\ r)$.

2. Compute $V = u_1G + u_2Q$.

3. Declare the signature valid if $V = R$.

If the message is signed correctly, then the following verification equation holds.

$$V = u_1G + u_2Q = s^{-1}mG + s^{-1}xQ = s^{-1}(mG + xaG) = kG = R\,.$$

The main difference in the ECDSA and the ElGamal system, is the verification procedure. In ElGamal system the verification process requires three computations of an integer times a point, whereas in ECDSA, only two such computations are required. These are the most expensive parts of the algorithm. In cases where many verifications are required, then the improved efficiency of the ECDSA and the ElGamal system are valuable.

Chapter 3

Scalar and Multi-Scalar Multiplication

The basic operation in most elliptic curve cryptosystems is a scalar multiplication kP and sum of scalar multiplications, that is

$$\sum_{j=1}^{n} k_j P_j \,,$$

where k_j are scalars and P_j are points on an elliptic curve for $j = 1, 2, \ldots, n$. This summation of scalar multiplications is called multi-scalar multiplication. For some of the cryptographic protocols, P_j is designated as a fixed point that generates a large, prime order subgroup of $E(\mathbb{F}_p)$, while for others P_j is an arbitrary point in such a subgroup.

In fact, multi-scalar multiplications are the most time consuming operations for elliptic curve cryptosystems where implementation are mainly on devices with constrain computational power and memory, therefore efficient operations are essential. Multi-scalar multiplication is required in many elliptic curve cryptosystems (ECC) such as in verification process of ElGamal digital signature, verification process of ECDSA, provable-secure digital signatures [39, 40], multi-party protocols [4] and protocols of Brands [7].

In most cases where multi-scalar multiplication is applied, the process is dominant in determining the overall efficiency. Hence, efficiency of multi-scalar multiplication is essential in elliptic curve cryptosystems. Conventional methods for computation of multi-scalar multiplication can be classified into two types. In methods of one type includes independent computation of the scalar multiples k_iG_i, followed by their addition. Such a method could be very expensive but in cases where some of the scalars are fixed then a comb method [29] combined with a window method could enhance the overall efficiency of the process. In the methods of the other type, the multi-scalar multiplication is computed in one stage, without separate computation of k_iG_i. This includes simultaneous methods such as Shamir [13] and Interleave [35] which utilizes binary representations for double-and-add algorithm.

This chapter reviews conventional algorithm for the computation of scalar multiplication. It also discusses some algorithms for computation of multi-scalar multiplication.

3.1 Scalar Multiplication

3.1.1 Binary Method

The simplest and far the oldest efficient method for scalar multiplication relies on the binary expansion of k. This is also known as *double-and-add* method. Algorithm 1

Algorithm 1 Scalar Multiplication: Binary Method

Input: A point $P \in E(\mathbb{F}_p)$, an n-bit integer $k = \sum_{j=0}^{\ell-1} d_j 2^j$, $d_j \in \{0, 1\}$.

Output: $Q = kP$.

1. $Q \leftarrow \mathcal{O}$
2. **for** $j = \ell - 1$ **down to** 0 **do**
3. $\quad Q \leftarrow 2Q$
4. $\quad$ **if** $k_j = 1$ **then**
5. $\quad\quad Q \leftarrow Q + P$
6. **return** Q

performs an ECADD operation each time for $d_j = 1$, hence the probability is 1/2 for ECADD operation. Therefore, on average, binary methods requires

$$nECDBL + n.\frac{1}{2}ECADD$$

operations to compute a scalar multiplication kP, where the scalar is represented in the binary representation.

3.2 Multi-Scalar Multiplication

The Shamir method [13] and Interleave method [35] are the two most efficient methods to compute multi-scalar multiplication due to simultaneous computations. The speed of these methods depends on the number of non zero entries or columns in the binary representations of the scalars, respectively. The following states some preliminaries for Interleave method and Shamir method.

A scalar d is a positive integer. A vector $(d_{n-1}, \ldots, d_0)$ is called a binary representation of d, if $d = \sum_{i=0}^{n-1} d_i 2^i$ and $d_i \in \mathcal{D}, \forall i = 0, \ldots, n-1$ where $\mathcal{D} = \{0, 1\}$. The following two algorithms of Interleave method and Shamir method computes $uP + vQ$, where u, v are the n bit scalars given in binary representation and $P, Q \in E(\mathbb{F}_p)$. Also, u_i and v_i denote the ith digit of u and v, respectively.

3.2.1 Interleave Method

Let the number of nonzero digits in a scalar u be called the Hamming weight and the expected density of it be called the average Hamming density (AHD(u)). As a result, Interleave method on average requires

$$n[AHD(u) + AHD(v)]ECADD + nECDBL$$

operations for the computation of $uP+vQ$, excluding the cost of the precomputation.

Algorithm 2 Multi-Scalar Multiplication: Interleave Method

Input: Points $P, Q \in E(\mathbb{F}_p)$, scalars u, v.

Output: $uP + vQ$.

1. $R_t \leftarrow tP, \forall t \in \mathcal{D}_1, t > 1$
2. $S_t \leftarrow tQ, \forall t \in \mathcal{D}_2, t > 1$
3. $X \leftarrow \mathcal{O}$
4. **for** $i = n - 1$ **down to** 0 **do**
5. $\quad X \leftarrow ECDLB(X)$
6. $\quad$ **if** $u_i \neq 0$ **then**
7. $\quad\quad X \leftarrow ECADD(X, R_{u_i})$
8. $\quad$ **if** $v_i \neq 0$ **then**
9. $\quad\quad X \leftarrow ECADD(X, R_{v_i})$
10. **return** X

3.2.2 Shamir Method

Let the number of nonzero columns in two scalars u, v be called the joint Hamming weight and the expected density of it be called average joint Hamming density (AJHD(u,v)). As a result, Shamir method on average requires

Algorithm 3 Multi-Scalar Multiplication: Shamir Method

Input: Points $P, Q \in E(\mathbb{F}_p)$, scalars u, v.

Output: $uP + vQ$.

1. $R_{t_1 t_2} \leftarrow t_1 P + t_2 Q, \forall t_1, t_2 \in \mathcal{D} \backslash \{0\}, t_1 > 0$
2. $R_{t0} \leftarrow tP, \forall t \in \mathcal{D}, t > 1$
3. $R_{0t} \leftarrow tQ, \forall t \in \mathcal{D}, t > 1$
4. $X \leftarrow \mathcal{O}$
5. **for** $i = n - 1$ **down to** 0 **do**
6. $\quad X \leftarrow ECDLB(X)$
7. $\quad$ **if** $(u_i, v_i) \neq (0, 0)$ **then**
8. $\quad\quad X \leftarrow ECADD(X, R_{u_i, v_i})$
9. **return** X

$$nAJHD(u,v)ECADD + nECDBL$$

operations to compute $uP + vQ$, excluding the cost of the precomputation.

Efficiency of these methods could be increased by deploying general base 2 representations of the scalars that provides less non zero entries or columns. The term "general" means, that $\mathcal{D}$ may be an arbitrary subset of $\mathbb{Z}$ and is not fixed to $\{0, 1\}$ as for the case of binary representation. Suppose $\mathcal{D} = \{0, \pm 1\}$, it is called the signed binary representation. If $\mathcal{D} = \{0, \pm 1, \ldots, \pm x\}$, it is called a signed representation. However, using general or signed representation, it requires the precomputation of some points. In fact, there is a trade-off between the number of points to precompute and the efficiency of a multi-scalar multiplication as discussed in [10].

Chapter 4

Addition Chains and Exponentiation

There are several situations in cryptography in which a number of exponentiations by a fixed exponents must be performed, such as RSA encryption and decryption, and ElGamal decryption [32]. In such cases, addition chain plays a vital role. It is used to efficiently compute an exponentiation or, more generally, several exponentiations. The purpose of an addition chain is to minimize the number of multiplications required for an exponentiation. For example, given an addition chain of length ℓ for the positive integer e then computation of g^e for any element g of an abelian group G, such that $g \neq 1$ requires exactly ℓ multiplications. Therefore, finding addition chains of smaller lengths are essential for efficient computations. However finding a shortest addition chain is known to be an NP-complete problem.

Scalar multiplication in elliptic curves is a special case of general problem of exponentiation in abelian groups. As such, it benefits from all the techniques available for the general problem, and the related short addition chain problem for integers. The latter is defined as follows. *Let k be a positive integer as the input. Starting from the integer 1, and computing at each step the sum of two previous results, what is the least number of steps required to reach k?*

Efficient algorithms for group exponentiation have received much attention in recent years in application to public key cryptography. However, the initial interest is from the ancient times. An excellent technical and historical account of exponentiation and the addition chain problem is given by Knuth [21] in chapter 4, who traces the problem back to 200 BC. A survey carried by Gordon [15] describes various fast methods, including some specialized to elliptic curve groups. Also various techniques and algorithms for exponentiations in the context of elliptic curve cryptography are described in [32].

There are certain properties of elliptic curve version which makes exponentiation algorithms more faster. Such as, the elliptic curve subtraction has virtually the same cost as addition, so the search space for fast algorithms can be expanded to addition-subtraction chains and signed binary representations. In this chapter some well known results and algorithms concerning addition chains and exponentiations are reviewed as preliminaries of this thesis.

4.1 Addition Chains

In this section, some classic definitions used in the study of addition chains and an overview on Fibonacci sequence are presented. More details could be cited from [3, 20].

Definition 4.1.1. *An addition chain computing an integer k is given by two sequences $v = (v_0, \ldots, v_\ell)$ and $w = (w_1, \ldots, w_\ell)$ such that $v_0 = 1, v_\ell = k, v_i = v_r + v_s$, for all $1 \leq i \leq \ell$ with respect to $w_i = (r, s)$ and $0 \leq r, s \leq i-1$. The length of the addition chain is ℓ.*

Definition 4.1.2. *An addition-subtraction chain is similar to an addition chain except that the coordinate $v_i = v_r + v_s$ is replaced by $v_i = v_r + v_s$ or $v_i = v_r - v_s$.*

Definition 4.1.3. *An Euclidean addition chain (EAC) is an addition chain which*

satisfies $v_1 = 1\,, v_2 = 2\,, v_3 = v_2 + v_1$ *and for all* $3 \leq i \leq \ell - 1$, *if* $v_i = v_{i-1} + v_j$ *for some* $j < i - 1\,$, *then* $v_{i+1} = v_i + v_{i-1}$ *or* $v_{i+1} = v_i + v_j$.

Definition 4.1.4. *The Fibonacci sequence is defined as* $F_n = F_{n-1} + F_{n-2}$ *for* $n \geq 2$ *where* $F_0 = 1$ *and* $F_1 = 1$.

The Fibonacci sequence has many properties [20, 47] we recall one here, by stating the following Binet's Formula.

Theorem 4.1.1. *Binet's Formula:*

$$F_n = \frac{\phi^n - (1-\phi)^n}{\sqrt{5}}\,, \quad \forall n \in \mathbb{N}\,,$$

where $\phi = \frac{1+\sqrt{5}}{2}$ *is the positive root of the real polynomial* $X^2 - X - 1\,$.

The following classical result is deduced from the above theorem.

$$\lim_{n\to\infty} \frac{F_n}{F_{n-1}} = \phi\,, \tag{4.1}$$

where ϕ is the golden ratio.

4.2 Exponentiation using Addition Chain

Once an addition chain for an integer k is found it is straight forward to deduce x^k.

Algorithm 4 Exponentiation using Addition Chain

Input: An element k of an abelian group G and an addition chain computing $k\,$.

Output: The element $x^k\,$.

1. $x_1 \leftarrow x$
2. **for** $i = 1$ **to** ℓ **do**
3. $\quad x_i \leftarrow x_r \times x_s$
4. **return** x_ℓ

Chapter 5

New Strategy for Addition Chains

The efficiency and security of most elliptic curve cryptosystems are based on exponentiation. One such method could be the use of addition-subtraction chain. In this chapter, we propose a new strategy to find doubling-free (SPA resistant) short addition-subtraction chain for an arbitrary integer by utilizing a precise golden ratio. We term it as the golden ratio addition-subtraction chain method or GRASC method for the sake of convenience.

5.1 Introduction

Elliptic curve cryptography was proposed independently by Koblitz [25] and Miller [34] in 1985. It facilitates two parties to generate a secret key for communication across an insecure channel. The most fascinating feature of elliptic curve cryptography is that it utilizes much smaller key of size 160 bits to provide same level of security as other cryptographic standards such as RSA of 1024 bits. The strength of elliptic curve cryptosystems (ECC) is based on the infeasibility of the elliptic curve discrete logarithm problem. The major building block of most ECC is computation of the form kP known as the scalar multiplication, where k is a positive integer (a secret scalar) and P is a point on an elliptic curve defined over a finite field. All exponentiation techniques for multiplicative groups could be easily adjusted for additive

groups as well. Therefore, efficient and secure exponentiation methods are vital for scalar multiplication in implementation of ECC. There are many exponentiation methods proposed in the literature [3], such as double-and-add using binary scalar representation, triple-and-add using ternary scalar representation, use of addition chains etc. Most of these methods are dependable on the secret scalar or exponent, hence through side-channel analysis it leaks secret information. This is known as side-channel attack, recently discovered by Kocher et al. [26]. In one type of side-channel attack, known as simple power analysis (SPA), the attackers uses the power consumption to monitor each operations. Due to each type of operations having difference in power consumption, helps attackers to retrieve secret scalar. One way to overcome SPA attack is the use doubling-free addition(-subtraction) chain [8]. It results in a fixed sequence of operations, hence attackers could not detect any information through SPA. Note that inversion of a point in elliptic curve cryptography is cost negligible, hence addition and subtraction operation involves same power consumption. The addition-subtraction chain involving one doubling, that is 2, is unaffected by SPA attack since the computation of $2P$ is required in almost all ECC.

Our contribution will deal with the construction of doubling-free short addition-subtraction chain, enhancing efficiency and security in applications to ECC. Although finding a minimal addition chain is known to be an NP-complete problem [32], we propose a reasonably short addition-subtraction chain involving mostly of Fibonacci pattern. In fact, there has been many strategies proposed in the literature [6, 21, 30, 49] to obtain a sub-optimal chain. Our proposed method is dependable on two parameters, therefore it is considered not to be sub-optimal. We experimentally select suitable parameters for a 160 bit integer for the best results. We will make comparison of our proposed method with the other doubling-free addition chain methods known in the literature.

The rest of the sections are organized as follows. In section 5.2, we propose a new strategy (GRASC) for finding moderately short addition-subtraction chain. In section 5.3, we discuss our experimental results and make comparisons with the previous methods in the literature. Section 5.4 concludes this chapter mentioning

some further work.

5.2 Proposed Strategy (GRASC Method)

In this section, we discuss our propose strategy for finding an efficient doubling-free short addition-subtraction chain by utilizing a precise golden ratio. We term it as the golden ratio addition-subtraction chain method or GRASC method in short.

The last term, v_ℓ in a doubling-free addition chain is maximal if the following condition holds:

$$v_i = v_{i-1} + v_{i-2} \quad \text{for} \quad i = 2, 3, \ldots, \ell\,, \tag{5.1}$$

that is, when v is a Fibonacci sequence. Thus, our aim is to maintain a Fibonacci pattern. Our strategy creates chain starting from the last term. As deduced from equation (4.1), the ratio between the two large succeeding terms in a Fibonacci sequence, maintains the value near ϕ, therefore we consider multiplying the last term by an inverse of the golden ratio to get its preceding term. That is

$$v_{i-1} \approx v_i \times \phi^{-1}\,, \tag{5.2}$$

where $\phi^{-1} = (\frac{-1+\sqrt{5}}{2})$ is the inverse of the golden ratio. Then we create the Fibonacci sequence by subtracting the $(i-1)$-th term from the i-th term to get the $(i-2)$-th term. Each time we check the ratio between two succeeding terms to be near golden ratio value, if not, then we take few actions and repeat the previous process to create Fibonacci sequence. We continue with this process until we reach some prescribed lower bound, a small term, thereafter we can efficiently find doubling-free short addition chain. We join this short addition chain to the previous chain to complete the overall chain. Experimentally, we found that a 160 bit Fibonacci integer has minimal chain length of 231, whereas GRASC method gives an average chain of length 258 for an arbitrary integer of 160 bit. In fact, GRASC methods gives minimal chain for Fibonacci numbers or equivalently if (5.2) holds but we do not guarantee for the case of non Fibonacci arbitrary integers to be minimal. We believe that GRASC

method gives moderately short addition-subtraction chain since it utilizes mostly of the Fibonacci pattern. The following describes the GRASC method in detail.

We consider making chain starting from the last term, which is the input k. Let u_i denote the reverse of v_i, that is, $u_i = v_{\ell-i}$. To maintain (5.2), we let

$$\begin{aligned} u_0 &= k\,, \\ u_1 &= [u_0 \times \phi^{-1}]\,, \\ u_i &= u_{i-2} - u_{i-1} \qquad \text{for} \qquad i = 2, 3, \ldots \end{aligned} \tag{5.3}$$

If continued with the procedure (5.3), we will not be able to achieve the best result, since u_i will exponentially deviate from $(u_{i-1} \times \phi^{-1})$ as i increases. In order to overcome this problem, we introduce the parameter MAXIMALGAP such that the above procedure (5.3) terminates whenever

$$|u_i - (u_{i-1} \times \phi^{-1})| > \text{ MAXIMALGAP or } u_i \leq \frac{u_{i-1}}{2}\,.$$

In such case, we define, new u_i to be the nearest integer of $(u_{i-1} \times \phi^{-1})$. We resume the procedure (5.3) with u_{i-1} and new u_i as the initial terms. Note that it is necessary to include old u_i in the chain between u_{i-1} and new u_i. As a consequence, we have a gap, $g_j = |\text{old } u_i - \text{new } u_i|$ which we include in the storage. Also note that a subtraction is involve, whenever old $u_i <$ new u_i. We introduce another parameter, LOWERBOUND, to terminate procedure (5.3) permanently. That is, when $u_i \leq$ LOWERBOUND. Note that the storage initially consists of $1, 2$, and 3. Later we have included all the g_j's in the storage. Once the execution of procedure (5.3) stops permanently, we include the last two u_i's of the chain in the storage. Thus, using the storage, we randomly find a short addition chain by avoiding the use of doubling, except for computation of $u_{\ell-1} = 2$. Finally, we join this chain to the third last u_i of the previous chain resulting in a moderately short addition-subtraction chain for the given input k. The storage capacity is dependent on the experimentally selected values for the two parameters. Note that in step 16 of Algorithm 5, if $u_i < u_{i+1}$ then g_j's will involve subtraction during exponentiation.

Algorithm 5 Golden Ratio Addition-Subtraction Chain Method

Input: An integer k, MAXIMALGAP and LOWERBOUND.

Output: Short addition-subtraction chain for k.

1. $\phi^{-1} \leftarrow \frac{-1+\sqrt{5}}{2}$
2. $u_0 \leftarrow k$
3. $u_1 \leftarrow [k \times \phi^{-1}]$
4. $u_2 \leftarrow u_0 - u_1$
5. $v = \{u_0, u_1, u_2\}$
6. $S = \{1, 2, 3\}$
7. $i \leftarrow 2$
8. $j \leftarrow 1$
9. **while** $u_i >$ LOWERBOUND **do**
10. **if** $|u_i - (u_{i-1} \times \phi^{-1})| >$ MAXIMALGAP or $u_i \leq \frac{u_{i-1}}{2}$ **then**
11. $i \leftarrow i + 1$
12. $u_i \leftarrow [u_{i-2} \times \phi^{-1}]$
13. $v \leftarrow v \cup \{u_i\}$
14. $u_{i+1} \leftarrow u_{i-2} - u_i$
15. $v \leftarrow v \cup \{u_{i+1}\}$
16. $g_j \leftarrow |u_i - u_{i+1}|$
17. $S \leftarrow S \cup \{g_j\}$
18. $j \leftarrow j + 1$
19. $i \leftarrow i + 1$
20. **else**
21. $i \leftarrow i + 1$
22. $u_i \leftarrow u_{i-2} - u_{i-1}$
23. $v \leftarrow v \cup \{u_i\}$
24. $S \leftarrow S \cup \{u_i, u_{i-1}\}$
25. $w \leftarrow$ a short addition chain including all terms from S
26. **return** $w \cup v$

Example 5.1 Evaluate Algorithm 5 for the inputs $k = 131456$, LOWERBOUND= 10 and MAXIMALGAP= 6.

We begin by letting

$$\begin{aligned} u_0 &= k = 131456\,, \\ u_1 &= [u_0 \times \phi^{-1}] = 81244\,, \\ u_2 &= u_0 - u_1 = 50212\,, \\ u_3 &= u_1 - u_2 = 31032\,, \\ u_4 &= u_2 - u_3 = 19180\,, \\ u_5 &= u_3 - u_4 = 11852\,, \\ u_6 &= u_4 - u_5 = 7328\,, \\ u_7 &= u_5 - u_6 = 4524\,, \\ u_8 &= u_6 - u_7 = 2804\,, \end{aligned}$$

since u_8 exceeds the MAXIMALGAP, that is $|2804 - (4524 \times \phi^{-1})| > 6$, we let

$$u_9 = [u_7 \times \phi^{-1}] = 2796\,.$$

There is a gap, $g_1 = |2804 - 2796| = 8$, which we include in the storage. Let

$$\begin{aligned} u_{10} &= u_7 - u_9 = 1728\,, \\ u_{11} &= u_9 - u_{10} = 1068\,, \\ u_{12} &= u_{10} - u_{11} = 660\,, \\ u_{13} &= u_{11} - u_{12} = 408\,, \\ u_{14} &= u_{12} - u_{13} = 252\,, \end{aligned}$$

$$\begin{aligned} u_{15} &= u_{13} - u_{14} = 156\,, \\ u_{16} &= u_{14} - u_{15} = 96\,, \\ u_{17} &= u_{15} - u_{16} = 60\,, \\ u_{18} &= u_{16} - u_{17} = 36\,, \\ u_{19} &= u_{17} - u_{18} = 24\,, \\ u_{20} &= u_{18} - u_{19} = 12\,, \end{aligned}$$

since $u_{20} \leq \frac{u_{19}}{2}$, we let

$$u_{21} = [u_{19} \times \phi^{-1}] = 15.$$

There is a gap, $g_2 = |12 - 15| = 3$, which we include in the storage. Let

$$u_{22} = u_{19} - u_{21} = 9\,.$$

We stop the above continuous procedure at u_{21}, since u_{22} transcends the given LOWERBOUND= 10. Now, we consider the storage which consists of pre-numbers $1, 2, 3$ and additional gap numbers $g_1 = 8$ and $g_2 = 3$. Further, we include $u_{21} = 15$ and $u_{22} = 9$ in the storage. Hence, we have

$$\{1, 2, 3, 8, 3, 15, 9\}\,.$$

We exclude the repeated numbers and rearrange it as

$$\{1, 2, 3, 8, 9, 15\}\,.$$

We search for a doubling-free short addition chain including all numbers from the storage. We already have $1\,, 2$ and $3\,$. Further, we insert 5, so that $3 + 5 \rightarrow 8\,$. It

follows that $1+8 \rightarrow 9$, $5+9 \rightarrow 14$ and $1+14 \rightarrow 15$. Hence, this completes our chain utilizing all the storage elements. Thus, we attain the following doubling-free short addition chain.

$$1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 8 \rightarrow 9 \rightarrow 14 \rightarrow 15.$$

Finally, we join this chain to the previous chain at u_{20}, resulting in a complete chain for $k = 131456$ with length 28.

Next, we utilize the above chain to compute exponent $k = 131456$ starting from numeral 1. We denote v as the addition-subtraction chain, where $v_{\ell-i} = u_i$. It follows that

$$\begin{aligned}
v_0 = \quad & v_{28-28} = \quad u_{28} \quad = \quad 1\,, \\
v_1 = \quad & v_{28-27} = \quad u_{27} \quad = \quad v_0 + v_0 = 2\,, \\
v_2 = \quad & v_{28-26} = \quad u_{26} \quad = \quad v_0 + v_1 = 3\,, \\
v_3 = \quad & v_{28-25} = \quad u_{25} \quad = \quad v_1 + v_2 = 5\,, \\
v_4 = \quad & v_{28-24} = \quad u_{24} \quad = \quad v_2 + v_3 = 8\,, \\
v_5 = \quad & v_{28-23} = \quad u_{23} \quad = \quad v_0 + v_4 = 9\,, \\
v_6 = \quad & v_{28-22} = \quad u_{22} \quad = \quad v_3 + v_5 = 14\,, \\
v_7 = \quad & v_{28-21} = \quad u_{21} \quad = \quad v_0 + v_6 = 15\,, \\
v_8 = \quad & v_{28-20} = \quad u_{20} \quad = \quad -v_2 + v_7 = 12\,, \\
v_9 = \quad & v_{28-19} = \quad u_{19} \quad = \quad v_5 + v_7 = 24\,, \\
v_{10} = \quad & v_{28-18} = \quad u_{18} \quad = \quad v_8 + v_9 = 36\,, \\
v_{11} = \quad & v_{28-17} = \quad u_{17} \quad = \quad v_9 + v_{10} = 60\,, \\
v_{12} = \quad & v_{28-16} = \quad u_{16} \quad = \quad v_{10} + v_{11} = 96\,, \\
v_{13} = \quad & v_{28-15} = \quad u_{15} \quad = \quad v_{11} + v_{12} = 156\,, \\
v_{14} = \quad & v_{28-14} = \quad u_{14} \quad = \quad v_{12} + v_{13} = 252\,, \\
v_{15} = \quad & v_{28-13} = \quad u_{13} \quad = \quad v_{13} + v_{14} = 408\,, \\
v_{16} = \quad & v_{28-12} = \quad u_{12} \quad = \quad v_{14} + v_{15} = 660\,, \\
v_{17} = \quad & v_{28-11} = \quad u_{11} \quad = \quad v_{15} + v_{16} = 1068\,, \\
v_{18} = \quad & v_{28-10} = \quad u_{10} \quad = \quad v_{16} + v_{17} = 1728\,,
\end{aligned}$$

$$
\begin{aligned}
v_{19} = \; v_{28-9} = \; u_9 \; &= \; v_{17} + v_{18} = 2796\,, \\
v_{20} = \; v_{28-8} = \; u_8 \; &= \; v_4 + v_{19} = 2804\,, \\
v_{21} = \; v_{28-7} = \; u_7 \; &= \; v_{18} + v_{19} = 4524\,, \\
v_{22} = \; v_{28-6} = \; u_6 \; &= \; v_{20} + v_{21} = 7328\,, \\
v_{23} = \; v_{28-5} = \; u_5 \; &= \; v_{21} + v_{22} = 11852\,, \\
v_{24} = \; v_{28-4} = \; u_4 \; &= \; v_{22} + v_{23} = 19180\,, \\
v_{25} = \; v_{28-3} = \; u_3 \; &= \; v_{23} + v_{24} = 31032\,, \\
v_{26} = \; v_{28-2} = \; u_2 \; &= \; v_{24} + v_{25} = 50212\,, \\
v_{27} = \; v_{28-1} = \; u_1 \; &= \; v_{25} + v_{26} = 81244\,, \\
v_{28} = \; v_{28-0} = \; u_0 \; &= \; v_{26} + v_{27} = 131456\,.
\end{aligned}
$$

5.3 Experimental Results and Discussion

In this section, we discuss our results and make comparisons with other doubling-free addition(-subtraction) chain methods in the literature [38].

We carried out an experiment to analyze the GRASC method using a python programming language on 1.66 GHz Intel Core Duo processor. We randomly selected 10000 integers k of 160 bits and set the searching ranges of the parameters of LOWERBOUND to be between 5 to 23 and the MAXIMALGAP to be between 5 to 15. It took 209 trials to obtain chains of lengths between 253 to 261 and on an average it took about 3 seconds to find each chain, as shown in Table 5.1. Whereas according to Meloni [38], in the case of EAC method, a 160-bit integer k will require testing of more than 45000 g's to find a chain of length 270, where g is randomly selected in the range $1 \leq g \leq k$. Hence, the best case for a EAC was found to be of length 320.

Our experimental result on Table 5.1 shows that chains of length between 257 to 259 could be found most efficiently using GRASC method. The best case was found to be chain of length 258. Note that GRASC method includes maximum of 26 points in the storage, see appendix for details. These points are discarded after being used during the scalar multiplication process. The data on Table 5.2 shows

Table 5.1: The distribution of chains based on GRASC method for 160 bit integers k.

GRASC length (ℓ)	# inputs k
261	15
260	389
259	2165
258	3610
257	2555
256	1003
255	219
254	37
253	7

Table 5.2: The average length of doubling-free addition chain for 160 bit integers

Method	Chain length
Fibonacci-and-add [38]	358
Signed Fib-and-add [38]	322
Window Fib-and-add [38]	292
EAC [38]	320
GRASC	258

that GRASC method has attained $28\%, 20\%, 12\%$ and 19% reduced in average chain length compared to Fibonacci-and-add, Signed Fib-and-add, Window Fib-and-add and EAC methods, respectively.

5.4 Conclusion

In this chapter we have proposed a new strategy to find an efficient doubling-free short addition-subtraction chain by utilizing a precise golden ratio. The empirical data shows that GRASC method has attained 12% to 28% reduced in average chain length compared to other doubling-free addition chain methods. Further work may include finding chains of much shorter lengths. One may consider giving explicit algorithms for GRASC method to suit applications in ECC. Also, reducing the storage content can make GRASC method more applicable to memory constraint devices due to ECC. The GRASC method has a unique way of creating chain, hence its real diligence is discussed in chapter 7 and chapter 8.

Chapter 6

Scalar Multiplication using GRAC Method

In this chapter we propose an efficient and secure (SPA resistant) elliptic curve scalar multiplication algorithm over odd prime fields. For this purpose, we propose an explicit algorithm for GRASC method discussed in chapter 5. We term it as golden ratio addition chain method or GRAC method for the sake of convenience.

6.1 Introduction

Elliptic curve cryptography was proposed independently in 1985 by Neal Koblitz [25] and Victor Miller [34]. Since then it is widely accepted due to its fascinating feature of having smaller key length of 160-bit in elliptic curve cryptosystems (ECC) to provide same level of security as for RSA with a 1024-bit of key length. An extensive amount of research has been dedicated to securing and accelerating its implementations. The overall efficiency of most ECC are dominated by computations of the form kP which is known as a scalar multiplication, where P is an elliptic curve point, and k is an arbitrary integer, which plays a role of a secret scalar. Hence, an efficient and secure scalar multiplications are essential in ECC. Most of the scalar

multiplication methods utilizes multiple operations such as double-and-add, triple-and-add etc. These operations when dependable on secret scalar, are susceptible to leak secret information through simple power analysis (SPA), which is one type of side-channel attack discovered by Kocher et al. [26]. The SPA monitors the power consumption of a single execution during scalar multiplication. The fact that different operations has different power consumption, helps attackers to retrieve secret data especially when the operations are dependable on secret scalars. Thus, in order to resist SPA attack, scalar multiplication should be implemented using fixed sequence of operations [8]. One of the solution could be the use of doubling-free addition chain for scalar multiplication. Note that one doubling computation involved in the chain is ineffective for SPA attack, since the computation of $2P$ is a necessary computation in almost all scalar multiplication.

In this chapter, we propose an explicit algorithm for the GRASC method in chapter 5 to make it compatible in presenting scalar multiplication algorithm. We term it as the golden ratio addition chain method or GRAC method for sake of our convenience. Note that the subtraction involved in the GRASC method has been excluded in our proposed algorithm (GRAC method), thus facilitating us in presenting an elegant scalar multiplication algorithm. Instead of using subtraction, we consider taking inversion of a point which is cost negligible in elliptic curve cryptography. This results in a series of addition operation during scalar multiplication, therefore resistance to SPA attack.

The rest of the sections are organized as follows. In section 6.2, we propose an explicit algorithm for GRASC method, term it as GRAC method. In section 6.3, we exploit GRAC method by proposing a scalar multiplication algorithm. In section 6.4, we discuss and compare our results with the previous methods known in the literature. Section 6.5 concludes this chapter mentioning some future work.

6.2 Proposed Explicit Algorithm

Here, we propose an explicit algorithm for the GRASC method and rename it as golden ratio addition chain method or GRAC method in short. This explicit algorithm will facilitate in proposing a scalar multiplication algorithm in the next section.

Algorithm 6 Golden Ratio Addition Chain Method

Input: A positive integer k, MAXIMALGAP and LOWERBOUND.

Output: $m = \{e_1, \ldots, e_{n+1}\}_{GRAC}$, $S = \{1, 2, 3, g_1, \ldots, g_{max}, u_{i-2}, u_{i-1}, u_i\}$, SAC.

1. $\phi^{-1} \leftarrow \frac{-1+\sqrt{5}}{2}$
2. $u_0 \leftarrow k$
3. $u_1 \leftarrow [u_0 \times \phi^{-1}]$
4. $u_2 \leftarrow u_0 - u_1$
5. $m = \{0, 0\}$
6. $S = \{1, 2, 3\}$
7. $i \leftarrow 2$
8. $j \leftarrow 1$
9. **while** $u_i >$ LOWERBOUND **do**
10. $\quad e_i \leftarrow 0$
11. $\quad$ **if** $|u_i - (u_{i-1} \times \phi^{-1})| >$ MAXIMALGAP or $u_i \leq \frac{u_{i-1}}{2}$ **then**
12. $\quad\quad u_{i+1} \leftarrow [u_{i-1} \times \phi^{-1}]$
13. $\quad\quad e_i \leftarrow 1$, $e_{i-1} \leftarrow 2$, $e_{i+1} \leftarrow 0$
14. $\quad\quad m \leftarrow m \cup \{e_i, e_{i-1}, e_{i+1}\}$
15. $\quad\quad g_j \leftarrow u_i - u_{i+1}$
16. $\quad\quad S \leftarrow S \cup \{g_j\}$
17. $\quad\quad j \leftarrow j + 1$
18. $\quad\quad u_{i+2} \leftarrow u_{i-1} - u_{i+1}$
19. $\quad\quad i \leftarrow i + 2$
20. $\quad$ **else**
21. $\quad\quad m \leftarrow m \cup \{e_i\}$

22. $i \leftarrow i+1$

23. $u_i \leftarrow u_{i-2} - u_{i-1}$

24. $S \leftarrow S \cup \{u_i, u_{i-1}, u_{i-2}\}$

25. $max \leftarrow j$

26. $n \leftarrow i-1$

27. $m \leftarrow$ reverse the arrangements in m and rename the elements in increasing order starting with numeral 1 to $n+1$

28. $SAC \leftarrow$ a short addition chain using absolute values of g_j's and other terms in S

29. **return** $m = \{e_1, \ldots, e_{n+1}\}_{GRAC}$, $S = \{1, 2, 3, g_1, \ldots, g_{max}, u_{i-2}, u_{i-1}, u_i\}$, SAC

Example 6.1 Evaluate Algorithm 6 for an input $k = 207062$, LOWERBOUND= 5 and MAXIMALGAP= 6.

First, we will find the GRAC representation m, during which we will obtain the elements for the storage S. Later, we will use all the storage elements to search for a short addition chain.
We begin by letting,

$$\begin{aligned}
u_0 &= k = 207062\,, & e_0 &= 0\\
u_1 &= [u_0 \times \phi^{-1}] = 127971\,, & e_1 &= 0\\
u_2 &= u_0 - u_1 = 79091\,, & e_2 &= 0\\
u_3 &= u_1 - u_2 = 48880\,, & e_3 &= 0\\
u_4 &= u_2 - u_3 = 30211\,, & e_4 &= 0\\
u_5 &= u_3 - u_4 = 18669\,, & e_5 &= 0\\
u_6 &= u_4 - u_5 = 11542\,, & e_6 &= 0\\
u_7 &= u_5 - u_6 = 7127\,, & e_7 &= 2\\
u_8 &= u_6 - u_7 = 4415\,, & e_8 &= 1
\end{aligned}$$

since u_8 exceeds the MAXIMALGAP, that is $|4415 - (7127 \times \phi^{-1})| > 6$, we let

$$u_9 = [u_7 \times \phi^{-1}] = 4405\,. \qquad e_9 = 0$$

There exist a gap, $g_1 = 4415 - 4405 = 10$, which we include in the storage. Let

$$\begin{aligned}
u_{10} &= u_7 - u_9 = 2722\,, & e_{10} &= 0\\
u_{11} &= u_9 - u_{10} = 1683\,, & e_{11} &= 0\\
u_{12} &= u_{10} - u_{11} = 1039\,, & e_{12} &= 0\\
u_{13} &= u_{11} - u_{12} = 644\,, & e_{13} &= 0\\
u_{14} &= u_{12} - u_{13} = 395\,, & e_{14} &= 0\\
u_{15} &= u_{13} - u_{14} = 249\,, & e_{15} &= 2\\
u_{16} &= u_{14} - u_{15} = 146\,, & e_{16} &= 1
\end{aligned}$$

since u_{16} exceeds MAXIMALGAP, that is $|146 - (249 \times \phi^{-1})| > 6$, we let

$$u_{17} = [u_{15} \times \phi^{-1}] = 154. \qquad e_{17} = 0$$

There exist a gap, $g_2 = 146 - 154 = -8$, which we include in the storage. Let

$$\begin{aligned}
u_{18} &= u_{15} - u_{17} = 95\,, & e_{18} &= 0\\
u_{19} &= u_{17} - u_{18} = 59\,, & e_{19} &= 0\\
u_{20} &= u_{18} - u_{19} = 36\,, & e_{20} &= 0\\
u_{21} &= u_{19} - u_{20} = 23\,, & e_{21} &= 0\\
u_{22} &= u_{20} - u_{21} = 13\,, & e_{22} &= 0\\
u_{23} &= u_{21} - u_{22} = 10\,. & e_{23} &= 0
\end{aligned}$$

We stop the above continuous procedure at u_{23}, since the next term,

$$u_{24} = u_{22} - u_{23} = 3\,,$$

transcends the given LOWERBOUND. We obtained the following storage.

$$S = \{1\,, 2\,, 3\,, g_1 = 10\,, g_2 = -8\,, u_{22} = 13\,, u_{23} = 10\,, u_{24} = 3\}\,.$$

We list $e_0, \ldots, e_{23}$ as elements of the set m.

$$m = \{0, 0, 0, 0, 0, 0, 0, 2, 1, 0, 0, 0, 0, 0, 0, 2, 1, 0, 0, 0, 0, 0, 0, 0\}\,.$$

Then we reverse the arrangements of the elements in the set m and rename it in increasing order starting from e_1. Thus, it results in the following GRAC representation.

$$m = \{0,0,0,0,0,0,0,1,2,0,0,0,0,0,0,0,1,2,0,0,0,0,0,0,0,0\}_{GRAC}\,.$$

Next, we randomly search for a doubling-free short addition chain (SAC) for absolute values of g_j's and other terms in the storage S. Hence, we consider the following terms from the storage S.

$$\{1,2,3,10,8,13,10,3\}\,.$$

Excluding the repeated numbers and rearranging results

$$\{1,2,3,8,10,13\}\,.$$

It follows that $3+10 \rightarrow 13$ and $2+8 \rightarrow 10$, We insert 5, so that $3+5 \rightarrow 8$. Next, we have $2+3 \rightarrow 5$. Remaining is $1, 2$ and 3, which are pre-included numbers in the storage. Hence, the following is the required short addition chain.

$$1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 8 \rightarrow 10 \rightarrow 13\,.$$

6.3 Application to Elliptic Curve Cryptosystems

In this section, we propose a SPA resistant scalar multiplication algorithm by utilizing the proposed GRAC method.

Algorithm 7 Scalar Multiplication using GRAC Method

Input: An integer k and $P \in E(\mathbb{F}_{q^k})$.

Output: kP.

Precomputation (GRAC method)

1. $m = \{e_1, \ldots, e_{n+1}\}_{GRAC}$
2. $S = \{1, 2, 3, g_1, \ldots, g_{max}, u_{i-2}, u_{i-1}, u_i\}$
3. SAC

Main loop

4. $G \leftarrow \emptyset$
5. **for** j=1 **to** max
6. $\quad G_j \leftarrow g_j P$ (using SAC)
7. $\quad G \leftarrow G \cup \{G_j\}$
8. $G \leftarrow$ reverse the arrangements in G and rename the elements in increasing order starting with numeral 1 to max
9. $T_0 \leftarrow u_i P$ (using SAC)
10. $T_1 \leftarrow u_{i-1} P$ (using SAC)
11. $T_2 \leftarrow u_{i-2} P$ (using SAC)
12. $j \leftarrow 1$
13. **for** $i = 2$ **to** n **do**
14. $\quad$ **if** $e_{i+1} = 0$ **then**
15. $\quad\quad T_{i+1} \leftarrow T_i + T_{i-1}$
16. $\quad$ **if** $e_{i+1} = 1$ **then**
17. $\quad\quad T_{i+1} \leftarrow T_i + G_j$
18. $\quad\quad j \leftarrow j + 1$
19. $\quad$ **if** $e_{i+1} = 2$ **then**
20. $\quad\quad T_{i+1} \leftarrow T_{i-1} + T_{i-2}$
21. **return** T_{n+1}

Hence, the required output $kP = T_{n+1}$. Note that Algorithm 7 involves storage of the preceding two points during scalar multiplication. Also, a temporary storage G containing max number of points g_j's, which are discarded during the scalar multiplication after being used, hence having less constraint on memory containing devices. The proposed algorithm uses a fixed sequence of addition operations therefore it is resistance to SPA attack.

Example 6.2 Compute $207062P$ using Algorithm 7.

Precomputation. (Example 6.1)

1. $m = \{\mathbf{0},\mathbf{0},0,0,0,0,0,1,2,0,0,0,0,0,0,1,2,0,0,0,0,0,0,0\}_{GRAC}$.

2. $S = \{1, 2, 3, g_1 = 10, g_2 = -8, u_{22} = 13, u_{23} = 10, u_{24} = 3\}$.

3. $SAC : 1 \to 2 \to 3 \to 5 \to 8 \to 10 \to 13$.

Evaluation stage.

For an input $P \in E(\mathbb{F}_{q^k})$, we utilize the above short addition chain (SAC) to compute the following: $2P, P + 2P = 3P, 2P + 3P = 5P, 3P + 5P = 8P$, taking it's inverse gives $-8P$, followed by $2P + 8P = 10P$ and $3P + 10P = 13P$. Hence, we have $G_1 = g_1P = 10P$ and $G_2 = g_2P = -8P$, which we store in G resulting as $G = \{10P, -8P\}$. Now, we reverse the arrangements in G and rename the terms in increasing order starting with numeral 1. Thus, $G = \{-8P, 10P\}$ where $G_1 = -8P$ and $G_2 = 10P$. Also, we have $T_0 = 3P$, $T_1 = 10P$, $T_2 = 13P$. Henceforth, we use the GRAC representation m, to compute T_i for $i = 2, \ldots, 23$ as follows.

$$
\begin{aligned}
&i = 2, \quad e_3 = 0, \quad T_3 = T_2 + T_1 = 13P + 10P = 23P,\\
&i = 3, \quad e_4 = 0, \quad T_4 = T_3 + T_2 = 23P + 13P = 36P,\\
&i = 4, \quad e_5 = 0, \quad T_5 = T_4 + T_3 = 36P + 23P = 59P,\\
&i = 5, \quad e_6 = 0, \quad T_6 = T_5 + T_4 = 59P + 36P = 95P,\\
&i = 6,, \quad e_7 = 0, \quad T_7 = T_6 + T_5 = 95P + 59P = 154P,\\
&i = 7, \quad e_8 = 1, \quad T_8 = T_7 + G_1 = 154P + (-8P) = 146P,\\
&i = 8, \quad e_9 = 2, \quad T_9 = T_7 + T_6 = 154P + 95P = 249P,\\
&i = 9, \quad e_{10} = 0, \quad T_{10} = T_9 + T_8 = 249P + 146P = 395P,\\
&i = 10, \quad e_{11} = 0, \quad T_{11} = T_{10} + T_9 = 395P + 249P = 644P,\\
&i = 11, \quad e_{12} = 0, \quad T_{12} = T_{11} + T_{10} = 644P + 395P = 1039P,\\
&i = 12, \quad e_{13} = 0, \quad T_{13} = T_{12} + T_{11} = 1039P + 644P = 1683P,\\
&i = 13, \quad e_{14} = 0, \quad T_{14} = T_{13} + T_{12} = 1683P + 1039P = 2722P,\\
&i = 14, \quad e_{15} = 0, \quad T_{15} = T_{14} + T_{13} = 2722P + 1683P = 4405P,
\end{aligned}
$$

$$
\begin{aligned}
&i = 15\,, \quad e_{16} = 1\,, \quad T_{16} = T_{15} + G_2 \;=\; 4405P + 10P \;=\; 4415P\,,\\
&i = 16\,, \quad e_{17} = 2\,, \quad T_{17} = T_{15} + T_{14} \;=\; 4405P + 2722P \;=\; 7127P\,,\\
&i = 17\,, \quad e_{18} = 0\,, \quad T_{18} = T_{17} + T_{16} \;=\; 7127P + 4415P \;=\; 11542P\,,\\
&i = 18\,, \quad e_{19} = 0\,, \quad T_{19} = T_{18} + T_{17} \;=\; 1154P + 7127P \;=\; 18669P\,,\\
&i = 19\,, \quad e_{20} = 0\,, \quad T_{20} = T_{19} + T_{18} \;=\; 18669P + 11542P \;=\; 30211P\,,\\
&i = 20\,, \quad e_{21} = 0\,, \quad T_{21} = T_{20} + T_{19} \;=\; 30211P + 18669P \;=\; 48880P\,,\\
&i = 21\,, \quad e_{22} = 0\,, \quad T_{22} = T_{21} + T_{20} \;=\; 48880P + 30211P \;=\; 79091P\,,\\
&i = 22\,, \quad e_{23} = 0\,, \quad T_{23} = T_{22} + T_{21} \;=\; 79091P + 48880P \;=\; 127971P\,,\\
&i = 23\,, \quad e_{24} = 0\,, \quad T_{24} = T_{23} + T_{22} \;=\; 127971P + 79091P \;=\; 207062P\,.
\end{aligned}
$$

6.4 Discussion

In this section, we discuss our results and make comparison with some previous methods in the literature.

In an experiment carried out in chapter 5 showed that the best case of GRASC method has chain of length 258, hence similar result holds for the GRAC method. Note that GRASC method is renamed as GRAC method. This is because we avoided the use of subtraction operation in the scalar multiplication; rather, we took inversion of a point whenever subtraction was involved. This facilitated us in proposing an elegant scalar multiplication algorithm. Note that GRAC method includes maximum of 26 points in the storage, as shown in the appendix. These points are discarded once being used during the scalar multiplication process.

The new point addition formula (NewADD) proposed by Meloni [38] is applicable to addition chains which involves Fibonacci type of additions. The GRAC method lacks the continuous Fibonacci pattern, hence we choose mixed coordinates to compute the cost for the proposed scalar multiplication algorithm. We have selected the best case of mixed coordinates [9] for addition and doubling operations to compute the cost of GRAC based algorithm as shown in Table 6.1. The total computational cost for GRAC based scalar multiplication algorithm involves $(\ell - 1)$ additions and one doubling, which is given by the following formula.

Table 6.1: Computational costs using mixed coordinates.

	Addition	Doubling
Operation	$\mathcal{A} + \mathcal{A} = \mathcal{J}^C$	$2\mathcal{A} = \mathcal{J}$
Costs	$5[m] + 3[s]$	$2[m] + 4[s]$

Table 6.2: Average cost of doubling-free scalar multiplication algorithms for 160 bit integers.

Algorithm	Coordinate	# [m]
Fibonacci-and-add [38]	NewADD	2311
Signed Fib-and-add [38]	NewADD	2088
Window Fib-and-add [38]	NewADD	1960
EAC-320 [38]	NewADD	2112
GRAC-258	Mixed	1907

$$\#[m] = (5[m] + 3[s])(\ell - 1) + (2[m] + 4[s])\,.$$

In Table 6.2, we compare the efficiency of GRAC based algorithm with other doubling-free algorithms proposed in [38]. Considering the best case of GRAC-258, it is evident from Table 6.2 that our proposed algorithm has outperformed Fibonacci-and-add by 18%, Signed Fib-and-add by 9%, Window Fib-and-add by 3% and EAC-320 by 10%.

6.5 Conclusion

In this chapter we have proposed a SPA resistant scalar multiplication algorithm. Thus, for our purpose we proposed an explicit algorithm (GRAC method) for the

GRASC method of chapter 5 to facilitate in presenting an elegant scalar multiplication algorithm. The proposed GRAC based scalar multiplication algorithm has preceded other scalar multiplication algorithm by 3% to 18%. Further work may include finding chains of much shorter lengths in order to improve the computational cost of the GRAC based scalar multiplication algorithm. Also, if one could reduce the storage capacity, then GRAC based algorithm could be more applicable to elliptic curve cryptosystems where implementations are considered on constraint memory devices such as smart cards.

Chapter 7

Multi-Exponentiation Method

The efficiency and security of most elliptic curve cryptosystems are based on multi-exponentiation, such as the verification process in elliptic curve digital signature algorithm. Simultaneous methods are considered to be the most efficient for multi-exponentiation. In this chapter, we propose a method to construct an addition chain for simultaneous multi-exponentiation, which has never been considered in the literature before. We discuss the strategy for the two dimension case, ${b_1}^e {b_2}^f$ and assume that such strategy could be generalized.

7.1 Introduction

Most elliptic curve cryptosystems [25, 34] require computations of several exponentials with distinct bases and distinct exponents which are termed as multi-exponentiation. The computation of each exponentiation separately and multiplying the result at the end could be very costly. In such case, simultaneous methods are considered to be more efficient for multi-exponentiation. The conventional simultaneous multi-exponentiation method are known as Shamir's trick [3] and Interleave method [35]. They are dependable on binary representation of exponents and utilizes square-and-multiply method, hence susceptible to simple power analysis attack (SPA), recently discovered by kocher [26]. It has been suggested by Byrne et al. [8]

that the use of doubling-free addition chain can provide resistance to SPA attack, hence we will consider it in our case. Note that addition chain involving one doubling, that is computation of 2, is unaffected by SPA attack since it is a necessary computation in almost all exponentiations involved in elliptic curve cryptosystems (ECCs). The use of addition chain for simultaneous multi-exponentiation has never been considered in the literature before. Conventionally, addition chains are not suitable for variable exponents and fixed bases but only suitable for fixed exponents and variable bases [32, 22]. In this paper, we will propose a novel method for simultaneous multi-exponentiation using addition chain.

We will extend the golden ratio addition-subtraction chain (GRASC) method of chapter 5 to suit the computation of multi-exponentiation. We will term it as simultaneous golden ratio addition chain method or SGRAC method in short. We will consider computation of two dimension case, that is ${b_1}^e {b_2}^f$ where b_1, b_2 are elements of an abelian group and $e, f \in \mathbb{Z}$. We assume that such strategy could be generalized.

The rest of the section is organized as follows. In section 7.2, we discuss the SGRAC method and state its algorithm. Later, we propose a multi-exponentiation algorithm based on SGRAC method. In section 7.3, we discuss our experimental results. Section 7.4 concludes this chapter mentioning some further work.

7.2 Simultaneous Golden Ratio Addition Chain Method

In the case of multi-exponentiation, such as the computation of $b_1^{e_0} b_2^{f_0}$, we shall follow the GRASC method with $u_i = e_0 x_1 + f_0 x_2$ as an input. We pair the exponents with variables x_1 and x_2 to distinguish between them. We select a suitable parameters, MAXIMALGAP and LOWERBOUND of the form $e_i x_1 + f_i x_2$. We compare the coefficients of each variables while checking the conditions given by the parameters. If either of the coefficients does not satisfy the condition, we take the required

actions as mentioned in GRASC method. Note that SGRAC method will include negative numbers in the storage, that is a gap g_j = old u_i − new u_i , is included in the storage contrary to GRASC method which only includes absolute values of g_j's. Hence, the exponentiation based on SGRAC method involves addition (multiplication) throughout the process. The following algorithm explains SGRAC method in detail.

Algorithm 8 Simultaneous Golden Ratio Addition Chain Method

Input: Exponents e_0 and f_0.

Output: $w = \{w_0, \ldots, w_\ell\}$, $G = \{g_1, \ldots, g_j, s_1, s_2\}$, SAC_{x_1} , SAC_{x_2} and m.

1. MAXIMALGAP ← choose a suitable parameter
2. LOWERBOUND ← choose a suitable parameter
3. $\phi^{-1} \leftarrow \frac{-1+\sqrt{5}}{2}$
4. $u_0 \leftarrow e_0x_1 + f_0x_2$ (x_1 and x_2 are variables)
5. $u_1 \leftarrow [u_0 \times \phi^{-1}]$
6. $u_2 \leftarrow u_0 - u_1$
7. $v = \{u_0, u_1, u_2\}$
8. $S = \{1x_1, 1x_2, 2x_1, 2x_2, 3x_1, 3x_2\}$
9. $m = \{0, 0\}$
10. $G = \emptyset$
11. $i \leftarrow 2$
12. $j \leftarrow 1$
13. **while** $u_i >$ LOWERBOUND **do**
14. **if** $|u_i - (u_{i-1} \times \phi^{-1})| >$ MAXIMALGAP or $u_i \leq \frac{u_{i-1}}{2}$ **then**
15. $m_i = 1\,, m_{i+1} = 0$
16. $m \leftarrow m \cup \{m_i\,, m_{i+1}\}$
17. $i \leftarrow i + 1$
18. $u_i \leftarrow [u_{i-2} \times \phi^{-1}]$
19. $v \leftarrow v \cup \{u_i\}$
20. $u_{i+1} \leftarrow u_{i-2} - u_i$
21. $v \leftarrow v \cup \{u_{i+1}\}$
22. $g_j \leftarrow (u_i - u_{i+1})$

23. $G \leftarrow G \cup \{g_j\}$
24. $j \leftarrow j + 1$
25. $i \leftarrow i + 1$
26. **else**
27. $m_i = 0$
28. $m \leftarrow m \cup \{\ m_i\}$
29. $i \leftarrow i + 1$
30. $u_i \leftarrow u_{i-2} - u_{i-1}$
31. $v \leftarrow v \cup \{u_i\}$
32. $S \leftarrow S \cup \{u_{i-1}, u_i\}$
33. $m_i = 2\,, m_{i+1} = 2$
34. $m \leftarrow m \cup \{m_i, m_{i+1}\}$
35. $max \leftarrow j$
36. **if** $e_i >$ LOWERBOUND **then**
37. $S \leftarrow S \cup \{f_{i-1}x_2\}$
38. $s_1 \leftarrow f_i x_2\,,\ s_2 \leftarrow (f_{i-1}x_2 - f_i x_2)$
39. $G \leftarrow G \cup \{s_1\,, s_2\}$
40. $SAC_{x_2} \leftarrow$ Short addition chain using absolute values of all x_2 terms in G and S
41. $j \leftarrow max + 1$
42. Repeat step 13 to step 32 for the x_1 terms only
43. $SAC_{x_1} \leftarrow$ Short addition chain using absolute values of all x_1 terms in G and S
44. $v \leftarrow v \cup SAC_{x_1}$
45. All m_i's are zero in SAC_{x_1}
46. **else**
47. $S \leftarrow S \cup \{e_{i-1}x_1\}$
48. $s_1 \leftarrow e_i x_1\,,\ s_2 \leftarrow (e_{i-1}x_1 - e_i x_1)$
49. $G \leftarrow G \cup \{s_1\,, s_2\}$
50. $SAC_{x_1} \leftarrow$ Short addition chain using absolute values of all x_1 terms in G and S
51. $j \leftarrow max + 1$
52. Repeat step 13 to step 32 for the x_2 terms only
53. $SAC_{x_2} \leftarrow$ Short addition chain using absolute values of all x_2 terms in G and S

54. $v \leftarrow v \cup SAC_{x_2}$
55. All m_i's are zero in SAC_{x_2}
56. $G \leftarrow$ reverse the elements g_j's in G and rename it in increasing order starting from numeral 1
57. $v \leftarrow$ reverse the arrangements of elements in v and rename in increasing
58. $m \leftarrow$ reverse the arrangements of elements in m and rename in increasing order starting from m_0 to m_ℓ
59. $w \leftarrow$ is a set containing index for the computation of each v_i in v
60. **return** $w = \{w_0, \ldots, w_\ell\}$, $G = \{g_1, \ldots, g_j, s_1, s_2\}$, SAC_{x_1} , SAC_{x_2} and m

Note that $u_i = e_i x_1 + f_i x_2$.

Example 7.1 Evaluate Algorithm 8 for the exponents $e_0 = 90287$ and $f_0 = 1835008$.

In order to distinguish between the two exponents, we introduce two variables x_1 and x_2. We experimentally select the suitable values for the LOWERBOUND and the MAXIMALGAP to be $10x_1 + 10x_2$ and $6x_1 + 6x_2$, respectively. Note that we will be comparing the coefficients of either x_1 or x_2, while checking the conditions using the two parameters. In the following, we will denote $u_i = e_i x_1 + f_i x_2$. We begin by letting

$$\begin{aligned}
u_0 &= e_0 x_1 + f_0 x_2 = 90287x_1 + 1835008x_2 , & m_0 = 0 \\
u_1 &= [u_0 \times \phi^{-1}] = 55800x_1 + 1134097x_2 , & m_1 = 0 \\
u_2 &= u_0 - u_1 = 34487x_1 + 700911x_2 , & m_2 = 0 \\
u_3 &= u_1 - u_2 = 21313x_1 + 433186x_2 , & m_3 = 0 \\
u_4 &= u_2 - u_3 = 13174x_1 + 267725x_2 , & m_4 = 0 \\
u_5 &= u_3 - u_4 = 8139x_1 + 165461x_2 , & m_5 = 0 \\
u_6 &= u_4 - u_5 = 5035x_1 + 102264x_2 , & m_6 = 0 \\
u_7 &= u_5 - u_6 = 3104x_1 + 63197x_2 , & m_7 = 1
\end{aligned}$$

since u_7 does not satisfy the condition $|u_7 - (u_6 \times \phi^{-1})| < 6x_1 + 6x_2$, therefore we utilize an inverse of a golden ratio to get the next u_i . That is

$$u_8 = [u_6 \times \phi^{-1}] = 3112x_1 + 63203x_2 . \qquad m_8 = 0$$

There exist a gap, $g_1 = u_7 - u_8 = -8x_1 - 6x_2$, which we include in the storage G. We continue by letting

$$\begin{aligned}
u_9 &= u_6 - u_8 = 1923x_1 + 39061x_2, & m_9 = 0\\
u_{10} &= u_8 - u_9 = 1189x_1 + 24142x_2, & m_{10} = 0\\
u_{11} &= u_9 - u_{10} = 734x_1 + 14919x_2, & m_{11} = 0\\
u_{12} &= u_{10} - u_{11} = 455x_1 + 9223x_2, & m_{12} = 0\\
u_{13} &= u_{11} - u_{12} = 279x_1 + 5696x_2, & m_{13} = 0\\
u_{14} &= u_{12} - u_{13} = 176x_1 + 3527x_2, & m_{14} = 1
\end{aligned}$$

since u_{14} does not satisfy the condition $|u_{14} - (u_{13} \times \phi^{-1})| < 6x_1 + 6x_2$, therefore we utilize an inverse of a golden ratio to get the next u_i. That is

$$u_{15} = [u_{13} \times \phi^{-1}] = 172x_1 + 3520x_2. \qquad m_{15} = 0$$

There exist a gap, $g_2 = u_{14} - u_{15} = 4x_1 + 7x_2$, which we include in the storage G. We continue by letting

$$\begin{aligned}
u_{16} &= u_{13} - u_{15} = 107x_1 + 2176x_2, & m_{16} = 0\\
u_{17} &= u_{15} - u_{16} = 65x_1 + 1344x_2, & m_{17} = 0\\
u_{18} &= u_{16} - u_{17} = 42x_1 + 832x_2, & m_{18} = 0\\
u_{19} &= u_{17} - u_{18} = 23x_1 + 512x_2, & m_{19} = 0\\
u_{20} &= u_{18} - u_{19} = 19x_1 + 320x_2, & m_{20} = 2\\
u_{21} &= u_{19} - u_{20} = 4x_1 + 192x_2. & m_{21} = 2
\end{aligned}$$

We stop the above continuous procedure at u_{21}, since the coefficient of x_1 in u_{21} transcends the coefficient of x_1 in the LOWERBOUND. We include $e_{20}x_1 = 19x_1$ in the storage S.

We let s_1 denote $e_{21}x_1 = 4x_1$ and s_2 denote $e_{20}x_1 - e_{21}x_1 = 15x_1$. We include s_1 and s_2 in G. Henceforth, we will only consider x_2 terms and resume the above process. We let

$$u_{22} = f_{20}x_2 - f_{21}x_2 = 128x_2, \qquad m_{22} = 1$$

since u_{22} does not satisfy the condition $|f_{22}x_2 - (f_{21}x_2 \times \phi^{-1})| < 6x_2$, therefore we utilize an inverse of a golden ratio to get the next u_i. That is

$$u_{23} = [f_{21}x_2 \times \phi^{-1}] = 119x_2. \qquad m_{23} = 0$$

There exist a gap, $g_3 = u_{22} - u_{23} = 9x_2$, which we include in the storage G. We continue by letting

$$\begin{aligned}
u_{24} &= f_{21}x_2 - u_{23} = 73x_2\,, & m_{24} &= 0\\
u_{25} &= u_{23} - u_{24} = 46x_2\,, & m_{25} &= 0\\
u_{26} &= u_{24} - u_{25} = 27x_2\,, & m_{26} &= 0\\
u_{27} &= u_{25} - u_{26} = 19x_2\,, & m_{27} &= 0\\
u_{28} &= u_{26} - u_{27} = 8x_2\,. & m_{28} &= 0
\end{aligned}$$

We stop the above process at u_{28}, since it transcends the given LOWERBOUND= $10x_2$. Hence, we include the last two x_2 terms, that is $19x_2$ and $8x_2$, in the storage S.

We have

$$G = \{g_1 = -8x_1 - 6x_2, g_2 = 4x_1 + 7x_2, g_3 = 9x_2, s_1 = 4x_1, s_2 = 15x_1\}\,.$$

Now we reverse the arrangements of g_j's in G and rename it in the increasing order starting with numeral 1. Hence, we have

$$G = \{g_1 = 9x_2, g_2 = 4x_1 + 7x_2, g_3 = -8x_1 - 6x_2, s_1 = 4x_1, s_2 = 15x_1\}\,.$$

Considering the storage S, we have

$$S = \{1x_1, 1x_2, 2x_1, 2x_2, 3x_1, 3x_2, 19x_1, 19x_2, 8x_2\}\,.$$

Now, using absolute values of all elements in G and S, we shall construct a short addition chain for x_1 terms and x_2 terms, separately.

First, we shall consider constructing doubling-free short addition chain including

absolute values of all x_1 terms in G and S. We will denote it as SAC_{x_1}. Hence, we have

$$\{1x_1, 2x_1, 3x_1, 19x_1, 4x_1, 8x_1, 4x_1, 15x_1\}.$$

Excluding the repeated terms and rearrangement results

$$\{1x_1, 2x_1, 3x_1, 4x_1, 8x_1, 15x_1, 19x_1\}.$$

It follows that $1x_1+2x_1 \rightarrow 3x_1$, $1x_1+3x_1 \rightarrow 4x_1$, $2x_1+3x_1 \rightarrow 5x_1$, $3x_1+5x_1 \rightarrow 8x_1$, $2x_1 + 8x_1 \rightarrow 10x_1$, $5x_1 + 10x_1 \rightarrow 15x_1$ and $4x_1 + 15x_1 \rightarrow 19x_1$. Hence, the following results the SAC_{x_1}.

$$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 8 \rightarrow 10 \rightarrow 15 \rightarrow 19\,.$$

Next, we shall consider constructing doubling-free short addition chain including absolute values of all x_2 terms in G and S. We will denote it as SAC_{x_2}. Hence, we have

$$\{1x_2, 2x_2, 3x_2, 19x_2, 8x_2, 9x_2, 7x_2, 6x_2\}.$$

Excluding the repeated terms and rearrangement results

$$\{1x_2, 2x_2, 3x_2, 6x_2, 7x_2, 8x_2, 9x_2, 19x_2\}.$$

It follows that $1x_2+2x_2 \rightarrow 3x_2$, $2x_2+3x_2 \rightarrow 5x_2$, $1x_2+5x_2 \rightarrow 6x_2$, $1x_2+6x_2 \rightarrow 7x_2$, $1x_2 + 7x_2 \rightarrow 8x_2$, $1x_2 + 8x_2 \rightarrow 9x_2$, $2x_2 + 9x_2 \rightarrow 11x_2$ and $8x_2 + 11x_2 \rightarrow 19x_2$. Hence, the following results the SAC_{x_2}.

$$1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 11 \rightarrow 19\,.$$

Finally, we join SAC_{x_2} at u_{26} to complete the overall chain for $e_0x_1 + f_0x_2$. Note that the old u_{27} and old u_{28} will be replaced with the ones in SAC_2. The continuation of the chain will be denoted as $u_{27} = 19x_2$, $u_{28} = 11x_2$, $u_{29} = 9x_2$, $u_{30} = 8x_2$, $u_{31} = 7x_2$, $u_{32} = 6x_2$, $u_{33} = 5x_2$, $u_{34} = 3x_2$, $u_{35} = 2x_2$, $u_{36} = 1x_2$. Note that $m_i = 0$ for $i = 27, \ldots, 36$.

We reverse the arrangements of m_i's and rename it in the increasing order starting from 0 to 36. That is

$$m = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,2,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0\}.$$

The set v contains all the u_i's of the above chain where $v = \{u_0, u_1, \ldots, u_{36}\}$. We reverse the arrangements of elements in v and rename it in the increasing order starting from v_0 to v_{36}. Hence, we have $v = \{v_0, v_1, \ldots, v_{36}\}$ where $v_i = u_{\ell-i}$ for all i.

Next, we let either $w_i = (j,k)$ or $w_i = (k)$ to denote the index of v_i's involved in the computation of each v_i in v. This could be found from the above chain. Hence, we have $w = \{w_0, w_1 = (0,0), w_2 = (0,1), \ldots, w_{36} = (34,35)\}$, where $w_0 = 1$.

Algorithm 9 Multi-Exponentiation using SGRAC Method

Input: b_1, b_2 are elements of an abelian group and e_0, $f_0 \in \mathbb{Z}$.

Output: The element $b_1^{e_0} b_2^{f_0}$.

Precomputation (SGRAC method)

1. $w = \{w_0, \ldots, w_\ell\}$, $G = \{g_1, \ldots, g_j, s_1, s_2\}$, SAC_{x_1}, SAC_{x_2} and m
2. $b_1 \leftarrow x_1$, $b_2 \leftarrow x_2$
3. Compute all elements in G using SAC_{x_1} and SAC_{x_2}, store it.

Main loop

4. **if** (step 36 of Algorithm 8 holds) **then**
5. $v_0 \leftarrow b_1^1$
6. **else**
7. $v_0 \leftarrow b_2^1$
8. $j \leftarrow 1$
9. $r \leftarrow 1$
10. **for** $i = 1$ **to** ℓ
11. **if** $m_i = 0$ **then**
12. $v_i \leftarrow v_j \times v_k$ $[w_i = (j,k)]$

13. **else if** $m_i = 1$ **then**
14. $v_i \leftarrow g_j \times v_k$ $[w_i = (k)]$
15. $j \leftarrow j + 1$
16. **else**
17. $v_i \leftarrow v_j \times v_k \times s_r$ $[w_i = (j, k)]$
18. $r \leftarrow r + 1$
19. **return** v_ℓ

Example 7.2 Evaluate Algorithm 9 for the inputs $e_0 = 90287$, $f_0 = 1835008$ and $b_1\,, b_2$ as elements of an abelian group.

Precomputation: (SGRAC method)

1. We have $w = \{w_0\,, w_1 = (0,0)\,, w_2 = (0,1), \ldots, w_{36} = (34,35)\}\,,\ G\,,\ SAC_{x_1}\,,$ SAC_{x_2} and m from Example 7.1 .

2. We replace x_1 by b_1 and x_2 by b_2 for all elements in G.

3. Then using SAC_{x_1} and SAC_{x_2} , we compute all the elements in G and again store it in G.

Hence, we get $G = \{g_1 = b_2^9\,, g_2 = b_1^4 b_2^7\,, g_3 = b_1^{-8} b_2^{-6}\,, s_1 = b_1^4\,, s_2 = b_1^{15}\}$.
The following results the evaluation stage. Step 36 of Algorithm 8 does not hold, therefore $v_0 = {b_2}^1$. It follows that

$$m_0 = 0\,, \quad w_0 = (-)\,, \quad v_0 = {b_2}^1\,,$$

$$m_1 = 0\,, \quad w_1 = (0,0)\,, \quad v_1 = v_0 \times v_0 = {b_2}^2\,,$$

$$m_2 = 0\,, \quad w_2 = (0,1)\,, \quad v_2 = v_0 \times v_1 = {b_2}^3\,,$$

$$m_3 = 0\,, \quad w_3 = (1,2)\,, \quad v_3 = v_1 \times v_2 = {b_2}^5\,,$$

$$m_4 = 0\,, \quad w_4 = (0,3)\,, \quad v_4 = v_0 \times v_3 = {b_2}^6\,,$$

$$m_5 = 0\,, \quad w_5 = (0,4)\,, \quad v_5 = v_0 \times v_4 = {b_2}^7\,,$$

$$m_6 = 0\,, \quad w_6 = (0,5)\,, \quad v_6 = v_0 \times v_5 = {b_2}^8\,,$$

$$m_7 = 0\,, \quad w_7 = (0,6)\,, \quad v_7 = v_0 \times v_6 = {b_2}^9\,,$$

$$m_8 = 0\,, \quad w_8 = (1,7)\,, \quad v_8 = v_1 \times v_7 = {b_2}^{11}\,,$$

$$m_9 = 0\,, \quad w_9 = (6,8)\,, \quad v_9 = v_6 \times v_8 = {b_2}^{19}\,,$$

$$m_{10} = 0\,, \quad w_{10} = (6,9)\,, \quad v_{10} = v_6 \times v_9 = {b_2}^{27}\,,$$

$$m_{11} = 0\,, \quad w_{11} = (9,10)\,, \quad v_{11} = v_9 \times v_{10} = {b_2}^{46}\,,$$

$$m_{12} = 0\,, \quad w_{12} = (10,11)\,, \quad v_{12} = v_{10} \times v_{11} = {b_2}^{73}\,,$$

$$m_{13} = 0\,, \quad w_{13} = (11,12)\,, \quad v_{13} = v_{11} \times v_{12} = {b_2}^{119}\,,$$

$$m_{14} = 1\,, \quad w_{14} = (13)\,, \quad v_{14} = g_1 \times v_{13} = {b_2}^{128}\,,$$

$$m_{15} = 2\,, \quad w_{15} = (12,13)\,, \quad v_{15} = v_{12} \times v_{13} \times s_1 = {b_1}^4 {b_2}^{192}\,,$$

$$m_{16} = 2\,, \quad w_{16} = (14,15)\,, \quad v_{16} = v_{14} \times v_{15} \times s_2 = {b_1}^{19} {b_2}^{320}\,,$$

$$m_{17} = 0\,, \quad w_{17} = (15,16)\,, \quad v_{17} = v_{15} \times v_{16} = {b_1}^{23} {b_2}^{512}\,,$$

$$m_{18} = 0\,, \quad w_{18} = (16, 17)\,, \quad v_{18} = v_{16} \times v_{17} = {b_1}^{42} {b_2}^{832}\,,$$

$$m_{19} = 0\,, \quad w_{19} = (17, 18)\,, \quad v_{19} = v_{17} \times v_{18} = {b_1}^{65} {b_2}^{1344}\,,$$

$$m_{20} = 0\,, \quad w_{20} = (18, 19)\,, \quad v_{20} = v_{18} \times v_{19} = {b_1}^{107} {b_2}^{2176}\,,$$

$$m_{21} = 0\,, \quad w_{21} = (19, 20)\,, \quad v_{21} = v_{19} \times v_{20} = {b_1}^{172} {b_2}^{3520}\,,$$

$$m_{22} = 1\,, \quad w_{22} = (21)\,, \quad v_{22} = g_2 \times v_{21} = {b_1}^{176} {b_2}^{3527}\,,$$

$$m_{23} = 0\,, \quad w_{23} = (20, 21)\,, \quad v_{23} = v_{20} \times v_{21} = {b_1}^{279} {b_2}^{5696}\,,$$

$$m_{24} = 0\,, \quad w_{24} = (22, 23)\,, \quad v_{24} = v_{22} \times v_{23} = {b_1}^{455} {b_2}^{9223}\,,$$

$$m_{25} = 0\,, \quad w_{25} = (23, 24)\,, \quad v_{25} = v_{23} \times v_{24} = {b_1}^{734} {b_2}^{14919}\,,$$

$$m_{26} = 0\,, \quad w_{26} = (24, 25)\,, \quad v_{26} = v_{24} \times v_{25} = {b_1}^{1189} {b_2}^{24142}\,,$$

$$m_{27} = 0\,, \quad w_{27} = (25, 26)\,, \quad v_{27} = v_{25} \times v_{26} = {b_1}^{1923} {b_2}^{39061}\,,$$

$$m_{28} = 0\,, \quad w_{28} = (26, 27)\,, \quad v_{28} = v_{26} \times v_{27} = {b_1}^{3112} {b_2}^{63203}\,,$$

$$m_{29} = 1\,, \quad w_{29} = (28)\,, \quad v_{29} = g_3 \times v_{28} = {b_1}^{3104} {b_2}^{63197}\,,$$

$$m_{30} = 0\,, \quad w_{30} = (27, 28)\,, \quad v_{30} = v_{27} \times v_{28} = {b_1}^{5035} {b_2}^{102264}\,,$$

$$m_{31} = 0\,, \quad w_{31} = (29, 30)\,, \quad v_{31} = v_{29} \times v_{30} = {b_1}^{8139} {b_2}^{165461}\,,$$

$$m_{32} = 0\,, \quad w_{32} = (30, 31)\,, \quad v_{32} = v_{30} \times v_{31} = {b_1}^{13174} {b_2}^{267725}\,,$$

$$m_{33} = 0\,, \quad w_{33} = (31, 32)\,, \quad v_{33} = v_{31} \times v_{32} = {b_1}^{21313}{b_2}^{433186}\,,$$

$$m_{34} = 0\,, \quad w_{34} = (32, 33)\,, \quad v_{34} = v_{32} \times v_{33} = {b_1}^{34487}{b_2}^{700911}\,,$$

$$m_{35} = 0\,, \quad w_{35} = (33, 34)\,, \quad v_{35} = v_{33} \times v_{34} = {b_1}^{55800}{b_2}^{1134097}\,,$$

$$m_{36} = 0\,, \quad w_{36} = (34, 35)\,, \quad v_{36} = v_{34} \times v_{35} = {b_1}^{90287}{b_2}^{1835008}\,.$$

7.3 Experimental Results and Discussion

In this section, we show the experimental data for the factors necessitated in obtaining the best results, that is the least chain length for SGRAC.

We carried out an experiment to analyze the two dimension multi-exponentiation based on SGRAC method using a python programming language on 1.66 GHz Intel Core Duo processor. We randomly selected 1000 integers k of 160 bits and set the searching range of the parameters, LOWERBOUND to be between 2 to 22 and MAXIMALGAP to be between 5 to 15. It took 209 trials to obtain chains of lengths between 291 to 306 as shown in Table 7.1. On average, it took about 2.89 seconds to find each chain. The Table 7.2 , 7.3 and 7.4 shows the distribution of MAXIMALGAP, LOWERBOUND and the storage that gives the least chain length, respectively. The average chain length is found to be 301 and the average storage capacity is found to be 26. The conventional Shamir's trick uses 160 squaring operations and 82 multiplication operation for the two dimensional case, where as multi-exponentiation based on SGRAC method utilizes 301 multiplication (best case). Hence, in comparison to Shamir's trick method, the SGRAC method seems to be costly but in terms of security, SGRAC method provides resistance to SPA attack in contrast to Shamir's trick, which is vulnerable to SPA attack.

Table 7.1: The distribution of chains for 1000 randomly selected integers k of 160 bit.

SGRAC length (ℓ)	# inputs k
291	1
292	1
293	0
294	3
295	5
296	13
297	25
298	66
299	103
300	139
301	215
302	177
303	152
304	70
305	26
306	4

Table 7.2: The distribution of MAXIMALGAP for 1000 randomly selected integers k of 160 bit.

MAXIMALGAP	5	6	7	8	9	10	11	12	13	14	15
# inputs k	51	98	100	81	137	93	103	88	75	100	74

Table 7.3: The distribution of LOWERBOUND for 1000 randomly selected integers k of 160 bit.

LOWERBOUND	4	5	6	7	8	9	10	11	12	13	14
# inputs k	115	77	102	87	86	79	71	61	66	54	36

LOWERBOUND	16	17	18	19	20	21	22	15
# inputs k	35	12	23	14	15	20	8	39

Table 7.4: The distribution of storage for 1000 randomly selected integers k of 160 bit.

Storage capacity	16	17	18	19	20	21	22	23	24	25	26	27
# inputs k	60	59	33	7	1	1	15	33	35	50	45	42

Storage capacity	31	32	33	34	35	36	28	29	30
# inputs k	71	71	79	89	70	61	55	55	68

7.4 Conclusion

In this chapter we have proposed a novel method for the computation of multi-exponentiation, that is by using doubling-free short addition chain. Such doubling-free addition chain provides resistance to SPA attack in applications to elliptic curve cryptosystems. The experimental results for the two dimensional case shows that SGRAC method gives an average chain of length 301 with average storage capacity of 26. Further work may include reducing the chain length and the storage capacity. In chapter 8, we will modify SGRAC method to suit the generalized case.

Chapter 8

Multi-Scalar Multiplication

The major building block of most elliptic curve cryptosystems are computation of multi-scalar multiplication. This chapter proposes a novel algorithm for simultaneous multi-scalar multiplication, by employing addition chains. Previously known methods utilizes double-and-add algorithm with binary representations. In order to accomplish our purpose, an efficient empirical method for finding addition chains for multi-exponents has been proposed.

8.1 Introduction

Multi-scalar multiplication is required in many elliptic curve cryptosystems (ECC) such as provable-secure digital signatures [39, 40], multi-party protocols [4] and protocols of Brands [7]. It is given by the formula $\sum_{i=1}^{t} k_i G_i$ where k_i is a scalar variable (exponent), G_i shows a rational point (base) on an elliptic curve and i is an integer in $[1, t]$ where $t \geq 2$.

In most cases where multi-scalar multiplication is applied, the process is dominant in determining the overall efficiency. Hence, efficiency of multi-scalar multiplication is essential in elliptic curve cryptosystems. Conventional methods for computation of multi-scalar multiplication can be classified into two types. In methods of one type includes independent computation of the scalar multiples $k_i G_i$, followed by

their addition. Such a method could be very expensive but in cases where some of the scalars are fixed then a comb method [29] combined with a window method could enhance the overall efficiency of the process. In the methods of the other type, the multi-scalar multiplication is computed in one stage, without separate computation of k_iG_i. This includes simultaneous methods such as Shamir [13] and Interleave [35] methods which utilizes binary representations for double-and-add algorithm.

In this chapter we propose a novel algorithm for simultaneous multi-scalar multiplication by utilizing addition chains. Hence, to accomplish our purpose, we modify the simultaneous golden ratio addition chain (SGRAC) method proposed in chapter 7 to suit the general case. We term it as *modified* simultaneous golden ratio addition chain method or *mod*-SGRAC method for the sake of convenience. We state the algorithm for 2-dimensional case and through experimental analysis we show that such strategy is suitable for general case. As a result, we give an experimental analysis for dimensions $2, 3, 4, 5$ and 6.

8.2 Simultaneous Addition Chain for Multi-Exponents

Here, we discuss an efficient empirical method for the construction of simultaneous addition chain for multi-exponents, considering the 2-dimensional case. We will discuss the similar strategy proposed in chapter 7 with slight modification to suit the general case. We term it as *modified* simultaneous golden ratio addition chain method or *mod*-SGRAC method for the sake of convenience.

The *mod*-SGRAC method constructs a chain starting from the last term, that is the input exponents, u and v. We pair the two exponents with variables x and y to distinguish it from each other. Hence, we let $w_i = u_ix + v_iy$ in general. Our aim is to follow a Fibonacci pattern using the fact from equation (4.1). Hence, we try to

maintain a near golden ratio value between succeeding terms. We begin by letting

$$\begin{aligned} w_0 &= ux + vy\,, \\ w_1 &= [w_0 \times \phi^{-1}]\,, \\ w_i &= w_{i-2} - w_{i-1} \qquad \text{for} \qquad i = 2, 3, \ldots \end{aligned} \tag{8.1}$$

Here w_i denotes the reverse of c_i that is, $w_i = c_{\ell-i}$. If continued with the procedure (8.1), w_i will exponentially deviate from $(w_{i-1} \times \phi^{-1})$ as i increases. In order to overcome this problem, a parameter MAXIMALGAP is introduced, where MAXIMALGAP $= u_{MG}x + v_{MG}y$. Hence, the above procedure (8.1) terminates whenever

$$|w_i - (w_{i-1} \times \phi^{-1})| > u_{MG}x + v_{MG}y \text{ or } w_i \leqslant \frac{w_{i-1}}{2}\,.$$

Note that the above inequalities holds for the corresponding x and y terms. Hence, a new w_i is defined to be the nearest integer of $(w_{i-1} \times \phi^{-1})$. Then procedure (8.1) is resumed with w_{i-1} and new w_i as the initial terms. The old w_i is included in the chain between w_{i-1} and new w_i. As a consequence, there is a gap g_j =old w_i−new w_i, which is included in the storage. Note that, subtraction is involved whenever old $w_i <$ new w_i.

We introduce another parameter LOWERBOUND as $u_{LB}x + v_{LB}y$. The above procedure (8.1) stops in any of the following three cases; (i) $(u_i < u_{LB})$ and $(v_i < v_{LB})$, (ii) $u_i < u_{LB}$ and $v_i > v_{LB}$, (iii) $u_i > u_{LB}$ and $v_i < v_{LB}$. The details of these three cases are included in the *mod*-SGRAC algorithm.

8.3 Proposed *mod*-SGRAC method

The following is an algorithm for a *modified* simultaneous golden ratio addition chain method.

Algorithm 10 *modified* Simultaneous Golden Ratio Addition Chain Method

Input: An integer $u\,, v\,$, MAXIMALGAP and LOWERBOUND.

Output: $m = \{m_1, \ldots, m_{n+1}\}_{SGRAC}\,$, $SAC_x\,$, SAC_y and S.

1. $\phi^{-1} \leftarrow \frac{-1+\sqrt{5}}{2}$
2. $w_i \leftarrow u_i x + v_i y$
3. $w_0 \leftarrow ux + vy$
4. $w_1 \leftarrow [w_0 \times \phi^{-1}]$
5. $w_2 \leftarrow w_0 - w_1$
6. $m = \{0, 0\}$
7. $S = \{1x, 1y, 2x, 2y, 3x, 3y\}$
8. $G \leftarrow \emptyset$
9. $i \leftarrow 2$
10. $j \leftarrow 1$
11. $(u_{MG} x + v_{MG} y) \leftarrow$ MG
12. $(u_{LB} x + v_{LB} y) \leftarrow$ LB
13. **while** $(u_i x > u_{LB} x)$ or $(v_i y > v_{LB} y)$ **do**
14. $\quad m_i \leftarrow 0$
15. $\quad$ **if** $|w_i - (w_{i-1} \times \phi^{-1})| >$ MG or $w_i \leqslant \frac{w_{i-1}}{2}$ **then**
16. $\quad\quad w_{i+1} \leftarrow [w_{i-1} \times \phi^{-1}]$
17. $\quad\quad g_j \leftarrow (w_i - w_{i+1})$
18. $\quad\quad S \leftarrow S \cup \{g_j\}$
19. $\quad\quad j \leftarrow j + 1$
20. $\quad\quad m_{i-1} \leftarrow 2\,,\ m_i \leftarrow 1\,,\ m_{i+1} \leftarrow 0$
21. $\quad\quad m \leftarrow m \cup \{m_{i-1}\,, m_i\,, m_{i+1}\,\}$
22. $\quad\quad w_{i+2} \leftarrow (w_{i-1} - w_{i+1})$
23. $\quad\quad i \leftarrow i + 2$
24. $\quad$ **else**
25. $\quad\quad m \leftarrow m \cup \{m_i\}$
26. $\quad\quad i \leftarrow i + 1$
27. $\quad\quad w_i \leftarrow (w_{i-2} - w_{i-1})$
28. **if** $(u_i < u_{LB})$ and $(v_i < v_{LB})$ **then**
29. $\quad m_{i-1} \leftarrow 3\,,\ m_i \leftarrow 3$
30. $\quad m \leftarrow m \cup \{m_{i-1}, m_i\}$

31. $T_1 \leftarrow w_{i-1}$, $T_0 \leftarrow w_i$
32. $S \leftarrow S \cup \{T_1 , T_0\}$
33. SAC_x
34. SAC_y
35. **else if** $u_i < u_{LB}$ and $v_i > v_{LB}$ **then**
36. $g_j \leftarrow u_{i-1}x$, $g_{j+1} \leftarrow u_i x$
37. $S \leftarrow S \cup \{g_j , g_{j+1}\}$
38. $j \leftarrow j + 2$
39. SAC_x
40. $v_{i+1}y \leftarrow v_{i-1}y$
41. $v_{i+2}y \leftarrow v_i y$
42. $m_{i+1} \leftarrow 0$, $m_{i+2} \leftarrow 0$
43. $m \leftarrow m \cup \{m_{i+1} , m_{i+2}\}$
44. $v_{i+3}y \leftarrow (v_{i+1}y - v_{i+2}y)$
45. $i \leftarrow i + 3$
46. Repeat step 13 to step 27 only for y terms
47. $m_{i-1} \leftarrow 3$, $m_i \leftarrow 3$
48. $m \leftarrow m \cup \{m_{i-1}, m_i\}$
49. $T_1 \leftarrow v_{i-1}y$, $T_0 \leftarrow v_i y$
50. $S \leftarrow S \cup \{T_1 , T_0\}$
51. SAC_y
52. **else**
53. $g_j \leftarrow v_{i-1}y$, $g_{j+1} \leftarrow v_i y$
54. $S \leftarrow S \cup \{g_j , g_{j+1}\}$
55. $j \leftarrow j + 2$
56. SAC_y
57. $u_{i+1}x \leftarrow u_{i-1}x$
58. $u_{i+2}x \leftarrow u_i x$
59. $m_{i+1} \leftarrow 0$, $m_{i+2} \leftarrow 0$
60. $m \leftarrow m \cup \{m_{i+1} , m_{i+2}\}$

61. $u_{i+3}x \leftarrow (u_{i+1}x - u_{i+2}x)$

62. $i \leftarrow i + 3$

63. Repeat step 13 to step 27 only for x terms

64. $m_{i-1} \leftarrow 3$, $m_i \leftarrow 3$

65. $m \leftarrow m \cup \{m_{i-1}, m_i\}$

66. $T_1 \leftarrow u_{i-1}x$, $T_0 \leftarrow u_i x$

67. $S \leftarrow S \cup \{T_0 , T_1\}$

68. SAC_x

69. $max \leftarrow j - 1$

70. $n \leftarrow i - 1$

71. $m \leftarrow$ reverse the arrangements in m and rename the elements in increasing order starting with numeral 1 to $n + 1$

72. **return** $m = \{m_1, \ldots, m_{n+1}\}_{SGRAC}$, SAC_x, SAC_y and S

Note that in step 15 we check if either of the inequalities are satisfied for the corresponding x terms or y terms. In steps $20, 29, 47$ and 64, the old m_{i-1} has been replaced with new m_{i-1} in m . Also note that SAC_x and SAC_y represents the construction of short addition chains using the absolute values of x terms and y terms from the storage, respectively. The symbols MG and LB represents MAXIMALGAP and LOWERBOUND, respectively.

Example 8.1 Evaluate Algorithm 10 for the inputs $u = 10361$, $v = 103864$, LOWERBOUND $= 5x + 5y$ and MAXIMALGAP $= 10x + 10y$.

First, we will find the *mod*-SGRAC representation m, during which we will obtain the elements for the storage S. Later, we will use all the storage elements to construct a short addition chain SAC_x and SAC_y.

We pair u and v with variables x and y to distinguish their computations. We

begin, letting

$$\begin{aligned}
w_0 &= 10361x + 103864y\,, & m_0 &= 0\\
w_1 &= [w_0 \times \phi^{-1}] = 6403x + 64191y\,, & m_1 &= 0\\
w_2 &= w_0 - w_1 = 3958x + 39673y\,, & m_2 &= 0\\
w_3 &= w_1 - w_2 = 2445x + 24518y\,, & m_3 &= 0\\
w_4 &= w_2 - w_3 = 1513x + 15155y\,, & m_4 &= 0\\
w_5 &= w_3 - w_4 = 932x + 9363y\,, & m_5 &= 2\\
w_6 &= w_4 - w_5 = 581x + 5792y\,, & m_6 &= 1
\end{aligned}$$

since v_6y exceeds the MAXIMALGAP, that is $|v_6y - (v_5y \times \phi^{-1})| > 5y$, we let

$$w_7 = [w_5 \times \phi^{-1}] = 576x + 5787y\,. \qquad m_7 = 0$$

There exists a gap, $g_1 = w_6 - w_7 = 5x + 5y$, which we include in the storage. We resume, letting

$$\begin{aligned}
w_8 &= w_5 - w_7 = 356x + 3576y\,, & m_8 &= 0\\
w_9 &= w_7 - w_8 = 220x + 2211y\,, & m_9 &= 0\\
w_{10} &= w_8 - w_9 = 136x + 1365y\,, & m_{10} &= 0\\
w_{11} &= w_9 - w_{10} = 84x + 846y\,, & m_{11} &= 0\\
w_{12} &= w_{10} - w_{11} = 52x + 519y\,, & m_{12} &= 2\\
w_{13} &= w_{11} - w_{12} = 32x + 327y\,, & m_{13} &= 1
\end{aligned}$$

since $v_{13}y$ exceeds the MAXIMALGAP, that is $|v_{13}y - (v_{12}y \times \phi^{-1})| > 5y$, we let

$$w_{14} = [w_{12} \times \phi^{-1}] = 32x + 321y. \qquad m_{14} = 0$$

There exists a gap, $g_2 = 6y$, which we include in the storage. We resume, letting

$$\begin{aligned}
w_{15} &= w_{12} - w_{14} = 20x + 198y\,, & m_{15} &= 0\\
w_{16} &= w_{14} - w_{15} = 12x + 123y\,, & m_{16} &= 3\\
w_{17} &= w_{15} - w_{16} = 8x + 75y\,, & m_{17} &= 3
\end{aligned}$$

Henceforth, we terminate the x term of w_i, since it transcends the corresponding LOWERBOUND $= 10x$. Thus, we store $g_3 = 12x$ and $g_4 = 8x$. We continue the

above process, considering the y terms only. We let

$$\begin{aligned} w_{18} &= 123y\,, & m_{18} &= 0 \\ w_{19} &= 75y\,, & m_{19} &= 0 \end{aligned}$$

Hence, we have

$$\begin{aligned} w_{20} &= w_{18} - w_{19} = 48y\,, & m_{20} &= 0 \\ w_{21} &= w_{19} - w_{20} = 27y\,, & m_{21} &= 0 \\ w_{22} &= w_{20} - w_{21} = 21y\,, & m_{22} &= 0 \end{aligned}$$

We stop the above continuous procedure at w_{22}, since the next term,

$$w_{23} = w_{21} - w_{22} = 6y\,,$$

transcends the given LOWERBOUND $= 10y$. Let $T_0 = 6y\,,\ T_1 = 21y$ and store it in $S\,$. Hence, we obtained the following storage.

$$S = \{1x, 1y, 2x, 2y, 3x, 3y, g_1 = 5x + 5y, g_2 = 6y, g_3 = 12x, g_4 = 8x,$$

$$T_1 = 21y, T_0 = 6y\}.$$

We list $m_0, \ldots, m_{22}$ as elements of the set m. Hence, we have

$$m = \{0, 0, 0, 0, 0, 2, 1, 0, 0, 0, 0, 0, 2, 1, 0, 0, 3, 3, 0, 0, 0, 0, 0\}.$$

Then we reverse the arrangements of the elements in the set m and rename it in the increasing order starting from m_1 to m_{23}. Thus, it results in the following *mod*-SGRAC representation.

$$m = \{0, 0, 0, 0, 0, 3, 3, 0, 0, 1, 2, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0, 0\}_{mod-SGRAC}.$$

Next, we shall consider constructing doubling-free short addition chain including absolute values of all x terms in S. We will denote it as SAC_x. Hence, we have

$$\{1x, 2x, 3x, 5x, 12x, 8x\}.$$

Excluding the repeated terms and rearrangement results

$$\{1x, 2x, 3x, 5x, 8x, 12x\}.$$

It follows that $1x + 2x \rightarrow 3x$, $2x + 3x \rightarrow 5x$, $3x + 5x \rightarrow 8x$, $3x + 8x \rightarrow 11x$ and $1x + 11x \rightarrow 12x$. Hence, the following results the SAC_x.

$$1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 8 \rightarrow 11 \rightarrow 12.$$

Now, we shall consider constructing doubling-free short addition chain including absolute values of all y terms in S. We will denote it as SAC_y. Hence, we have

$$\{1y, 2y, 3y, 5y, 6y, 21y, 6y\}.$$

Excluding the repeated terms and rearrangement results

$$\{1y, 2y, 3y, 5y, 6y, 21y\}.$$

It follows that $1y + 2y \rightarrow 3y$, $2y + 3y \rightarrow 5y$, $y + 5y \rightarrow 6y$, $5y + 6y \rightarrow 11y$, $6y + 11y \rightarrow 17y$, $3y + 17y \rightarrow 20y$ and $1y + 20y \rightarrow 21y$. Hence the following results the SAC_y.

$$1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 11 \rightarrow 17 \rightarrow 20 \rightarrow 21\,.$$

8.4 Application to Elliptic Curve Cryptosystems

In this section, we propose a multi-scalar multiplication algorithm by utilizing the proposed *mod*-SGRAC method.

Algorithm 11 Multi-Scalar Multiplication using *mod*-SGRAC

Input: An integer u, v and $P, Q \in E(\mathbb{F}_p)$.

Output: $uP + vQ$.

Precomputation (*mod*-SGRAC method)

1. $m = \{m_1, \ldots, m_{n+1}\}_{SGRAC}$
2. S
3. SAC_x and SAC_y
4. $G \leftarrow \emptyset$
5. $x \leftarrow P$, $y \leftarrow Q$

6. **for** $j = 1$ **to** max
7. $g_j \leftarrow$ compute using SAC_x and SAC_y
8. $G \leftarrow G \cup \{g_j\}$
9. $G \leftarrow$ reverse the arrangements in G and rename the elements in increasing order starting with numeral 1 to max
10. $T_0 \leftarrow$compute using SAC_x or SAC_y
11. $T_1 \leftarrow$compute using SAC_x or SAC_y

Main loop

12. $j \leftarrow 1$
13. **for** $i = 1$ **to** n **do**
14. **if** $e_{i+1} = 0$ **then**
15. $T_{i+1} \leftarrow T_{i-1} + T_i$
16. **else if** $e_{i+1} = 1$ **then**
17. $T_{i+1} \leftarrow T_i + G_j$
18. $j \leftarrow j + 1$
19. **else if** $e_{i+1} = 2$ **then**
20. $T_{i+1} \leftarrow T_{i-2} + T_{i-1}$
21. **else** $e_{i+1} = 3$ **then**
22. $T_{i+1} \leftarrow T_{i-1} + G_j$
23. $j \leftarrow j + 1$
24. **return** T_{n+1}

Hence, the required output $uP + vQ = T_{n+1}$. Note that Algorithm 11 involves storage of the preceding two points during scalar multiplication. Also, a temporary storage G containing max number of points g_j's, which are discarded ones being used, during the multi-scalar multiplication.

Example 8.2 Compute $10361P + 103864Q$ using Algorithm 11.

Precomputation: Example 8.1 results the following.

$m = \{0,0,0,0,0,3,3,0,0,1,2,0,0,0,0,0,1,2,0,0,0,0,0\}_{SGRAC}$.
$S = \{1x, 1y, 2x, 2y, 3x, 3y, g_1 = 5x + 5y, g_2 = 6y, g_3 = 12x,$
$g_4 = 8x, T_1 = 21y, T_0 = 6y\}$.
$SAC_x : 1 \to 2 \to 3 \to 5 \to 8 \to 11 \to 12\,.$
$SAC_y : 1 \to 2 \to 3 \to 5 \to 6 \to 11 \to 17 \to 20 \to 21\,.$

Next, we replace x and y with P and Q in S, respectively. Then, we compute all g_j's using SAC_x and SAC_y for $j = 1$ to 4, which are then stored in G. Hence, we have $G = \{g_1 = 5P + 5Q, g_2 = 6Q, g_3 = 12P, g_4 = 8P\}$. Now we reverse the arrangements in G and rename the elements in the increasing order starting with numeral 1 to 4. Hence, we have $G = \{g_1 = 8P, g_2 = 12P, g_3 = 6Q, g_4 = 5P + 5Q\}$. We compute $T_0 = 6Q$ and $T_1 = 21Q$ using SAC_y.

Evaluation Stage.

Henceforth, we use the *mod*-SGRAC representation of m, to compute T_i for $i = 1$ to $i = 23$ as follows.

$$
\begin{aligned}
&i = 1\,, \quad m_2 = 0\,, \quad T_2 = T_0 + T_1 \;=\; 27Q\,,\\
&i = 2\,, \quad m_3 = 0\,, \quad T_3 = T_1 + T_2 \;=\; 48Q\,,\\
&i = 3\,, \quad m_4 = 0\,, \quad T_4 = T_2 + T_3 \;=\; 75Q\,,\\
&i = 4\,, \quad m_5 = 0\,, \quad T_5 = T_3 + T_4 \;=\; 123Q\,,\\
&i = 5\,, \quad m_6 = 3\,, \quad T_6 = T_4 + g_1 \;=\; 8P + 75Q\,,\\
&i = 6\,, \quad m_7 = 3\,, \quad T_7 = T_5 + g_2 \;=\; 12P + 123Q\,,\\
&i = 7\,, \quad m_8 = 0\,, \quad T_8 = T_6 + T_7 \;=\; 20P + 198Q\,,\\
&i = 8\,, \quad m_9 = 0\,, \quad T_9 = T_7 + T_8 \;=\; 32P + 321Q\,,\\
&i = 9\,, \quad m_{10} = 1\,, \quad T_{10} = T_9 + g_3 \;=\; 32P + 327Q\,,\\
&i = 10\,, \quad m_{11} = 2\,, \quad T_{11} = T_8 + T_9 \;=\; 52P + 519Q\,,\\
&i = 11\,, \quad m_{12} = 0\,, \quad T_{12} = T_{10} + T_{11} \;=\; 84P + 846Q\,,\\
&i = 12\,, \quad m_{13} = 0\,, \quad T_{13} = T_{11} + T_{12} \;=\; 136P + 1365Q\,,\\
&i = 13\,, \quad m_{14} = 0\,, \quad T_{14} = T_{12} + T_{13} \;=\; 220P + 2211Q\,,
\end{aligned}
$$

$$
\begin{aligned}
&i = 14\,, \quad m_{15} = 0\,, \quad T_{15} = T_{13} + T_{14} \;=\; 356P + 3576Q\,,\\
&i = 15\,, \quad m_{16} = 0\,, \quad T_{16} = T_{14} + T_{15} \;=\; 576P + 5787Q\,,\\
&i = 16\,, \quad m_{17} = 1\,, \quad T_{17} = T_{16} + g_4 \;=\; 581P + 5792Q\,,\\
&i = 17\,, \quad m_{18} = 2\,, \quad T_{18} = T_{15} + T_{16} \;=\; 932P + 9363Q\,,\\
&i = 18\,, \quad m_{19} = 0\,, \quad T_{19} = T_{17} + T_{18} \;=\; 1513P + 15155Q\,,\\
&i = 19\,, \quad m_{20} = 0\,, \quad T_{20} = T_{18} + T_{19} \;=\; 2445P + 24518Q\,,\\
&i = 20\,, \quad m_{21} = 0\,, \quad T_{21} = T_{19} + T_{20} \;=\; 3958P + 39673Q\,,\\
&i = 21\,, \quad m_{22} = 0\,, \quad T_{22} = T_{20} + T_{21} \;=\; 6403P + 64191Q\,,\\
&i = 22\,, \quad m_{23} = 0\,, \quad T_{23} = T_{21} + T_{22} \;=\; 10361P + 103864Q\,.
\end{aligned}
$$

8.5 Experimental Results and Discussion

In this section, we show the experimental analysis for multi-scalar multiplication algorithm based on *mod*-SGRAC method for 2-dimensional case. We consider the factors necessitated in obtaining the best results.

We carried out an experiment to analyze Algorithm 11 using a python programming language on 2.60 GHz intel celeron processor. We randomly selected 1000 integers u and v of 160 bits and set the searching range of the parameters, LOWERBOUND to be between $4x + 4y$ to $23x + 23y$ and MAXIMALGAP to be between $5x+5y$ to $16x+16y$. It took 209 trials to obtain chains of lengths between 295 to 316, as shown in Table 8.4. On average it took about 11.31 seconds to find each chain. Tables 8.1, 8.2, 8.3, 8.5 and 8.6 shows the distribution of precomputation, storage, main loop, MAXIMALGAP and LOWERBOUND, respectively. The average chain length is found to be 307 and the average storage capacity is found to be 17. Experimental results shows that *mod*-SGRAC method achieves 63% of Fibonacci pattern, but overall we can not guarantee the optimality of the *mod*-SGRAC method. This may be considered as a further research problem.

In a similar experiment as mentioned above, we give an experimental analysis of multi-scalar multiplication based on *mod*-SGRAC method for dimensions $t = 2, \ldots, 6$ as shown in Table 8.7. We randomly selected 1000 scalars k_i's of 160 bits

Table 8.1: The distribution of precomputation chain lengths for 1000 randomly selected integers u and v of 160 bit.

length (ℓ)	12	13	14	15	16	17	18	19	20
# inputs	31	198	463	144	103	37	12	10	2

Table 8.2: The distribution of storage for 1000 randomly selected integers u and v of 160 bit.

storage capacity	14	15	16	17	18	19
# inputs	5	42	173	397	345	38

Table 8.3: The distribution of main loop chain lengths for 1000 randomly selected integers u and v of 160 bit.

length (ℓ)	280	281	282	283	284	285	286	287
# inputs	2	1	-	5	6	8	26	30

length (ℓ)	288	289	290	291	292	293	294	295
# inputs	37	60	80	80	90	110	109	118

length (ℓ)	296	297	298	299	300	301	302	303
# inputs	103	49	41	20	15	7	2	1

for the respective dimensions. It shows that there is a linear increase in the storage capacity and the lengths of addition chains with respect to the dimensions.

We consider mixed coordinates system to compute the computational cost for the proposed method as shown in Table 8.8. This is cheaper compared to the other coordinate systems, as proposed by Cohen et al. [9] . Note that we will use $[i], [s]$ and $[m]$ to denote the cost of one inversion, one squaring and one multiplication,

Table 8.4: The distribution of total chain lengths for 1000 randomly selected integers u and v of 160 bit.

length (ℓ)	295	296	297	298	299	300	301	302
# inputs	1	2	1	3	3	13	25	34

length (ℓ)	303	304	305	306	307	308	309	310
# inputs	48	71	83	105	121	127	125	103

length (ℓ)	311	312	313	314	315	316
# inputs	60	33	22	11	7	2

Table 8.5: The distribution of MAXIMALGAP for 1000 randomly selected integers u and v of 160 bit.

MAXIMALGAP	5	6	7
# inputs	783	203	14

Table 8.6: The distribution of LOWERBOUND for 1000 randomly selected integers u and v of 160 bit.

LOWERBOUND	4	5	6	7	8	9	10	11
# inputs	248	154	197	137	75	54	34	28

LOWERBOUND	12	13	14	15	16	17	18	19
# inputs	14	12	8	15	8	4	3	4

LOWERBOUND	20	21	22
# inputs	3	1	1

Table 8.7: Analysis of simultaneous multi scalar multiplication based on *mod*-SGRAC method for higher dimensions.

Dimension (t)	Avg. storage	Average precomputation	Average Main loop	Average Cost	Fibonacci pattern	Average Run time
2	17	14 ECADD + 2 ECDBL	293 ECADD	307 ECADD + 2 ECDBL	63%	10.7 sec
3	26	21 ECADD + 3 ECDBL	333 ECADD	354 ECADD + 3 ECDBL	54%	13.9 sec
4	35	28 ECADD + 4 ECDBL	372 ECADD	400 ECADD + 4 ECDBL	47%	17.5 sec
5	44	34 ECADD + 5 ECDBL	411 ECADD	445 ECADD + 5 ECDBL	43%	21 sec
6	53	41 ECADD + 6 ECDBL	449 ECADD	490 ECADD + 6 ECDBL	39%	24.3 sec

Table 8.8: Computational costs using mixed coordinates.

	Addition	Doubling
Operation	$\mathcal{A} + \mathcal{A} = \mathcal{J}^C$	$2\mathcal{A} = \mathcal{J}$
Costs	$5[m] + 3[s]$	$2[m] + 4[s]$

Table 8.9: Comparison with 2-dimensional simultaneous multi-scalar multiplication methods for 160 bit integers u and v.

Algorithm	Avg. storage	Average precomputation	Avg. ECADD	Avg. ECDBL	Average Cost	#[m]
Shamir [13]	1	1 ECADD	120	160	121 ECADD + 160 ECDBL	1728
Shamir+JSF [45]	4	4 ECADD	82	160	86 ECADD + 160 ECDBL	1469
Shamir+JSF_3 [10]	10	10 ECADD + 2 ECDBL	59	160	69 ECADD + 162 ECDBL	1353
Interleave [35]	-	-	160	160	160 ECADD + 160 ECDBL	2016
Interleave+wMOF [10]	10	10 ECADD + 2 ECDBL	59	160	69 ECADD + 162 ECDBL	1353
Our Proposed	**17**	**14 ECADD + 2 ECDBL**	**293**	**-**	**307 ECADD + 2 ECDBL**	2283

respectively. We shall always leave out the cost of field additions. Generally, it is assumed $[s] = 0.8[m]$ for curves over odd prime field [14]. Note that in Table 8.9, the computational cost of multiplication is denoted by $\#[m]$. The comparison in

Table 8.9 shows that our proposed method is slightly costly compared to the other methods available in the literature. However, for further research, if one is able to reduce the length of *mod*-SGRAC, then the proposed method can precede other methods.

8.6 Conclusion

In this chapter we have proposed a novel algorithm for the simultaneous computation of multi-scalar multiplication by employing addition chains. In order to accomplish our purpose, we have proposed an efficient empirical method to generate addition chains for multi-exponents simultaneously. The analysis from Table 8.7 shows that there is a linear increase in the cost with respect to the dimension of the multi-scalar multiplication. The comparison on Table 8.9 shows that our proposed method is slightly costly compared to the other methods available in the literature. However, further work may include reducing storage capacity and chain length of *mod*-SGRAC method in order to enhance efficiency of the proposed algorithm.

Chapter 9

Conclusion

In this thesis, importance of addition chains in applications to elliptic curve cryptosystems were shown. In particular, addition chains have been exploited for the computations of scalar and multi-scalar multiplication. These computations are considered as the basic operations for elliptic curve cryptosystems. This thesis consists of four major results which are discussed separately in chapters $5, 6, 7$ and 8, respectively. These achievements are described in details in the following paragraphs.

In chapter 5, a new strategy to find an efficient doubling-free short addition-subtraction chain has been proposed. It was termed as golden ratio addition-subtraction chain (GRASC) method. An empirical data showed that GRASC method has attained 12% to 28% reduced in the average chain length compared to the other doubling-free addition chain methods. The results obtained shows usefulness in applications to elliptic curve cryptosystems, specially in computations of scalar multiplication.

In chapter 6, an explicit algorithm (GRAC method) for the GRASC method of chapter 5 has been proposed. This was later exploited in proposing a SPA resistance scalar multiplication algorithm. The analysis showed that the proposed scalar multiplication algorithm using GRAC method has preceded other scalar multiplication algorithm by 3% to 18%. The results obtained have an impact in applications to elliptic curve cryptosystems.

In chapter 7, a novel method for the computation of multi-exponentiation was proposed by exploiting the results of chapter 5. More specifically, a two dimensional case was considered. It was termed as simultaneous golden ratio addition chain (SGRAC) method. An experimental results for the two dimensional case showed that SGRAC method gives an average chain of length 301 with average storage capacity of 26. The results obtained shows usefulness in applications to elliptic curve cryptosystems, that is, computations of multi-scalar multiplication.

In chapter 8, a novel algorithm for simultaneous computation of multi-scalar multiplication was proposed by employing addition chains. In fact, the results of chapter 7 was modified to accomplish this purpose. The empirical analysis in Table 8.7 showed that there is a linear increase in the cost with respect to the dimension of the multi-scalar multiplication. The comparison in Table 8.9 showed that the proposed method is slightly costly compared to the other methods available in the literature.

9.1 Further Research

The four major results of the thesis are inherited from the proposed strategy for short addition-subtraction chain discussed in chapter 5, known as the golden ratio addition-subtraction chain method or GRASC method. Therefore, the efficiency of the four major results could be effectively increase by reducing the chain length of the GRASC method. This could be considered as a primary future work. Also, it is important to further reduce the storage capacity of the GRASC method in order to increase its impetus in elliptic curve cryptosystems. Although the results discussed in the thesis are spectacular in applications to elliptic curve cryptosystems, they are based on empirical analysis. Therefore, there is a strong desire to present the theoretical aspects of the major results discussed in the thesis for further research.

Appendix

Here, we give an estimation of the storage capacity. That is the number of g_j's required in GRAC method for a 160 bit integer. Note that the number of g_j's are same as the number of new u_i being computed.

We have $\left\{u_0 = k\,, u_1 = [\frac{k}{\phi}]\,, u_2 = u_0 - u_1\,, u_3 = u_1 - u_2\,, \ldots\right\}$.

Note that Fibonacci sequence gives the optimum result, therefore we have a desire to maintain such pattern for the GRAC method. Thus, we will check at every step to maintain the property of a Fibonacci sequence given by equation (4.1).

For $i = 1$

$$\left|u_1 - \frac{k}{\phi}\right| \;\leq\; \frac{1}{2}\,.$$

For $i = 2$

$$\begin{aligned}
\left|u_2 - \frac{k}{\phi^2}\right| &= \left|u_0 - u_1 - \frac{k}{\phi^2}\right| \\
&= \left|k - u_1 - \frac{k}{\phi^2}\right| \\
&= \left|k - \frac{k}{\phi} - \frac{k}{\phi^2} + \frac{k}{\phi} - u_1\right| \\
&= \left|\frac{k}{\phi^2}(\phi^2 - \phi - 1) + \frac{k}{\phi} - u_1\right|\,, \text{ using Theorem 4.1.1, we get} \\
&= \left|\frac{k}{\phi} - u_1\right| \leq \frac{1}{2}\,.
\end{aligned}$$

For $i = 3$

$$\begin{aligned}
\left|u_3 - \frac{k}{\phi^3}\right| &= \left|u_1 - u_2 - \frac{k}{\phi^3}\right| \\
&= \left|u_1 - \frac{k}{\phi} + \frac{k}{\phi^2} - u_2 - \frac{k}{\phi^3} + \frac{k}{\phi} - \frac{k}{\phi^2}\right| \\
&= \left|(u_1 - \frac{k}{\phi}) + (\frac{k}{\phi^2} - u_2) - \frac{k}{\phi^3}(1 - \phi^2 + \phi)\right| \text{ , using Theorem 4.1.1, we get} \\
&= \left|u_1 - \frac{k}{\phi}\right| + \left|\frac{k}{\phi^2} - u_2\right| \\
&\leq \frac{1}{2} + \frac{1}{2} = 1\,.
\end{aligned}$$

Hence, in general we have

$$\left|u_i - \frac{k}{\phi^i}\right| \leq \left|u_{i-2} - \frac{k}{\phi^{i-2}}\right| + \left|u_{i-1} - \frac{k}{\phi^{i-1}}\right| .$$

Assuming MAXIMALGAP to be 11, we will check the number of terms exist before MAXIMALGAP is exceeded. It follows that

$$\begin{aligned}
i = 4: &\quad \tfrac{1}{2} + 1 &&= \tfrac{3}{2}\,, \\
i = 5: &\quad 1 + \tfrac{3}{2} &&= \tfrac{5}{2}\,, \\
i = 6: &\quad \tfrac{3}{2} + \tfrac{5}{2} &&= 4\,, \\
i = 7: &\quad \tfrac{5}{2} + 4 &&= \tfrac{13}{2}\,, \\
i = 8: &\quad 4 + \tfrac{13}{2} &&= \tfrac{21}{2}\,, \\
i = 9: &\quad \tfrac{13}{2} + \tfrac{21}{2} &&= 17\,.
\end{aligned}$$

The MAXIMALGAP is exceeded at $i = 9$, therefore at $i = 10$, a new u_i is computed. Hence, we could infer that in worst case, a new u_i is computed once in every 10th term for GRAC method. Therefore, the maximum number of points in the storage for an average chain of length 258 will be 26 .

Bibliography

[1] M. Anshel and Goldfeld. Zeta functions, one-way functions, and pseudorandom number generators. *IEEE Duke Mathematics Journal*, vol.88, pp.371-390, 1997.

[2] R.M. Avanzi. On multi-exponentiation in Cryptography. *Cryptology ePrint Archive: report 2002/154*, 2000.

[3] R.M. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography.* CRC Press, 2005.

[4] M. Bellare, J.A. Garray, and T. Rabin. Fast Batch verification for modular exponentiation and digital signatures. *Advances in Cryptology-EUROCRYPTO'98*, volume 1403 of *Lecture Notes in Computing Science*, pages 236-250. Springer-Verlag, 1998.

[5] I.F. Blake, G. Seroussi, and N.P. Smart. *Elliptic Curves in Cryptography.* Number 256 in London Mathematical Society Lecture Note Series. Cambridge University Press, 1999.

[6] J. Bos and M. Coster. Addition chain heuristics. *Advances in Cryptology-CRYPTO'89*, volume 435 of *Lecture Notes in Computing Science*, pages 400-407. Springer-Verlag, 1989.

[7] S. Brands. Rethinking Public Key Infrastructures and Digital Certificates-Building in Privacy. *MIT Press*, p.356, 2000.

[8] A. Byrne, N. Meloni, F. Crowe, W.P. Marnane, A. Tisserand, and E.M. Popovici. SPA Resistant Elliptic Curve Cryptosystem using Addition Chains. *International Conference on Information Technology-ITNG'07*, pp.995-1000, 2007.

[9] H.Cohen, A. Miyaji, and T. Ono. Efficient elliptic curve exponentiation using mixed coordinates. *Advances in Cryptology-ASIACRYPT'98*, volume 1514 of *Lecture Notes in Computing Science*, pages 51-65. Springer-Verlag, 1998.

[10] E. Dahmen, K. Okeya, and T. Takagi. A New Upper Bound for the Minimal Density of Joint Representations in Elliptic Curve Cryptosystems. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E90-A, No.5, pp.1-9, May 2007.

[11] N. Demytko. A new elliptic curve based analogue of RSA $\mathbb{Z}_n$. *Advances in Cryptology-EUROCRYPT'93*, volume 765 of *Lecture Notes in Computing Science*, pages 40-49. Springer-Verlag, 1994.

[12] W. Diffie, and M.E. Hellman. New directions in cyptography. *IEEE Trans. Inform. Theory* 22(6) pp. 644-654.

[13] T. ElGamal. A public key cryptosystem and signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, vol.31, pp.469-472, 1985.

[14] K. Fong, D. Hankerson, J. Lòpez, and A. Menezes. Field inversion and point halving revisited. *IEEE Transactions on Computers*, 53(8):1047-1059, Aug.2004.

[15] D.M. Gordon. A survey of fast exponentiation methods. *Journal of Algorithms*, vol.27, pp.129-146, 1998.

[16] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography.* Springer-Verlag, 2004.

[17] IEEE P1363-2000. Standard specifications for public key cryptography. *IEEE Transactions on Information Theory*, vol.39, pp.1639-1646, 1993.

[18] D. Kahn. *The Codebreakers*. Macmillan, new York, 1967.

[19] A.W. Knapp. Elliptic curves, volume 40 of *Mathematical Notes*. Princeton University Press, Princeton, NJ, 1992.

[20] D. Knuth. Fundamental Algorithms. *The Art of Computer Programming*, volume 1, Addision-Wesley,(1981).

[21] D. Knuth. Seminumerical Algorithm (arithmetic). *The Art of Computer Programming*, volume 2, Addision-Wesley,(1981).

[22] K. Kobayashi, H. Morita, and M. Hakuta. Multiple Scalar-Multiplication Algorithm over Elliptic Curve. *IEICE Transactions on Information and System*, E84-D, No.2, pp.271-276, Feb.2001.

[23] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203-209, Jan. 1987.

[24] N. Koblitz. Introduction to elliptic curves and modular forms. volume 97 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition, 1993.

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

[26] P. Kocher, J. Jaffe, B. Jun. Differential power analysis. volume 1666 of *Lecture Notes in Computing Science*, pages 388-397. Springer-Verlag, 1999.

[27] K. Koyama, U. Maurer, T. Okamoto, and S.A. Vanstone. New public key scheme based on ellipticc curves over the ring $\mathbb{Z}_n$. *Advances in Cryptology-CRYPTO'91*, volume 576 of *Lecture Notes in Computing Science*, pages 252-266. Springer-Verlag, 1992.

[28] K. Koyama and T. Tsuroaka. Speeding up elliptic curve cryptosystems by using a signed binary window method. *Advances in Cryptology - CRYPTO'92*, volume 740 of *Lecture Notes in Computing Science*, pages 345-357. Springer-Verlag, 1993.

[29] C.H. Lim and P.J. Lee. More Flexible Exponentiation with Precomputation. *Advances in Cryptology-CRYPTO'94*, volume 839 of *Lecture Notes in Computing Science*, pages 95-107. Springer-Verlag, 1994.

[30] D.C. Lou, C.C. Chang. An adaptive exponentiation method. *The Journal of Systems and Software*, volume 42, pp.59-69, 1998.

[31] A. Menezes, T. Okamoto, and S.A. Vanstone. Reducing elliptic curve algorithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, vol.39, pp.1639-1646, 1993.

[32] A.J. Menezes, P.C. vanOorschot, and S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[33] MFIPS 186-2. Digital signature standard. *Ffederal Information Processing Standards Publication 186. U.S. Dept. of Commerce/National Institute of Standards and Technology*, 2000.

[34] V.S. Miller. Uses of elliptic curves in cryptography. In H.C. Williams, editor, *Advances in Cryptology-CRYPTO'85*, volume 218 of *Lecture Notes in Computing Science*, pages 417-428. Springer-Verglag, 1986.

[35] B. Möller. Algorithms for Multi-exponentiation. *Selected Areas in Cryptography*, volume 2259 of *Lecture Notes in Computing Science*, pages 165-180. Springer-Verglag, 2001.

[36] R.A. Mollin. *An Introduction to Cryptography.* Chapman Hall/CRC Press, Boca Raton, Florida, 2000.

[37] F. Morain and J. Olivos. Speeding up the computations on an elliptic curve using addition-subtraction chains. *Inform. Theor. Appl.*, volume 24, pp.531-543, 1990.

[38] M. Nicolas. New Point Addition Formulae for ECC Applications. *Arithmetic of Finite Fields*, volume 4547 of *Lecture Notes in Computing Science*, pages 189-201. Springer-Verlag, 2007.

[39] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. *Advances in Cryptology-CRYPTO'92*, volume 740 of *Lecture Notes in Computing Science*, pages 31-53. Springer-Verlag, 1993.

[40] T. Okamoto. Practical identification schemes as secure as the DL and RSA problems. *http://grouper.ieee.org/groups/1363/addendum.html#Okamoto*, March 1999.

[41] R. Schroeppel, H. Orman, S. O'Malley, and O. Spatscheck. Fast key exchange with elliptic curve systems. *Advances in Cryptology - CRYPTO'95*, volume 963 of *Lecture Notes in Computing Science*, pp.43-56. Springer-Verlag, 1995.

[42] J. H. Silverman. The Arithmetic of Elliptic Curves. volume 106 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1986.

[43] J. H. Silverman. Advanced Topics in the Arithmetic of Elliptic Curves. volume 151 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1994.

[44] N. Smart. The Hessian form of an elliptic curve. *CHES2001*, volume 2162 of *Lecture Notes in Computing Science*, pages 118-125. Springer-Verglag, 2001.

[45] J.A. Solinas. Low-weight Binary Representations for Pairs of Integers, University of Waterloo, Technical Report CORR 2001-41, 2001, available at http://www.carc.math.uwaterloo.ca.

[46] S.A. Vanstone and R.J. Zuccherato. Elliptic curve cryptosystems using curves of smooth order over the ring $\mathbb{Z}_n$. *IEEE Transactions on Information Theory*, vol.43, pp.1231-1237, 1997.

[47] N. Vorobiev. *Fibonacci Numbers.* Birkhuser Verlag, 2002.

[48] L.C. Washington. *Elliptic Curves - Number Theory and cryptography.* Chapman Hall/CRC Press, Boca Raton, New York, 2003.

[49] Y. Yacobi. Exponentiating faster with addition chains. *Advances in Cryptology-EUROCRYPT'90*, volume 473 of *Lecture Notes in Computing Science*, pages 222-229. Springer-Verglag, 1990.

[50] S. Yasuyuki and S. Kouichi. Efficient Scalar Multiplications on Elliptic Curves with Direct Computations of Several Doublings. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E84-A, No.1, pp.120-129, 2001.

List of Publications

Refereed International Journals

1. **R.R. Goundar**, K. Shiota, and M. Toyonaga. New Strategy for Doubling-Free Short Addition-Subtraction Chain. *Applied Mathematics & Information Sciences-An International Journal*, vol.2, issue 2, pp.123-133, Dixie W Pub. U.S.A., Jan 2008.

2. **R.R. Goundar**, K. Shiota, and M. Toyonaga. SPA Resistant Scalar Multiplication using Golden Ratio Addition Chain Method. *IAENG-International Journal of Applied Mathematics*. To appear.

3. **R.R. Goundar**. A Novel Multi-Exponentiation Method. *IAENG-International Journal of Applied Mathematics*, pp.89-95, vol.37, issue 2, Dec 2007.

4. **R.R. Goundar**, K. Shiota, and M. Toyonaga. A Novel Method for Elliptic Curve Multi-Scalar Multiplication. *International Journal of Applied Mathematics and Computer Sciences*. To appear.

Conference

K. Shiota, **R.R. Goundar**, and M. Toyonaga. Efficient and Secure Scalar Multiplication for Pairing-Based Cryptosystems. *SJCIEE*, pp.354, 17-24, Sep 2007.