CHAPTER 5

ECDSA OVER GF(p) WITH STATIC SCHEDULING

This chapter suggests an improved methodology for finding the point computation on Elliptic Curve Digital Signature Algorithm (ECDSA) over GF(P) for the simultaneous processing. It also describes e-Transactions through this methodology for high speed operation.

5.1 INTRODUCTION

In recent days, Elliptic Curve Digital Signature Standard (ECDSA) over Prime field GF(p) is used for Short Message Services (SMS), Financial Transactions and exchanging secure personal information on wireless environment (Asvhini a/p Subramaniam et al 2012). It is a combination of ECC and DSA algorithms. It is well suitable for wireless environments because of the following advantages (Forouzan BA 2006).

1. Limited memory space
2. Long lifetime
3. Power Consumption
4. Reusability
5. portability

In June 2009, the National Institute of Standard and Technology (NIST) has published a document called the Federal Information Processing
Standard (FIPS) 186-3 version for Digital Signature Standard (DSS) applications. Gary Locke and Patrick Gallagher (2009) have dealt with the procedures of DSS in FIPS PUB 186-3 through the following two DSS approaches:

1. RSA based DSS
2. ECC based DSS

The RSA (Rivast-Shamir-Adleman) based DSS is not suitable for wireless and smart card applications, because it needs more power for its computation. Next, the key size of RSA is much longer than the key size of ECC. The following Table 5.1 compares the ratio between the number of bits needed to define the RSA key size and the ECC key size to provide the same level security (Xing Zhang et al 2007).

**Table 5.1 Ratio between RSA key size and ECC key size which provides the same security level**

<table>
<thead>
<tr>
<th>RSA</th>
<th>ECC</th>
<th>Key size</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>P</td>
<td>Ratio</td>
</tr>
<tr>
<td>512</td>
<td>112</td>
<td>5:1</td>
</tr>
<tr>
<td>1024</td>
<td>163</td>
<td>6:1</td>
</tr>
<tr>
<td>2048</td>
<td>224</td>
<td>9:1</td>
</tr>
<tr>
<td>3072</td>
<td>256</td>
<td>12:1</td>
</tr>
<tr>
<td>7680</td>
<td>384</td>
<td>20:1</td>
</tr>
<tr>
<td>15360</td>
<td>512</td>
<td>30:1</td>
</tr>
</tbody>
</table>
Today, the 1024-bit RSA cannot be possible for implementation, while 163-bit ECC can do computation. At the same time, the ECDSA algorithm has a lack of implementations and a low speed processing of point computations. Hence, this chapter suggests to use an optimized point multiplication, which is already discussed in chapter 3 for this implementation to speed up the processes.

This chapter has seven sections to recommend the optimized point multiplication on ECDSA over GF(p). The section-5.2 gives the overview of ECDSA over GF(p) and subsequently the section-5.3 explains an innovative implementation for ECDSA over GF(p). The result of this technique is given in the section-5.4 and it is analyzed and compared with the section-5.5. Then, the application is discussed in the section-5.6. Finally, it is concluded in the section-5.7.

5.2 OVERVIEW OF ECDSA OVER GF(p)

ECDSA is an algebraic structure of Elliptic Curve equation, finite field and the different functionality of DSA (Arockia Jansirani et al 2011). It has three procedures known as digital signature generation, verification, and key generation with domain parameters.

5.2.1 ECDSA Domain Parameters

The different domain parameters of ECDSA over GF(p) are denoted as follows (Zhi-Gang Chen et al 2007):

ECDSA_Domain_parameters(q,FR,a,b,{domain_parameter_seed},G, n, h)

where, q—finite field size
FR — basis
a, b— finite field elements that define the equation of the curve
domain parameter seed—an optional bit string that is present if the elliptic curve was randomly generated.

\[ G \rightarrow \text{a base point of prime order (i.e., } G = (xG, yG)), \]

\( n \rightarrow \text{the order of the point } G, \)

\( h \rightarrow \text{the cofactor (which is equal to the order of the curve divided by } n) \)

**Table 5.2 Domain parameters and its corresponding value of GF GF(p) where } p \text{ is a 256 bits prime suggested by NIST**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q )</td>
<td>( 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 )</td>
</tr>
<tr>
<td>FR</td>
<td>256 bits</td>
</tr>
<tr>
<td>( a )</td>
<td>-3</td>
</tr>
<tr>
<td>( b )</td>
<td>0x 5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0CC53B0F63BCE3C3E27D2604B</td>
</tr>
<tr>
<td>domain parameter seed</td>
<td>0x C49D3608 86E70493 6A6678E1 139D26B7 819F7E90</td>
</tr>
<tr>
<td>( n )</td>
<td>0x FFFFFFFFFF00000000 FFFFFFFFFFFFFFFFFF BCE6FAADA7179E84F3B9CAC2FC632551</td>
</tr>
<tr>
<td>( h )</td>
<td>1</td>
</tr>
<tr>
<td>( x )</td>
<td>0x 6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D812DEB33A0F4A13945D898C296</td>
</tr>
<tr>
<td>( y )</td>
<td>0x 4FE342E2 FE1A7F9B 8EE7EB4A 7C0F9E16 2BCE33576B315ECECBB6406837BF51F5</td>
</tr>
<tr>
<td>( G )</td>
<td>(x,y)</td>
</tr>
<tr>
<td>( n \times G )</td>
<td>an origin point of (x,y)</td>
</tr>
</tbody>
</table>
For example, the following parameters in Table 5.2 are considered for ECDSA over GF(p) implementations, where the value of p is a prime denoted with the length of 256 bits. The ECDSA domain parameters have the following characteristics to support for implementation (Mohammad Noor Nabi et al 2007).

1. ECDSA parameters create the common abstract groups.
2. Domain parameter generation should be Complicated and common values of ECDSA, which are assumed from the standardized published list.
3. Domain parameter Validation should be simple and common.
4. Key pair generation should be simple and specific.
5. Key pair validation should be simple and common.

5.2.2 Key Generation

The ECDSA needs the private and public key pairs for the digital signature generation and verification operations. So the key pair is generated based on the following procedure and distributed to communication parties on the secure environment (Asvhiini a/p Subramaniam et al 2012).

1. Define $E_p(a,b)$ where $a, b$ variable, $p$ prime number
2. Select another prime $q$ where $p-1=0(mod)q$
3. Choose an integer $d$ as private key
4. Choose $e_1(x,y)$ from EC point set
5. Find $e_2(x, y)=d \times e_1(x, y)$
6. User private key is $d$ and public key is $(e_1, e_2, p, q, a, b)$
5.2.3 Digital Signature Generation

The signing is a process to generate two signatures based on the original information, and it is sent to another side with the original information shown in Figure 5.1.

Figure 5.1 Different functionality of digital signature generation in ECDSA over GF(p)

**Signing on Sender side:**

1. Choose the random number \( r \) where \( 1 < r < q - 1 \)
2. Find \( P(x, y) = r \times e_1(x, y) \)
3. Find \( S_1 = x \pmod{q} \) from \( p \)
4. Find \( S_2 = (h(M) + d \times S_1) r^{-1} \pmod{q} \)
5. Send \( M, S_1 \) and \( S_2 \) to the receiver side
5.2.4 Digital Signature Verification

The verification is a process on receiver side to separate signatures and message. Then the new signature is regenerated based on the same message and is verified with received signatures shown in Figure 5.2.

Verifying on Receiver Side:

1. Calculate $A = h(M)S_2^{-1}(\mod q)$
2. Calculate $B = S_2^{-1}S_1(\mod q)$
3. Find $T(x,y) = A \times e_1(x,y) + B \times e_2(x,y)$
4. Is $x = S_1(\mod q)$ than verified otherwise rejected

![Diagram of Digital Signature Verification](image)

**Figure 5.2 Different functionality of digital signature verification in ECDSA over GF(p)**

In this proposed methodology, the hash function ($h(m)$) is implemented with SHA-1 to generate a hash code for authentication. The hash code is a 160 bits value with the maximum size of $2^{61}$ bytes. This entire
message is divided into the equal number of blocks and the each block size is assumed as 64 bytes. These blocks are manipulated 80 times iteratively with feed backing of the output block. Besides, the two byte sizes of five registers are used for 4 byte length of word processing to produce output. That is $5 \times 32$ bits = 160 bits (Mohamed Khalil Hani 2006).

5.3 PROPOSED METHODOLOGY FOR ECDSA OVER GF(p)

The EC has a number of consistencies with the DSA to support the implementation ECDSA over prime field. They are listed as follows:

1. Both algorithms support the ElGamal signature scheme.
2. The domain parameters and key pairs of both algorithms are relatively difficult to generate for authentication.
3. In their current version of DSA and ECDSA apply the same version of SHA-1 as the sole cryptographic hash function.

The architecture of ECDSA over GF(p) is defined with four layers based on its operations. The bottom layer is related to the basic prime field operations, which defines the point addition, subtraction, multiplication and division over modulo operation. The second layer is used to define the basic group operations in Elliptic Curve such as the point addition and doubling. The third layer is known as scheduling layer, which helps to perform the point multiplication in parallel. This layer helps to reduce the number of hazards and stalls before execution and it also creates a possibility for parallel processing. The final layer is the top layer known as ECDSA layer, which computes the different point operations of EC over GF(p).

The scalar multiplication is the central operation of ECDSA. The operation of scalar multiplication in the inside branches can be executed in parallel through static scheduling, which is discussed in Chapter 3.2. So the
computation of PA and PD are now done in parallel with the help of a few additional parameters to hold intermediate results. The respective intermediate parameters of PA and PD are not needed throughout execution.

5.4 EXPERIMENTAL SETUP AND RESULT

For experimental, \( y^2 = x^2 + ax + b \) and \( E_p(a,b) \) mentioned in Equations (3.1) and (3.2) are considered with the values of \( p=11111, \ a=1 \) and \( b=1 \). The same point \( (x=0, y=1) \) is assumed for both point multiplications. To evaluate, the performance of the proposed work, the system with 2.33 Ghz Intel processor and 2 GB RAM are considered. Besides, the message digest is simulated through SHA-1 through online (US FIPS PUB 180-3 2013). The point multiplication for digital signing and digital verification of ECDSA over \( GF(p) \) are implemented with ‘C’ language under Ubuntu OS.

The following information is considered as input for SHA-1 to compute the message authentication code \( h(m) \).

Message:

“Dear Sir / Madam,

Your NEFT Fund transfer with UTR FDRLH13043053547 for Rs. 20000.00 has been credited on 12/02/2013 at 15:33:24

Thank you for banking with Federal Bank”

Output-\( h(m) \):

\[ 97a73a8d3ad1a3aded5ea75d0cea79d3215cd901 \]

The character of output information is converted into EC points through mapping techniques for ECDSA point computations to produce
signatures $S_1$ and $S_2$. It is encrypted with the public key of receiver and private key of sender to produce digital signature. These signatures are attached with the message and transmitted to sender side.

$$(M, S_1, S_2) = (M, x \, (\text{mod}) \, q \, \text{from} \, p, (h(M) + d \times S_1)r^{-1} \, (\text{mod}) \, q)$$

$M$ = Message

$S_1 = x \, (\text{mod}) \, q \, \text{from} \, p$

$S_2 = (h(M) + d \times S_1)r^{-1} \, (\text{mod}) \, q$

The receiver verifies the received information $(M, S_1, S_2)$ and regenerates signature to validate the same. Then, it is validated by using sender the public key $(e_1, e_2, p, q, a, b)$ and the private key $(d)$ of receiver.

It is mathematically validated by:

$$T(x,y) = A.e_1(x,y) + B.e_2(x,y)$$

But, the value of $A = (h(m) / S_2) \, \text{mod} \, q$ and $B = (S_1 / S_2) \, \text{mod} \, q$ are substituted in the above equation.

So, $T(x,y) = e_1(x,y).(h(m) / S_2) \, \text{mod} \, q + e_2(x,y) \, (S_1 / S_2) \, \text{mod} \, q$

$= e_1(x,y).(h(m) \, \text{mod} \, q / S_2) + d \times e_1(x,y). (S_1 / S_2) \, \text{mod} \, q$

$= e_1(x,y) / S_2(h(m) + S_1 \times d) \, \text{mod} \, q$

$= e_1(x,y) / S_2(S_2.r) \, \text{mod} \, q$

$= e_1(x,y) \times r \, \text{mod} \, q$

$= P(u,v) \, \text{mod} \, q$

$= u \, \text{mod} \, q$ which is equal to $x$ then it is first signature

$= S_1(\text{Accepted})$ otherwise it is rejected
5.5 PERFORMANCE ANALYSIS

The performance of proposed ECDSA over GF(p) is mainly based on the number of point multiplications. The number of point multiplications in ECDSA over GF(p) is denoted in the following Table 5.3.

Table 5.3 Number of point multiplication needed for the various operations in ECDSA over GF(p)

<table>
<thead>
<tr>
<th>ECD A Operation Name</th>
<th>Equation</th>
<th>Number of Point Multiplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key generation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verifying</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Number of point multiplication in ECDSA</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

The number of point multiplications is scheduled through proposed methodology for parallel execution. Here, the execution time is based on the number of processors. When the dual processors are used, the 50% of work is allocated for each processor. This proposed code scheduling helps to minimize the different amount of dependencies in the ECDSA over prime field, which are identified from Tables 5.3 and 3.8.

The two point multiplications in digital signing and three point multiplications in digital verification are reduced through with dual processor. Here, each processor is reducing the data dependencies from 24% to 21.5%, the control dependencies from 12.5% to 4% and the loop carried dependencies from 16.5% to 4.5% simultaneously.
5.6 DISCUSSION AND APPLICATION

The Digital Signature is used in the Electronic Mail, Electronic Funds Transfer, Software Distribution, Data Storage, SSL (Secure Sockets Level), PGP (Pretty Good Privacy), XML Signatures and other applications, which require data integrity assurance and data origin authentication (Constantin Popescu 2011). It is also used for sending SMS (Short Message Services) on wireless network or mobile devices, because the ECDSA key sizes are so much shorter than RSA keys to provide same level security. So the lengths of the public and private keys are much shorter in ECDSA. Besides, it is used for faster processing in lower demands of memory and bandwidth (Zhu Liehuang et al 2006).

The ECDSA is faster for the signing and decryption procedures, but slower in the signature verification and encryption. If it is implemented with the proposed methodology, the speed of the signature verification and encryption will be increased to support Fast Electronic Fund Transfers, Supply Chain Management, Internet Marketing, Online Transaction Processing, and Electronic Data Interchange.

5.7 SUMMARY

This proposed methodology for Elliptic Curve Digital Signature Algorithm (ECDSA) promises a better alternative for RSA based DSA. It is the highest strength per bit of any public-key cryptography system known today. Besides, it will speed up the ECDSA over prime field based applications. This methodology brings the algorithms within the view of industry standards. But, the design of ECDSA over GF(p) for parallel computation is not an easy task. It has more mathematics complexity to implement the same. This efficient implementation of ECDSA over GF(p) supports the stronger and more faster processing based on the small key size, high performance, lower computational cost, and code scheduling.