FPGA IMPLEMENTATION OF ELLIPTIC CURVE CRYPTOGRAPHY

Lavanya .P 1, Prof. Leelavathi .G 2, Dr. Siva S. Yellampalli 3

1,2,3Department of VLSI Design and Embedded Systems, UTL Technologies, VTU Extension Centre, (India)

ABSTRACT

The rising growth of data communication and electronic transactions over the internet has made security to become the most important issue over the network. The widely used algorithms for public-key cryptosystems are RSA, Diffie-Hellman key agreement, the digital signature algorithm and systems based on elliptic curve cryptography (ECC). ECC offer the smallest key size and highest strength per bit compared to any other public key cryptosystem, since there is currently no known sub-exponential time algorithm to solve the discrete logarithm problem. Smaller key sizes make them highly suitable for hardware implementation on FPGAs.

The main objective of the project is to design and develop the Montgomery Modular Multiplier for ECC cryptosystem system with optimized design for increasing the efficiency of the system by optimizing the area and throughput parameters based on FPGA.

Keywords: Cryptography, ECC, Montgomery modular multiplier, FPGA, Atrix.

I. INTRODUCTION

As the Internet and other forms of electronic communication become more prevalent, electronic security is becoming increasingly important. Cryptography is used to protect e-mail messages, credit card information, and corporate data. One of the most popular cryptography systems used on the Internet is Pretty Good Privacy because it's effective and free.

Cryptography is used for confidentiality, authentication, data integrity, and non-repudiation, which can be divided into two families: secret-key cryptography and public-key cryptography. Secret-key cryptography which usually has a relatively compact architecture and smaller key size than public-key cryptography is often used to encrypt/decrypt sensitive information or documents. Public-key cryptography offers fundamental technology for key agreement, encryption/decryption (two keys), and digital signatures. In contrast to secret-key cryptography, public-key cryptography usually has a lower throughput rate and more complicated computation. Modern cryptography intersects the disciplines of mathematics, computer science, and electrical engineering. Applications of cryptography include ATM cards, computer passwords, and electronic commerce.
In recent years the need for secure communication over computer networks has grown significantly, especially with the widespread use of a possibly transparent medium like Internet for online banking and other forms of e-commerce. Such applications use public key cryptosystems like RSA and Elliptic Curve Cryptosystem (ECC) [1][2].

Elliptic Curve Cryptosystems are emerging as a new generation of cryptosystems based on public key cryptography. They offer the smallest key size and highest strength per bit compared to any other public key cryptosystem, since there is currently no known sub-exponential time algorithm to solve the discrete logarithm problem. Smaller key sizes make them highly suitable for hardware implementation on FPGAs.

II. ELLIPTIC CURVE CRYPTOGRAPHY

In 1985, Elliptic Curve Cryptography (ECC) was proposed independently by cryptographers Victor Miller (IBM) and Neal Koblitz (University of Washington). ECC is based on the difficulty of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP). Like the prime factorization problem, ECDLP is another "hard" problem that is deceptively simple to state: Given two points, P and Q, on an elliptic curve, find the integer n, if it exists, such that P = nQ.

ECC may be employed with many Internet standards, including CCITT X.509 certificates and certificate revocation lists (CRLs), Internet Key Exchange (IKE), Transport Layer Security (TLS), XML signatures, and applications or protocols based on the cryptographic message syntax (CMS).

RSA had been the mainstay of PKC for over a quarter-century. ECC, however, is emerging as a replacement in some environments because it provides similar levels of security compared to RSA but with significantly reduced key sizes. NIST use the following table to demonstrate the key size relationship between ECC and RSA, and the appropriate choice of AES key size as shown in TABLE 1.

<table>
<thead>
<tr>
<th>ECC Key Size</th>
<th>RSA Key Size</th>
<th>Key-Size Ratio</th>
<th>AES Key Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>163</td>
<td>1,024</td>
<td>1:6</td>
<td>n/a</td>
</tr>
<tr>
<td>256</td>
<td>3,072</td>
<td>1:12</td>
<td>128</td>
</tr>
<tr>
<td>384</td>
<td>7,680</td>
<td>1:20</td>
<td>192</td>
</tr>
<tr>
<td>512</td>
<td>15,360</td>
<td>1:30</td>
<td>256</td>
</tr>
</tbody>
</table>

Key sizes in bits.  

Source: Certicom, NIST

TABLE 1: Comparison of Key sizes
III. ENCRYPTION & DECRYPTION

This scheme will be demonstrated using Alice and Bob as sender and receiver of a secret message, respectively. Typically, the message consists of some large secret number, which is subsequently used by the two parties to open a conventional secure communication channel. The coordinates of the points on the elliptic curve itself serve as a pool of numbers to choose from.

3.1 The Encryption Operation

Step 1: Bob chooses a point \( \alpha \) on an elliptic curve \( E \) over some \( \mathbb{Z}_p \) and an integer \( z \) between 1 and the order of the abelian group \( E \).

Step 2: Bob computes \( \beta = z \alpha \) on the curve and publishes \( \alpha, \beta, E, \) and \( p \). He keeps his private key \( z \) secret.

Step 3: Suppose Alice wants to send a message to Bob. Alice picks an integer \( k \) between 1 and the order of \( E \), which will be her private key.

Step 4: To encrypt a message, Alice looks up Bob’s public key. As the message, she selects a point \( x \) on the elliptic curve \( E \). Next, Alice performs the following encryption operation to encrypt the message:

\[
e_k(x, k) = (k\alpha, x + k\beta) = (y_1, y_2).
\]

The encrypted message is \( y = (y_1, y_2) \); it includes Alice’s public key \( y_1 \).

3.2 The Decryption Operation

Step 5: Alice sends Bob the encrypted message. To decrypt the message, Bob uses the decryption operation:

\[
d_z(y_1, y_2) = y_2 - zy_1 = (x + k\beta) - z(k\alpha) = x + k(z\alpha) - z(k\alpha) = x,
\]

where \( z \) is Bob’s private key.

Note the interlocking of public and private keys here: Bob’s private key \( z \) will decrypt this message correctly, because it matches his public key \( \beta = z\alpha \), and he can be sure that it was Alice who transmitted this message, since nobody else is in possession of the private key \( k \) that matches her public key \( y_1 = k\alpha \).

IV. MONTGOMERY’S POINT MULTIPLICATION

Point multiplication is to compute \( kP \), where \( k \) is an integer and \( P \) is a point on an elliptic curve \( E \) defined over a field \( \mathbb{F}_q \). Point multiplication is also called scalar multiplication, and it dominates the execution time of elliptic curve cryptographic schemes. There are several algorithms for point multiplication over elliptic curve. The Montgomery point multiplication algorithm used in our work. Block diagram for point multiplier is shown in Fig. 1.
V. HARDWARE AND SOFTWARE IMPLEMENTATION OF ECC

5.1 Software Requirement Specification
- Operating System: Windows XP with SP2
- Synthesis Tool: Xilinx 12.2.
- Simulation Tool: Modelsim6.3c.

5.2 Hardware Requirement Specification
- Minimum Intel Pentium IV Processor
- Primary memory: 2 GB RAM
- Artix-7 FPGA
- Artix 7 FPGA development board
- Device- XC7A100T-3 CSG324
- USB Supple cable

V. VHDL SIMULATION RESULTS
The functional working of the whole system was verified including simulations and tests of each module. The results obtained after VHDL implementation of ECC algorithm with Montgomery multiplication at various stages of implementation is summarized below:

To evaluate the results, encryption and decryption of the message is implemented in VHDL. The key based information hiding are analyzed, synthesized and simulated with different input patterns. Fig. 2 shows the snapshot of schematic view of entire code represented in VHDL. It consists of 11 bit data i.e, data (0:10), input of 112 bit i.e (0:111), reset input with rst of single bit, clock signal given by clk.
For a 11 bit input data “0DE” and 11 bit key “6AD”, Fig. 3 shows the timing diagram and encryption and decryption results. Fig. 4 shows the simulation clock frequency of 403.177 MHz and time 2.480 ηSec and Fig. 5 shows the area utilization of 846 slices and 0% of the device utilization for the same 11 bit input data and key size.
For a 112 bit input data “1bd5bbd5bbd5bbd5bbd5bbd5bbd5" and 112 bit key “deaddeaddeaddeaddeaddeaddeaddead”, Fig. 6 shows the timing diagram and encryption and decryption results. Fig.7 shows the simulation clock frequency of 362.884 MHz and time 2.756 ηSec and Fig.8 shows the area utilization of 5722 slices and 4% of the device utilization for the same 112 bit input data and key size.
VI. SUMMARY OF RESULTS

<table>
<thead>
<tr>
<th>Device</th>
<th>Key size (Bits)</th>
<th>Clock (MHz)</th>
<th>Time (ȠSec)</th>
<th>Area (slices)</th>
<th>Usage (%)</th>
<th>Encryption time (ms)</th>
<th>Decryption time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atrix-7</td>
<td>112</td>
<td>362.884</td>
<td>2.756</td>
<td>5722</td>
<td>4%</td>
<td>3.94</td>
<td>3.96</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>403.177</td>
<td>2.480</td>
<td>846</td>
<td>0%</td>
<td>4.37</td>
<td>4.40</td>
</tr>
</tbody>
</table>

**TABLE 2: ECC simulation summary of results**
For a key length of 112 bits, the simulation time taken for encryption is 3.94 ms and decryption time is 3.96 ms. While, the clock frequency and time is 362.884 MHz and 2.756 μs. 5722 slices of area are utilized contributing to 6% of the total area on the device.

For a key length of 11 bits, the simulation time taken for encryption is 4.37 ms and decryption time is 4.4 ms. While, the clock frequency and time is 403.177 MHz and 2.48 μs. 846 slices of area are utilized contributing to 0% of the total area on the device.

VII. CONCLUSION AND FUTURE SCOPE

We implemented all our designs on Xilinx Artix-7 device. We used Model Sim simulator and the Xilinx ISE toolset. Table 6.1 presents our results on the Artix-7 FPGA. No partitioning was required for 112-bit Montgomery point multiplier, comfortably fitting well within the FPGA. Hence we have presented an efficient technique for implementing the Elliptic Curve Cryptographic Scheme on Artix-7 FPGA.

Future enhancement of this work is to accommodate several other modules, as the device utilization is very less. Also the security can also be enhanced by increasing the key size without compromising on the area and time constraints.

REFERENCES


