CHAPTER 6

ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM AND CERTIFICATE

6.1 INTRODUCTION

Message authentication protects two parties who exchange messages from any third party. However, message authentication does not protect the two parties against each other. In situations where the sender and receiver do not trust each other, Digital Signatures are required in addition to message authentication. A digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature. The signature is formed by taking the hash of the message and encrypting the message with the private key of the creator. This signature guarantees the source and the integrity of the message. Thus, digital signatures are used to detect unauthorized users from modification of the data and also to authenticate the identity of the signatory. In addition, the recipient of signed data can use a digital signature in proving to a third party that the signature was in fact generated by the signatory.

A Digital Signature Algorithm (DSA) includes digital signature generation and signature verification processes. A signatory uses the generation process to generate a digital signature on data and a verifier uses a process to verify the authenticity of the signature. Each signatory has a public and private key. The private key is used in the signature generation process. The signatory termed as the key pair owner is the only entity, authorized to use the private key to generate digital signatures. The private key should remain secret so as to prevent other entities from claiming to be the key pair owner and also in using the private key to generate fraudulent signatures. Hence, the private key should be known only by the key pair owner.
This module of the work focuses in reducing the generation and verification timings of Elliptic Curve Digital Signature Algorithm (ECDSA) [74]. Integration of the developed high speed and less complex ECC algorithm for signature generation and verification process has been carried out. Their performance in terms of improvement in signature generation and verification timings in comparison with RSA and DSA were analyzed. Energy cost analysis of the signature algorithms and also their performance timings for different hash functions were also simulated and compared. A hybrid certificate was also proposed where the expensive Elliptic Curve scalar multiplication operations are offloaded to the security manager thereby improving the certificate generation and verification timings.

6.2 REQUIREMENTS FOR DIGITAL SIGNATURES

Approved digital signature algorithms are designed to prevent an adversary from generating the same signature as the signatory on a different message [75]. In other words, signatures are designed so that they cannot be forged. Public key is used in the signature verification process. This public key need not be kept secret but its integrity should be maintained. Anyone can verify a correctly signed message using the public key. For both the signature generation and verification processes, the signed message is compressed by means of an approved hash function [99]. Both the uncompressed message and the digital signature are made available to a verifier. A verifier requires an assurance that the public key belongs to the entity that claims to have generated a digital signature and also possesses the associated private key.

A digital signature requires a string of binary bits which depends on the message being signed. It is computed using a set of rules and a set of parameters that permits the identity of the signatory and the integrity of the data to be verified. The input for DSA to generate the digital signature is termed as a message digest.

Digital signatures require two levels of functionality. At the lower level, there should be a function that produces an authentication which is a value that can be used to authenticate a message. This lower level function should be used at the higher level to enable the receiver to verify the authenticity of the message. Hash function is used to produce an authenticator. A hash function accepts a variable size
message ‘M’ as input and produces a fixed size output referred to as a hash code ‘H(M)’ or an authenticator. The hash code is also referred to as a message digest or hash value. The hash code is a function of all the bits of the message and provides an error detection capability. A change to any bit or bits in the message results in a change in the hash code.

FIPS 180-2 [100] approved digital signature algorithms are widely used with an appropriate hash function as specified in the Secure Hash Standard (SHS). The digital signature is provided to the intended verifier along with the signed data. The verifying entity verifies the signature by using the public key and the same hash function that was used to generate the signature. Digital signatures intend to achieve three important security services as given below:

- **Authentication**: It is concerned with assurance of identity.
- **Data Integrity**: It is the assurance that data has not been modified since the signature was applied. Digital signatures provide excellent data integrity services by virtue of the digital signature value being a function of the message digest. Even the slightest modification of a digitally signed message would result in failure of the signature verification process.
- **Non-Repudiation**: It is concerned with providing evidence to a third party that a party participated in a transaction and thereby protects other parties in the transaction against false denial of participation.

### 6.3 ECDSA GENERATION AND VERIFICATION

The ECDSA is a variant of RSA and DSA which operates on Elliptic Curve groups [101]. The developed high speed and less complex ECC algorithm has been integrated in ECDSA. The proposed EC variant not only provides smaller key sizes for the same security level but achieves significant improvement in the ECDSA generation and verification timings.
6.3.1 Key Generation and Signature Generation Algorithm

Initially, the curve parameters \((n, P, d, Q, h)\) must be agreed upon.

- ECDSA key generation. Each entity \(A\) does the following:
  
  i) Select an elliptic curve \(E\) defined over \(F_{2^m}\). The number of points in \(E(F_{2^m})\) should be divisible by a large prime \(n\).
  
  ii) Select a point \(P \in E(F_{2^m})\) of order \(n\).
  
  iii) Select a statistically unique and unpredictable integer \(d\) in the interval \([1, n-1]\).
  
  iv) Compute \(Q = dP\)
  
  v) \(A\)'s public key is \((E, P, n, Q)\); \(A\)'s private key is \(d\).

The block diagram representation of ECDSA signature generation algorithm is illustrated in Figure 6.1.

![Block Diagram of Signature Generation Algorithm](image)

Figure 6.1 Block Diagram of Signature Generation Algorithm
• ECDSA signature generation. To sign a message m, A does the following:
  i) Select a statistically unique and unpredictable integer k in the interval \([1, n - 1]\).
  ii) Compute \(kP = (x, y)\) and \(r = x \mod n\).

       If \(r = 0\), then go to step 1.

  iii) Compute \(k^{-1} \mod n\).

  iv) Compute \(s = k^{-1}(h(m) + dr) \mod n\), where h is the secure Hash Algorithm (SHA-1).

  v) If \(s = 0\), then go to step 1.

  vi) The signature for the message m is the pair of integers \((r, s)\).

6.3.2 Signature Verification Algorithm

Figure 6.2 illustrates the block diagram representation of the ECDSA signature verification algorithm. To verify A's signature \((r, s)\) on m, B should do the following:

  i) Obtain an authentic of A's public key \((E, P, n, Q)\). Verify that r and s are integers in the interval \([1, n - 1]\).

  ii) Compute \(w = s^{-1} \mod n\) and \(h(m)\).

  iii) Compute \(u_1 = h(m) w \mod n\) and \(u_2 = rw \mod n\).

  iv) Compute \(u_1P + u_2Q = (x_0, y_0)\) and \(v = x_0 \mod n\).

  v) Accept the signature if and only if \(v = r\).
6.4 PERFORMANCE OF DIGITAL SIGNATURE ALGORITHMS

ECC multiplication in the Ring is faster than multiplication in a field defined by pentanomials. This reduced time complexity in multiplication improves the signature generation and verification timings in ECDSA. The timings and energy cost for key generation, signature generation and verification for RSA, DSA and ECDSA were simulated and analyzed. Comparisons of DSA and ECDSA for various hash methods were also performed and tabulated.
6.4.1 Timing Analysis of RSA, DSA and ECDSA

Figures 6.3, 6.4 and 6.5 illustrate the timing analysis of the key generation, signature generation and verification for RSA, DSA and ECDSA respectively. Key generation timings for ECC outperforms RSA and DSA for all key sizes and is apparent as the key sizes increases. For RSA and DSA, the key sizes used for simulation were 512, 1024 and 2048 bits. In the case of ECC, the key sizes used for simulation were 128,163 and 193 bits. It was inferred form the simulation that ECC key generation time increases linearly with key size when compared to RSA which increases exponentially.

![Graph](image)

**Figure 6.3** RSA Key Generation, Signature Generation and Verification Timings
Figure 6.4 DSA Key Generation, Signature Generation and Verification Timings

Figure 6.5 ECDSA Key Generation, Signature Generation and Verification Timings
6.4.2 Energy Cost of Digital Signature Algorithms

Figures 6.6, 6.7 and 6.8 illustrate the comparison of the energy cost for generating and verifying RSA, DSA and ECDSA. The 163 bit key size for ECDSA is compared with the proven equivalent 1024 bit key size for RSA. The energy values are reported for the three main steps associated with digital signature algorithms viz., key generation, signature creation (Sign) and signature verification (Verify).

From the analysis of results, it is found that ECDSA consumes less energy than DSA. However, ECDSA and RSA digital signature algorithms have complementary energy costs. RSA performs signature verification efficiently while ECDSA imposes a smaller cost for signature generation.

![Energy Cost of RSA Key Generation, Signature Generation and Verification](image)

Figure 6.6 Energy Cost of RSA Key Generation, Signature Generation and Verification
Figure 6.7  
Energy Cost of DSA Key Generation, Signature Generation and Verification

Figure 6.8  
Energy Cost of ECDSA Key Generation, Signature Generation and Verification
The difference between the energy costs of signature generation and verification in RSA is much greater than in ECDSA. If a mobile client is required to perform frequent signature generation, then it seems preferable to use ECDSA. On the other hand, if the frequency of signature verification is greater than signature generation, then RSA digital signature algorithm can be employed.

6.5 DIGITAL CERTIFICATES

Digital Certificate [102] is simply a public key together with the device ID and certification expiration date signed by CA. To prevent impersonation attack, certificates are used in key establishment protocol. Certificate provides a mechanism to check cryptographically to whom the public key belongs and if the device is a legitimate member of a particular network.

The computation complexity and power consumption of symmetric key based protocols are negligible when compared to public key operations. However, the key management for symmetric key based protocols is complicated and is always subject to attack by adversaries.

ECC is used in the proposed protocol to perform security functions on sensors with limited computing resources. Compared to other public key crypto algorithms, much smaller key sizes are required with ECC to provide a desired level of security at faster processing speed and smaller key storage requirements.

The use of a trusted interface to pre-establish a certificate and root key in a device thwarts both active and passive attacks in subsequent key establishment protocols. The certificates are acquired before each device joins the network through an out-of-band interface. The Elliptic Curve implicit certificate scheme is used [103] because of the resulting low communication complexity which is a dominant factor for low bit transmission channels in sensor networks.
6.5.1 Implicit Certificate Mutual Authentication

First, an Elliptic Curve \( E \) defined over \( \text{GF}(2^m) \) of large order \( n \) is selected and made public to all users. As before, given an integer \( k \) and a point \( P \in \text{GF}(2^m) \), scalar multiplication is the process of adding \( P \) to itself \( k \) times. CA selects a random integer \( q_{CA} \) as its static private key and computes the static public key \( Q_{CA} = q_{CA} \times P \). To obtain a certificate and the static private-public key pair, the sensor \( U \) randomly selects a temporary key pair \( (g_U, G_U) \) and sends \( G_U \) to CA via a secure out-of-band interface. CA verifies \( U \)'s identity and the authenticity of the request received from \( U \). CA also selects a temporary key pair \( (g_{CA}, G_{CA}) \) and computes the Elliptic Curve point \( B_U = G_U + G_{CA} \).

![Figure 6.9 Implicit Certificate Generation and Verification](image)

Figure 6.9 illustrates the implicit method of certificate generation and verification process. The implicit certificate \( IC_U \) for \( U \) is constructed as the
concatenation of CA's static public key $Q_{CA}$, the device identity $ID_U$, the Elliptic Curve point $B_U$ and the certificate expiration date $t_U$, i.e., $(Q_{CA}, ID_U, B_U, t_U)$. CA then applies a one-way hash function $H$ on $IC_U$ and derives an integer $e \in [2, n - 2]$ from $H(IC_U)$ following the conversion routine.

Finally, CA computes U's private key reconstruction data $s_U = g_{CA}e_U + q_{CA} \pmod{n}$. U's public key $Q_U = e_U B_U + Q_{CA}$, and sends $s_U$ and $IC_U$ back to U. After U receives the implicit certificate from CA, it computes the hash value $H(IC_U)$ and derives an integer $e_U$ from $H(IC_U)$. U also computes its static private key $q_U = s_U + g_{U} e_{U} \pmod{n}$ and public key $Q_U = q_U \times P$. U then proceeds to reconstruct the public key $Q'_U = e_U B_U + Q_{CA}$. If $Q'_U = Q$, U accepts the certificate and outputs the static key pair $(q_U, Q_U)$; otherwise it rejects the certificate. By repeating the same process, the security manager V acquires its certificate $IC_V$ and static key pair $(q_V, Q_V)$.

6.5.2 Proposed Hybrid Certificate Generation Algorithm

The hybrid algorithm focuses on reducing the Elliptic Curve random point scalar multiplications on sensor nodes by offloading the computation burden to more powerful security managers and replacing the expensive public key operations by efficient symmetric key operations.

Figure 6.10 depicts the certificate generation process using ECDSA algorithm. The sensor U randomly chooses integer $q_U \in [2, n-2]$ as its private key and computes the public key $Q_U = q_U \times P$.

In order to receive a certificate, the sensor sends its public key $Q_U$ together with its user identity through an out-of-band secure interface to CA. CA uses its private key $q_{CA}$ to sign the hashed value of the concatenation of the public key, the device identity $ID_U$ and the certification expiration date $t_U$. The CA then sends the signed message $(r_U, s_U)$ together with its public key $Q_U$ through the secure channel to the terminal. By repeating the same process, the security manager V acquires its certificate $(r_V, s_V)$.
The certificate generation processes for sensor U and security manager V are performed offline and before they join the network. At the beginning of the hybrid key establishment protocol both U and V send to the other side their public key, device ID, certificate and the expiration time. The expensive public key decryption and Elliptic Curve scalar multiplication of a random point are all performed by the security manager which is more computationally powerful. Then the mutual certificate authentication between the sensor and the security manager is executed.

Figure 6.11 illustrates the hybrid method of mutual authentication and certificate verification. In real time execution of the hybrid key establishment protocol, the sensor is required to compute three Elliptic Curve scalar multiplication of fixed points that is two for verifying the ECDSA signature and another one for
generating the ephemeral key. The other operations performed are one symmetric key decryption, one modular multiplication, one modular squaring, one modular addition, one hash, one key derivation and two random number generations. The expensive public key decryption and Elliptic Curve scalar multiplication of a random point are all moved to the security manager side.

![Figure 6.11 Mutual Authentication and Certificate Verification](image)

### 6.6 PERFORMANCE OF DIGITAL CERTIFICATES

The time taken for certificate generation and verification is compared between the Implicit and Hybrid Algorithms using various hash methods viz., MD2, MD5, SHA-1, SHA-256, SHA-384 and SHA-512. This helps us to prove which Hash method would be suitable for the Certificate Authority to issue certificates. The results of simulations carried out in MATLAB 7 are presented in Table 6.1. Better generation and verification timings were obtained by integrating ECC multiplication using Ring.
Table 6.1 Comparison of Implicit and Hybrid algorithms using different Hash Methods

<table>
<thead>
<tr>
<th>HASH METHOD</th>
<th>CERTIFICATE GENERATION (seconds)</th>
<th>CERTIFICATE VERIFICATION (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Implicit</td>
<td>Hybrid</td>
</tr>
<tr>
<td>MD 2</td>
<td>0.273</td>
<td>0.187</td>
</tr>
<tr>
<td>MD 5</td>
<td>0.255</td>
<td>0.204</td>
</tr>
<tr>
<td>SHA-1</td>
<td>0.229</td>
<td>0.162</td>
</tr>
<tr>
<td>SHA-256</td>
<td>0.275</td>
<td>0.198</td>
</tr>
<tr>
<td>SHA-384</td>
<td>0.298</td>
<td>0.209</td>
</tr>
<tr>
<td>SHA-512</td>
<td>0.311</td>
<td>0.267</td>
</tr>
</tbody>
</table>

Using Table 6.1, it is found that when compared to the Implicit scheme, the Hybrid scheme achieves significant improvement in terms of timing. Also with respect to certificate generation/verification process of a sensor node, SHA-1 is found to be economical in terms of timing.

6.7 CONCLUSION

ECDSA was simulated using the developed high speed and less complex ECC algorithm. The simulations were carried out for different hash functions so as to make a comparison of their timing performances. The proposed ECDSA was also compared with RSA and DSA signature generation and verification algorithms. ECDSA with SHA-1 resulted in the best signature generation and verification timings. A hybrid digital certificate using ECC was also proposed. This certificate was found to be the best in a resource constrained environment like the sensor network.