CHAPTER 4

VERIFIABLE ENCRYPTION OF AN ELLIPTIC CURVE DIGITAL SIGNATURE

4.1 INTRODUCTION

This chapter addresses the Verifiable Encryption of Elliptic Curve Digital Signature. The protocol presented is an Adjudicated protocol, that is, the Trusted Third Party (TTP) takes part in the protocol only when there is a dispute. This scheme can be used to build efficient fair exchanges and certified e-mail protocols for contract signing.

Exchanging items over the Internet is considered as a difficult problem, and it is called as the *fair exchange proble*m. For example when a message is sent over the Internet, there is no assurance that it will be delivered to the intended recipient. Even if the message has been delivered, the recipient may claim otherwise. Giuseppe Ateniese (2004) presented the Verifiable Encryption of RSA signature scheme, Cramer-Shoup signature scheme and Schnorr signature scheme which is used for the certified electronic mail protocols. There have been several approaches to solve the fair exchange problem which are based on different definitions of fairness. Fairness is interpreted as *equal computational effort* by Even et al (1985). In this paper, it is assumed that two parties, Alice and Bob, have equal computational power and they exchange their items bit-by-bit by taking turn. This approach does not require the intervention of a trusted third party but it involves many rounds of interactions. A probabilistic approach is adopted by Ben-Or et al

(1990), and in his paper, the probability of successfully completing the protocol is gradually increased after every round of interaction. Asokan et al (2000) introduced the *optimistic* approach. It relies on the existence of a trusted third party which is invoked only in case of an exception. As long as the two parties follow the exchange protocol, there is no need for the trusted party's intervention, but if one deviates from the protocol then the trusted party can easily restore fairness. This approach results in particularly efficient fair exchange protocols for generic items. Asokan et al (1998) and Bao et al (1998) have built fair exchange protocols by means of *verifiable encryption* of digital signatures. Camenisch and Damgard (2000) generalized the schemes given by Asokan et al (1998) to achieve more efficient schemes that can be proved secure without relying on random oracles.

4.2 PROBLEM DEFINITION

Suppose Alice and Bob wanted to sign a contract. Let us assume that they were communicating via electronic mail. We know that the National Stock Exchange (NSE) will fluctuate with respect to time. It may rise or go down depend on so many factors. Let us assume that Alice e-mailed to Bob stating the message that

"*I will bet you Rs.1000 that the NSE raises 3% tomorrow- Alice!"*

Bob then sent back the message

"OK, but if not, then I will get your CD collection- Bob".

They decided to make up a real contract. Alice and Bob drew up a contract. Alice signed her copy and sent it to Bob. Then she waited for a very long time and still there is no response from Bob.

Here, Alice is sending mail to Bob, and she wants some evidence that Bob received the mail. If not there are two possibilities for the above contract signing scenario. They are case (1): suppose that the next day, the NSE rose 3% as mentioned in the contract, Alice will eagerly anticipating her money, she may e-mail Bob, including a copy of their contract. But much to her disappointment, Bob may respond like

"I didn't hear anything from you- Bob!"

(Since seeing the NSE index, Bob may conclude that he is going to lose the bet). case (2): alternatively, suppose the NSE go down to -2%. In that case, Bob may send an email to Alice stating that

"I am coming for CD collection- Bob."

When Bob stopped responding, Alice assumed that he had not received her copy of the contract. In fact, Bob was waiting to see whether he liked the way the contract would turn out. Only after he was sure of the outcome, he decides whether or not to honor the contract. He could do this because *Alice was bound to the contract before Bob*. Alice has no way to determine if Bob's failure to respond is malicious. Furthermore, she has no means of convincing others of his malicious actions. This is the contract signing problem.

The lack of *atomicity* is the primary cause of the contract signing problem. A series of operations is atomic means that either all of them happen, or none of them happen. A more thorough explanation of how atomicity is important for electronic commerce is given by Tygar (2003). When two parties are connected to each other via a possibly unreliable network, ensuring atomicity becomes a serious problem.

In a *contract signing problem*, there are two or more parties who are trying to agree on a contract. Each party will digitally sign the contract to signal their agreement. A variant of the contract signing problem is the *return receipt problem* or *certified mail problem*. Here, Alice is sending mail to Bob, and she wants some evidence that Bob received the mail. This is crucial for such applications as mail-order business. A return receipt protocol tries to ensure that Bob gets the mail only if Alice gets a receipt which will convince everyone that Bob actually got the mail. Both of these problems can be thought of as special cases of the *fair exchange problem*: Alice and Bob each have something the other wants. They want to exchange their items such that Alice gets Bob's item if and only if Bob gets Alice's item. In the case of contracts, the item is a signature on a shared contract.

4.3 SOLUTION FOR CONTRACT SIGNING PROBLEM

The problem is independent of the cryptographic tools used to sign the contract. It is independent of whatever kind of money used for fulfilling the contract. Cryptography alone will not solve the contract problem. Instead, the right approach is to use cryptographic tools to build *protocols* for solving problems. The requirements for a two-party contract signing protocol are *Fairness*: neither party should be bound to the contract without the other party being bound as well. If one party decides to abort the contract signing protocol, the other party should know that the protocol is over. *Completeness*: the protocol should be robust against adversaries attempting to cause it to abort without the consent of either party. *Non-Repudiation*: the protocol should not allow parties to arbitrarily decide to withdraw their support from a contract after the protocol is over. If Alice and Bob sign a contract, Alice should not be able to break the contract without Bob's consent and vice versa. *Efficiency*: signing the contract should require a minimum of messages and computation time.

4.4 VERIFIABLE ENCRYPTION

Suppose that Alice and Bob have agreed on a common message m = ["*I will bet you Rs.1000 that the NSE raises 3% tomorrow- Alice!"*|| *"OK, but if not, then I will get your CD collection- Bob".*]. Alice generates her signature $s = Sig_{sk_A}(m)$ on the message m and sends it to Bob "encrypted" under the public key of a TTP, T. That is the cipher text $\psi = E_{pub_T}(s)$. The problem, now, is that Alice must prove to Bob that the signature is valid, even though it is encrypted. At the same time Bob should send the receipt acknowledging the message sent by Alice. If there is any dispute arises between Alice and Bob then the Trusted Third Party T is able to extract s from the encryption that Alice has sent to Bob and adjudicate the problem while no useful information is revealed to others about s.

In general, given an instance s of a digital signature scheme on an arbitrary message m, the verifiable encryption of s is a process to prove or disprove that the cipher text of s is valid or not. Giuseppe Ateniese (2004) proposed the verifiable encryption of an RSA signature scheme. In this chapter Elliptic Curve Digital Signature Algorithm (ECDSA) is used to derive the scheme. The verifiable encryption scheme consists of five processes viz. key generation, encryption, decryption, initialization with the TTP and arbitration by TTP (if required), followed by a protocol. In this chapter, the Trusted Third Party will take part whenever there is a dispute between Alice and Bob.

4.5 MATHEMATICAL MODEL

The participating entities are Alice (prover), Bob (verifier) and the Trusted Third Party T. Suppose that Alice and Bob have agreed on a common message m. Let sk_A be the secret key of Alice and pub_T be the public key of

TTP. Alice generates her signature $s = Sig_{sk_A}(m)$ on the message m and sends it to Bob "encrypted" using the public key of a TTP, T. That is she sends the cipher text $\psi = E_{pub_T}(s)$ .The problem is that Alice must prove to Bob that the signature is valid, even though it is encrypted. After Bob is convinced he has to send a receipt message for acknowledging the message sent by Alice. If there is any dispute arises between Alice and Bob, then T is able to extract s from the encryption that Alice has sent to Bob and adjudicate the problem, while no useful information is revealed to others about s . In general, given an instance s of a digital signature scheme on an arbitrary message m the verifiable encryption $E_{pub_T}(s) = \psi$ of s can be verified using a protocol to prove or disprove that the cipher text for s is valid or not.

Giuseppe Ateniese (2004) proposed the verifiable encryption of an RSA signature scheme. In this chapter Elliptic Curve Digital Signature Algorithm (ECDSA) is used to derive the scheme. The verifiable encryption of an ECDSA signature scheme consist of the following processes and protocols.

1. Key generation
2. Signature generation
3. Signature verification
4. Initialization with the TTP
5. Protocol for obtaining the session key
6. Protocol to convince the verifier
7. Arbitration by the TTP (if required)

For the key generation, signature generation and signature verification Elliptic Curve Digital Signature Algorithm is used. In

Initialization phase Elliptic curve cryptosystem is used. For obtaining the session key Elliptic Curve Menezes-Qu-Vanstone (ECMQV) is used. Protocol to convince the verifier and arbitration by the TTP are the original contributions of this chapter.

4.6 ASSUMPTIONS FOR VERIFIABLE ECDSA SCHEME

It is assumed that the communication is carried over private and authenticated channels. The protocol provides fairness; specifically, it ensures that the sender receives the receipt if and only if the recipient will have the message in his mailbox within a finite period of time. In this protocol the TTP is invoked only in case of dispute. As long as both Alice and Bob follow the protocol steps, there is no need to involve the trusted entity in the protocol. Moreover, the protocol is designed to make sure that Alice cannot misbehave. Only Bob is allowed to cheat by not sending the receipt message. Since the sender initiates the exchange process, it appears natural to desire that the recipient of the message be relieved by any burden caused by malicious senders. Whenever a message is sent to Bob, Alice will go for Initialization phase. In this phase Alice send her certificate along with the base point P.

4.7 PROCESS 1: KEY GENERATION

The following are the domain parameters used for the ECDSA signature scheme:

1. A field size q, where either $q = p$, an odd prime, or $q = 2^m$
2. An indication FR (field representation) of the representation used for the element of F_q.
3. A bit string *seed E* of length at least 160 bits, if the elliptic curve was generated verifiably at random.

4. Two field elements a and b in F_q which define the equation of the elliptic curve E over F_q (i.e., $y^2 = x^3 + ax + b$ in the case $p > 3$, and $y^2 + xy = x^3 + ax^2 + b$ in the case $p = 2$).
5. Two field elements x_G and y_G in F_q which define a finite point $G = (x_G, y_G)$ of prime order in E.
6. The order ρ of the point G, with $\rho > 2^{160}$ and $\rho > 4\sqrt{q}$ and
7. The cofactor $cf = \#E(F_q)/\rho$.

Alice selects a random integer d and compute $Q = dG$. The public key is the elliptic curve point Q and the secret key is d.

4.8 PROCESS 2: SIGNATURE GENERATION

To sign a message m, Alice with domain parameters $D = (q, FR, a, b, G, \rho, cf)$ and associated key pair (d, Q) does the following:

1. Select a random or pseudorandom integer k, $1 \le k \le \rho - 1$.
2. Compute $kG = (x_1, y_1)$ and $r = x_1 \bmod n$. If $r = 0$ then go to step 1.
3. Compute $k^{-1} \bmod \rho$.
4. Compute $e = SHA-1(m)$. //Secure Hash Algorithm//
5. Compute $s = k^{-1}(e + dr) \bmod \rho$. If $s = 0$ go to step 1.
6. A's signature for the message m is (r, s).

4.9 PROCESS 3: SIGNATURE VERIFICATION

To verify Alice's signature (r,s) on m, Bob obtain an authentic copy of Alice's domain parameters $D=(q,FR,a,b,G,\rho,cf)$ and associated public-key Q. Bob then does the following:

1. Verify that r and s are integers in the interval $[1,\rho-1]$.
2. Compute $e=SHA-1(m)$.
3. Compute $w=s^{-1} \bmod \rho$.
4. Compute $u_1=ew \bmod \rho$ and $u_2=rw \bmod \rho$.
5. Compute $X=u_1G+u_2Q$. If $X=\infty$, then reject the signature. Otherwise, compute $v=x_1 \bmod \rho$ where $X=(x_1,y_1)$.
6. Accept the signature if and only if $v=r$.

4.10 PROCESS 4: ECDSA INITIALIZATION

Alice sends (G,Q) to T along with a certificate $Cert_A$. Let A be the string of data identifying Alice. TTP verifies (G,Q) is the public key of Alice. If the verification is done, then TTP randomly selects $\bar{g} \in Z_\rho^*$ and sets $g=\bar{g}^2 \bmod \rho$. Let x be the secret key and $y=g^x$ be the public key of the TTP T. T signs g using his secret key x and send back the certificate $Cert_{T:A}$ containing the parameters $g \bmod \rho$ and $g^x \bmod \rho$. This initialization phase is done only once. To resolve the dispute, if arises between Alice and Bob, the TTP and Alice should agree upon a common secret value called the session key. The process of obtaining the shared secret value is given in section 4.5.5. The initialization phase with the TTP is given in Figure 4.1.

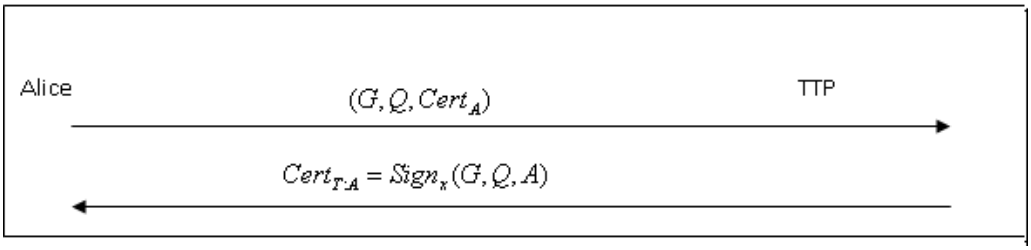

Figure 4.1 Initialization phase

4.11 PROTOCOL FOR OBTAINING SESSION KEY

MQV (Menezes-Qu-Vanstone) is an authenticated protocol for key agreement based on the Diffie-Hellman scheme. Like other authenticated Diffie-Hellman schemes, MQV provides protection against an active attacker. The protocol can be modified to work in an arbitrary finite group, and, in particular, elliptic curve groups, where it is known as elliptic curve MQV (ECMQV). MQV was initially proposed by Menezes et al (1995). It was modified by Laurie Law et al (1998).

The aim of this process is that the two entities Alice and TTP can establish a shared secret key between themselves. This is achieved by following four steps. Let A, T be the unique identity of Alice and TTP respectively. $D = (q, FR, S, a, b, P, \rho, cf)$ are elliptic curve domain parameters, KDF is a key derivation function, which is defined as $KDF(l) = (w_1, w_2)$, where the input l is the binary string of length b bits and the output w_1 is the first $\frac{b}{2}$ bits and w_2 is the next $\frac{b}{2}$ bits. H is the hash function. If R is an elliptic curve point then $\overline{R}$ is defined to be the integer representation of the x-coordinates of R.

Step 1: Alice selects a random number $k_A \in [1, n-1]$, computes $R_A = k_A P$, and sends A, R_A to TTP

Step 2: TTP does the following

2.1 Select $k_T \in [1, n-1]$ and compute $R_T = k_T P$

2.2 Computes $S_T = (k_T + \overline{R}_T d_T) \bmod n$ and $Z = cf.S_T(R_A + \overline{R}_A Q_A)$

2.3 $(u,v) \leftarrow \text{KDF}(\text{x}_Z)$, where x_Z is the x co-ordinate of Z.

2.4 Compute $t_T = H_{k_1}(2, T, A, R_T, R_A)$.

2.5 Send T, R_T, t_T to Alice

Step 3: Alice does the following:

3.1 Compute $\text{S}_\text{A} = (\text{k}_\text{A} + \overline{\text{R}}_\text{A}\text{d}_\text{A}) \bmod \text{n}$ and $Z = cf.S_A(R_T + \overline{R}_T Q_T)$

3.2 $(u,v) \leftarrow \text{KDF}(\text{x}_Z)$, where x_Z is the x co-ordinate of Z.

3.3 Compute $t = H_u(2, T, A, R_T, R_A)$ and verify that $t = t_T$

3.4 Compute $t_A = H_u(3, A, T, R_A, R_T)$ and send t_A to T.

Step 4: TTP computes $t = H_u(3, A, T, R_A, R_T)$ and verifies that $t = t_A$.

Now the session key shared between Alice and TTP is v.

4.12 PROTOCOL FOR CONVINCING THE VERIFIER

Given the message m, Alice selects a random integer k, $1 \le k \le n-1$ and computes the following:

$$kG = (x_1, y_1) \tag{4.1}$$

$$r = x_1 \bmod n \tag{4.2}$$

$$b = SHA - 1(m) \tag{4.3}$$

$$s = k^{-1}(b + dr) \bmod n \tag{4.4}$$

Alice's signature for the message m is (r, s). Then Alice encrypts the ECDSA signature via ElGamal (1985) Encryption scheme with the public key $y = g^x$. Let $sig = r \| s$ .Alice selects a random a and computes

$$k_1 = sig.y^a \tag{4.5}$$

$$k_2 = g^a \tag{4.6}$$

Alice sends the cipher text $(k_1 \| k_2)$ to Bob. Now Alice should prove to Bob that she has correctly generated her signature.

The participating entities are the Alice and Bob. The agenda is that Alice encrypts the ECDSA signature and sends to the verifier and subsequently convinces the verifier that the cipher text indeed decrypts to the correct signature. This protocol makes use of Elliptic curves cryptosystem for encrypting the message. Let E be the Elliptic Curve cyclic group of order ρ. Alice encrypts the ECDSA signature information as

$$sigB = A \tag{4.7}$$

and send the elliptic curve point A to the verifier. Now Alice wish to convince Bob that Equation (4.7) will obtain using the repeated addition of the value sig, without revealing the signature. This is achieved using the following protocol. The protocol is as follows:

Alice randomly chooses $a \in [-\rho, \rho]$ and compute

$$S = aB \tag{4.8}$$

where $S \in E$.

Bob chooses a random challenge c and sends to Alice.

Alice sends the response by computing

$$r = a - c.sig \tag{4.9}$$

Bob accepts the verifiable encryption of ECDSA signature if

$$S = rB + cA \tag{4.10}$$

Equation (4.10) holds because, using (4.7), (4.8), (4.9)

$$rB + cA = (a - c.sig)B + c.(sig.B) = aB - c.sig.B + c.sig.B = aB = S$$

4.13 PROCESS 5: ARBITRATION BY TTP

The dispute arises if Bob was not convinced with the Alice message or he may deny sending the proper receipt to Alice. In this case Alice may contact the TTP for adjudication. Alice will encrypt the ECDSA signature $sig = r \| s$, using the shared session key v and encrypt it via symmetric key encryption scheme and sent $E_v(sig = r \| s)$, $(k_1 \| k_2)$ to TTP.

The TTP will do the following: using the shared session key v, decrypt the symmetric key encryption scheme and retrieve the valid signature $sig = r \| s$ sent by Alice. After getting the cipher text $(k_1 \| k_2)$, TTP can decrypt using his secret key x by computing $(k_1)(k_2)^{-x}$ to get back the ECDSA signature $sig = r \| s$. This is because form equation's (4.6), (4.7) $(k_1)(k_2)^{-x} = (sig.y^a)(g^a)^{-x} = (sig.(g)^{x.a})(g)^{-x.a} = sig$. If the verification is done, then the TTP will send the receipt on behalf of Bob else reject it.

The overall verifiable encryption scheme is given in Figure 4.2. The protocol for obtaining the session key between Alice and TTP is shown in steps 1, 2 and 3. The protocol for convincing the verifier (Bob) is shown in steps 4, 5, 6 and 7. If Bob denies sending the receipt, arbitration by TTP is required. It is shown in steps 8 and 9.

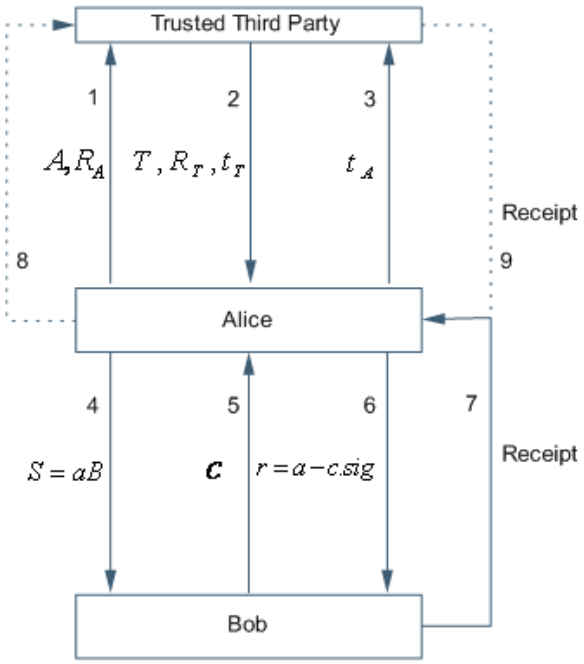

Figure 4.2 Overall verifiable encryption scheme

4.14 PERFORMACE EVALUATION

The performance of the scheme in terms of number of keys, computational complexity has been analyzed in this section. The following notation were used to analyze the performance of the scheme:

- **E** is the time for modular Exponentiation
- **M** is the time for modular multiplication
- **I** is the time for a modular inverse Computation
- **H** is the time for performing a one-way Hash

Here the time for performing modular addition/subtraction computation is ignored. The performance evaluation of the proposed scheme is given in Table 4.1 and the comparison of time needed is given in Table 4.2.

Table 4.1 Performance evaluation of the proposed scheme

Key Generation	1E+1H
Signature Generation	E+2M+H+I
Signature Verification	2E+2M+H+2M
Alice's Computation	1E+1M
Bob's Computation	2E

Table 4.2 Comparison of time needed

Description	**Key Size**	**ECDSA (nano seconds)**	**RSA (nano seconds)**
Key Generation	4 bit	356886	387654
	8 bit	376534	427876
	16 bit	567342	724543
	32 bit	812450	976345
Signature Generation	4 bit	398765	431256
	8 bit	478765	490988
	16 bit	598109	736543
	32 bit	923450	984320
Signature Verification	4 bit	412389	438569
	8 bit	529876	576209
	16 bit	635098	742987
	32 bit	987654	995760
Alice's Computation	4 bit	349876	398765
	8 bit	398735	419087
	16 bit	487654	523413
	32 bit	653876	765098
Bob's Computation	4 bit	437654	450982
	8 bit	498076	523190
	16 bit	589734	597652
	32 bit	865409	908456

4.15 INFERENCE AND CONCLUSION

Based on the numerical data, the time needed for the five major computation processes in this scheme viz. key generation, signature generation, signature verification, Alice's computation and Bob's Computation had been plotted in Figures 4.3, 4.4, 4.5, 4.6 and 4.7 respectively. This scheme is compared with RSA scheme. It is inferred that the performance of the ECDSA based scheme is very similar to RSA based scheme; therefore no additional time is required for processing the proposed scheme.

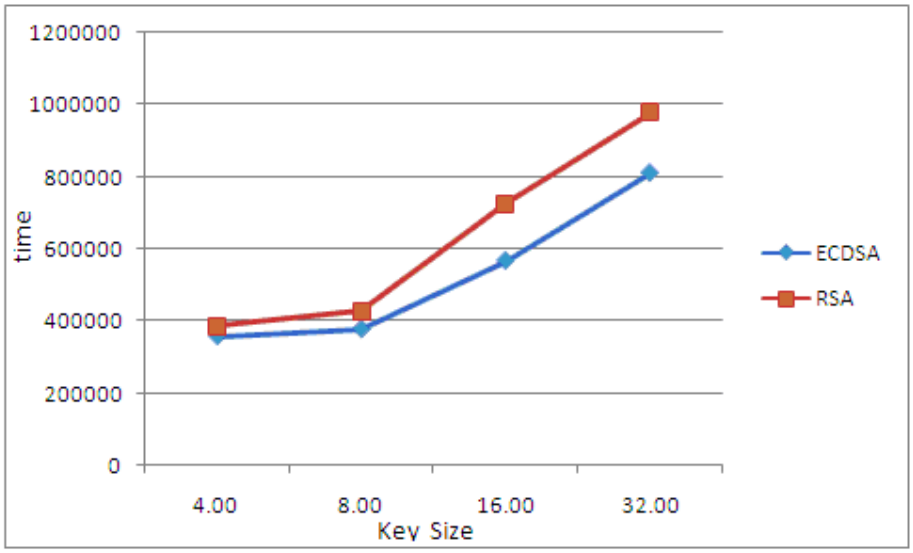

Figure 4.3 Key generation process

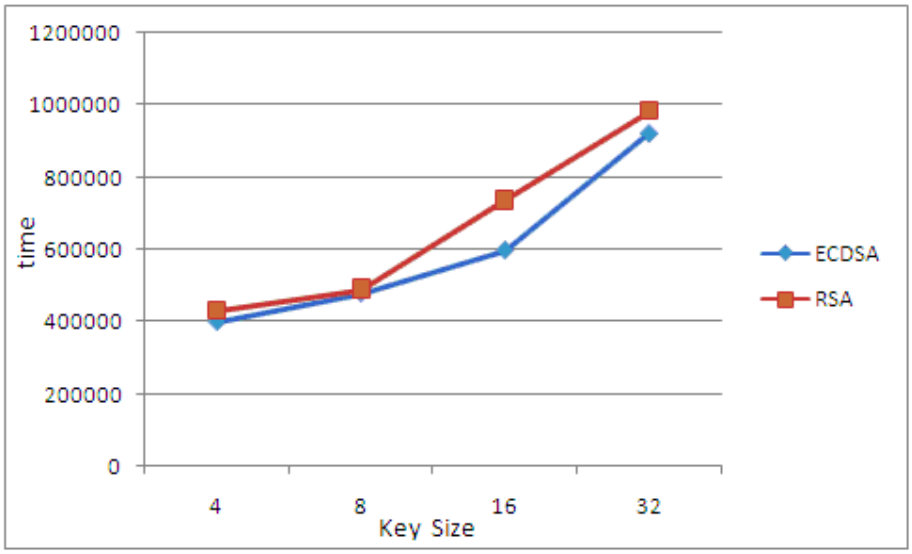

Figure 4.4 Signature generation process

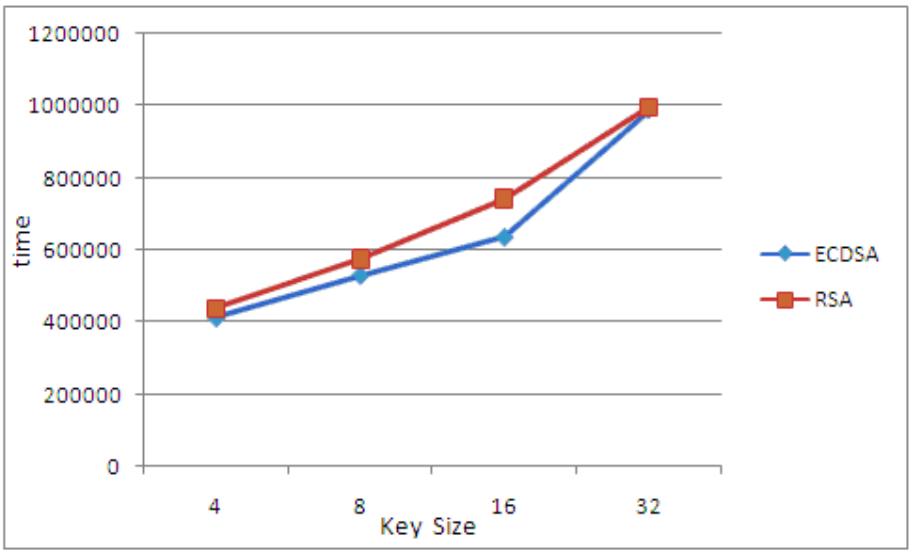

Figure 4.5 Signature verification process

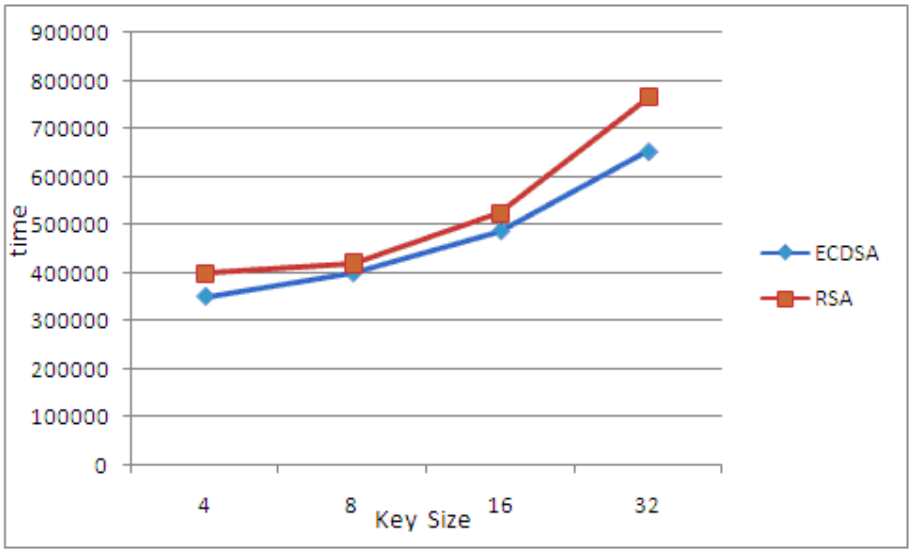

Figure 4.6 Alice's computation

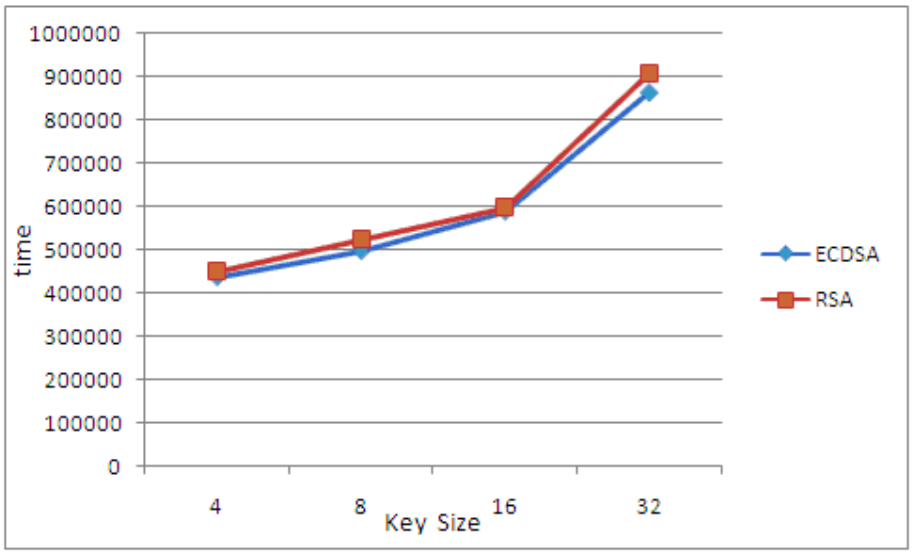

Figure 4.7 Bob's computation

Moreover, implementation of this scheme using Elliptic curve will yield the advantage that the key size is lesser than the RSA based schemes. Figure 4.8 shows the advantage of using Elliptic curve cryptography than RSA cryptosystem. To provide the designated security level, for RSA it requires 1024 bits and to provide the same security level, ECC requires only 192 bits.

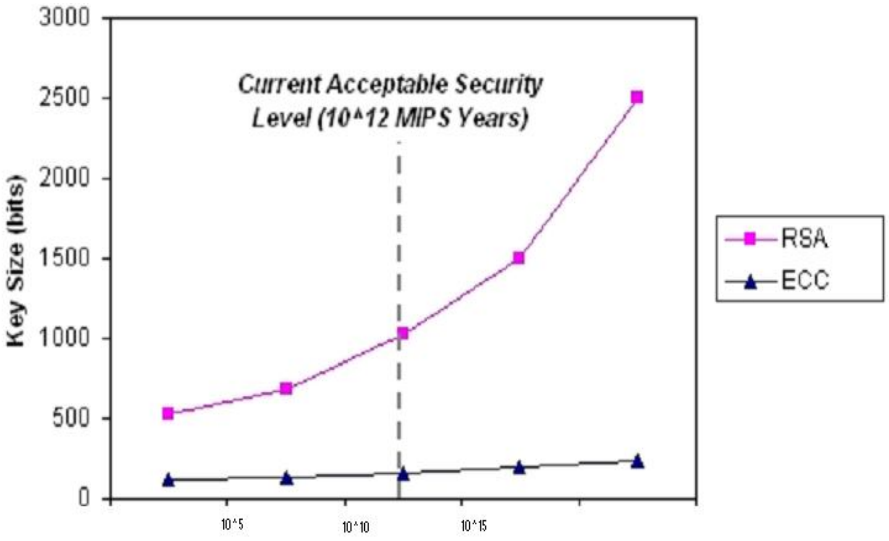

Figure 4.8 Comparison of security levels of ECC and RSA

Based on the discussions made, the following conclusions can be drawn:

- Using Verifiable Encryption of Elliptic Curve Digital Signature Algorithm (ECDSA), the key size used for the exchange of communication is comparatively less than RSA based signature scheme. The key size is reduced to approximately 5.3 time that of the key size used for RSA.
- Experimental data shows that there is no additional overhead of time is needed in implementing the proposed scheme.