CHAPTER 2

PROXY BLIND DISTRIBUTED DIGITAL SIGNATURE SCHEME

2.1 INTRODUCTION

This chapter addresses a new type of digital signature scheme named as Proxy Blind Distributed Signature Scheme which is based on Discrete Log Problem (DLP). The need for Proxy Blind Distributed Digital Scheme arises whenever a person delegate his signing authority to the intended person, there is no guarantee that the proxy signer will be in the position to do the work (signing process). The proxy signing system may be engaged with some other work or the system may be corrupted due to some malicious programs at that time. In this case, if the system is modeled with proxy signature scheme, then the original signer cannot achieve his goal. Hence the signing delegation is distributed into a group of persons with a threshold version. This technique will overcome the problem of unavailability of the system in a proxy signature scheme. Furthermore, this distributed concept will increase the robustness of the signature scheme.

Chaum (1982) introduced the concept of blind signature scheme. Using the blind scheme a user A can obtain the signature of any given message, without revealing any information about the message. His work was devoted to create an electronic version of money. He introduced the notation of “coins” and “blind signatures”. He claimed that it was the only way to ensure anonymity. There are two privacy properties for blind signatures; they

are untraceability and unlinkability like a coin. A coin cannot be easily traced from the bank to the shop which is called untracebility; furthermore, two spending of a same user cannot be linked together and it is termed as unlinkability.

Mambo et al (1996) introduced the concept of proxy signature scheme. In a proxy signature scheme, a potential signer delegates his signing capability to a proxy entity, who signs a message on behalf of the original signer. Yi and Xiao (2001) introduced the concept of proxy blind signature scheme. Further extension was done by Tan et al (2002) using DLP and Elliptic Curve Discrete Log Problem (ECDLP). This scheme is useful in several applications such as e-voting, e-payment and mobile agent environments. In a proxy blind signature scheme, the proxy signer is allowed to generate a blind signature on behalf of the original signer. Tan et al (2002) analyzed the security properties for a good proxy blind signature scheme. They are (i) Distinguishability: the proxy blind signature must be distinguishable from the normal signature. (ii) Non-repudiation: both the original signer and the proxy signer cannot deny their signatures against anyone. (iii) Verifiability: the proxy blind signature can be verified by everyone. (iv) Unforgeability: only the designated proxy signer can create the proxy blind signature. (v) Unlinkability: when the signature is revealed, the proxy signer cannot identify the association between the message and the blind signature generated.

Stinson and Stroble (2001) propose a distributed version of Schnorr's signature scheme, which is as secure as the non-distributed one. Pointcheval and Stern (2002) proved that this distributed signature scheme is unforgeable.

In a distributed signature scheme, a set of users shares the secret key of a standard signature scheme. If an authorized subset of members

collaborates, they can produce a valid signature on a message. The receipt can be verified for the correctness of this signature by any user, but cannot know whether it has been generated in a standard or a distributed way. These schemes are said to be unforgeable if an adversary cannot be able to compute a valid signature for any arbitrary message with un-authorized subset of members. The signing protocol is robust if the dishonest participants are detected and furthermore output of the protocol is always a valid signature. In other words, a signing protocol is said to be robust if it always produces a correct output, even in the presence of some tolerated subset of dishonest participants.

In the following section, the construction of proxy blind distributed digital signature is explained. The need for this scheme is that whenever a person delegates his signing authority to the intended person, there is no guarantee that the proxy signer will be in the position to sign and hence the signing delegation is distributed into a group of persons with a threshold version.

2.2 MATHEMATICAL MODEL

The participating entities in the digital signature scheme are the original signer A, a trusted dealer B, a group of proxy signer (servers) B_1, $B_2 \ldots B_n$ and the user C. The user C request the original signer to blindly sign the given message m which in turn direct the signing process to the group of proxy servers. The group of proxy servers will generate a shared secret key and blindly sign the message m on behalf of A. The user C will verify the validity of the signature using the verification process. The schematic diagram of this model is given in Figure 2.1.

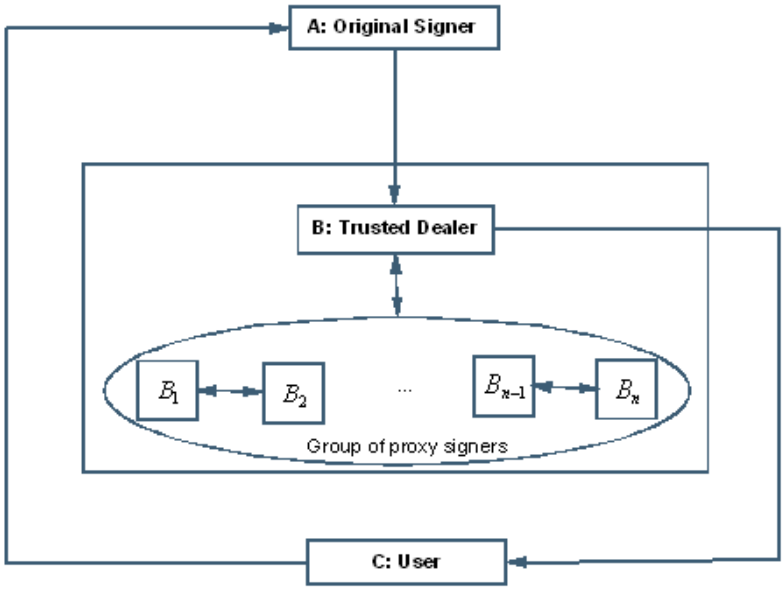

Figure 2.1 Proxy blind distributed signature scheme

The construction of a proxy blind distributed signature scheme contains the following processes:

1. Distribution of secret shares to the group of proxy servers by B
2. Verification for the valid shares by the individual proxy servers
3. Joint generation of keys by the group of servers
4. Generation of proxy keys (public key, secret key) by B
5. Signature generation by B
6. Signature verification by C

In process 1, for the distribution of secret shares and reconstruction of secret shares, Shamir's (1979) scheme is used. In the process 2,

Chor's (1985) model has been generalized for verification of valid shares. Processes 3, 4, 5 and 6 are the original contributions of this chapter.

2.3 PROCESS 1: DISTRIBUTION OF SECRET SHARES TO THE GROUP OF PROXY SERVERS BY B

Secret sharing forms the basis of threshold cryptography. Shamir (1979) constructed a very efficient such scheme for any n and t, where n is the number of authorized group of members and t is the threshold value less than n. Such schemes are called $t-out-of-n$ secret sharing schemes. Shamir's secret sharing scheme is a threshold scheme based on polynomial interpolation. It allows a dealer to distribute a secret value *s* to *n* servers, such that at least $t < n$ servers are required to reconstruct the secret. The protocol is information theoretically secure, i.e., fewer than *t* (threshold value) servers cannot gain any information about the secret by them. A $n-out-of-n$ secret sharing is a scheme where all n shares are needed to reconstruct, and if even one share is missing then there is absolutely no information about the secret. The trusted dealer B uses Shamir's secret sharing scheme to compute the secret share and distribute to the group of proxy servers B_1, B_2…B_n.

2.3.1 Shamir's Secret Sharing Scheme

Shamir's secret sharing scheme is as follows: Given a secret value $s = a_0$ and numbers n, t with $t < n$. Let $d = t - 1$ and $p < n$ be a prime number. Let $F_p = \{0,1,2,...p-1\}$ be the field with modulo *p* arithmetic. Let $B_1, B_2, ...B_n$ be the authorized group of *n* servers. To share $a_0 \in F_p$, a dealer $B \notin \{B_1,...,B_n\}$ choose independently at random $a_1,...,a_d \in F_p$. The number $a_0,...,a_d$ defines a polynomial $p(x)$ of degree d in the following way

$$p(x) = a_0 + a_1 x + ... + a_d x^d \tag{2.1}$$

Define $s_i = p(i), 1 \le i \le n$. The i^{th} share will be s_i. To recover a_0 among a group of t servers with in S, every server reveals its share and they publicly recover the secret

$$a_0 = p(0) = \sum_{i \in S} \lambda_{0,i}^{S} s_i \tag{2.2}$$

where

$$\lambda_{0,i}^{S} = \prod_{\substack{j \in S, \\ j \neq i}} \frac{i}{i - j} \tag{2.3}$$

is the Lagrange's coefficients. The following are the properties of the Shamir's secret sharing scheme. (i) Perfect Security: given any t shares, the polynomial is uniquely determined and hence the secret a_0 can be computed. However, given $t-1$ or fewer shares, information regarding the secret cannot be derived. (ii) Ideal: the size of each share is exactly same as the size of the secret. (iii) Extendable: additional shares may easily be created, simply by calculating the polynomial in additional points. (iv) Flexible: it is flexible to assign different weights (by the number of shares) to different authorities. (v) homomorphic property: Shamir's secret sharing scheme has the following homomorphism property

$$F(x_1 + y_1, x_2 + y_2, ...x_n + y_n) = F(x_1, x_2, ...x_n) + F(y_1, y_2, ...y_n) \tag{2.4}$$

2.4 PROCESS 2: VERIFICATION FOR THE VALID SHARES BY THE INDIVIDUAL PROXY SERVERS

In cryptography, a secret sharing scheme is verifiable if auxiliary information is included in the process so that it allows the users to verify the consistency of their shares. In the previous scheme it is assumed that the

dealer is reliable, however, a misbehaving dealer can distribute inconsistent shares to the participants, from which they will not be able to reconstruct a secret. To prevent such malicious behavior of the dealer, one needs to implement a protocol through which a consistent dealing can be verified by the recipients of shares. The problem of verifiable secret sharing is to convince shareholders that their shares (collectively) are, t-Consistent, meaning that every subset of t shares out of n (that the dealer distributed) defines the same secret.

The concept of verifiable secret sharing (VSS) was first introduced by Chor et al (1985). The VSS protocol consists of two phases: a sharing phase and a reconstruction phase. In a sharing phase, initially the dealer holds secret as input and each user holds an independent random input. The sharing phase may consist of several rounds. At each round each server can privately send messages to other servers and they can also broadcast a message. Each message sent or broadcasted by a server is determined by its input, its random input and messages received from other servers in previous rounds. In the reconstruction phase each server provides its entire view from the sharing phase and a reconstruction function is applied and is taken as the protocol's output.

2.4.1 Feldman's Verifiable Secret Sharing Scheme

A commonly used example of a simple VSS scheme is the protocol by Paul Feldman (1987), which is based on Shamir's secret sharing scheme combined with any Homomorphic encryption scheme. In order to make these shares verifiable, the dealer distributes the coefficients of $p(x) = a_0 + a_1 x + ... + a_d x^d$ as commitments. Let g be the base elements in Z_p then the commitments that must be given are

$$c_0 = g^{a_0}, c_1 = g^{a_1}, ..., c_d = g^{a_d} \tag{2.5}$$

Once these commitments are given, any server can verify their share.

To verify $s_i = p(i) \bmod p$ is a valid share or not the party i can check that

$$g^{s_i} = g^{p(i)} = g^{a_0 + a_1 i + \ldots + a_d i^d} = c_0 c_1^i \ldots c_d^{i^t} = \prod_{j=0}^{d} c_j^{i^j} \tag{2.6}$$

2.4.2 Generalization of Verifiable Secret Sharing Scheme

Let the group of n servers be indexed with numbers $1,2,\ldots n$. Let $I = \{1,2,\ldots n\}$ be the set denoting the n servers. Let $\psi : I \to (Z_p)^t$ be a function which maps an element in 1-dimensional space to an element in t-dimensional space. Define $\psi(i) = (1, i, i^2, \ldots, i^{t-1})$ .If U is any subset of I and D is the interactive algorithm then $\psi(D) = \sum_{i \in U} a_i \psi(i)$ for some $a_i \in Z_p$ .Each server i executes (VSS) verifiable secret sharing scheme playing the role of a dealer. That is, they chooses two random vectors $v_i = (v_i^{(1)}, \ldots, v_i^{(t)})$ and $w_i = (w_i^{(1)}, \ldots, w_i^{(t)})$, in $(Z_p)^t$ where $v_i . \psi(D) = k_i \bmod p$ is the random secret distributed by servers. Server i computes the pair

$$(s_{ij}, s_{ij}^{'}) = (v_i . \psi(j), w_i . \psi(j)), \text{ for } 1 \le j \le n \tag{2.7}$$

and sends to server j and then broadcast the public commitments

$$C_{im} = g^{v_i^{(m)}} h^{w_i^{(m)}}, \text{ for } 1 \le m \le t \tag{2.8}$$

where h and g are the base elements in Z_p.

Each server j checks that

$$g^{s_{ij}} h^{s'_{ij}} = \prod_{m=1}^{t} \left(C_{im}\right)^{\psi(j)^{(m)}} \tag{2.9}$$

holds. If it is true, then the server i is accepted else server i is rejected. The proof of the result is given in Theorem 2.3.

2.5 PROCESS 3: JOINT GENERATION OF KEYS BY THE GROUP OF SERVERS

The joint generation of secret key by the group of servers is also known as distributed key-generation protocol (DKG). This protocol is used for generating a public key and a sharing of the corresponding secret key. This protocol ensures that the corrupted parties learn no information about the secret key. These protocols have been implemented for the common public key types, discrete logarithm and RSA. Usually they assume synchronous network and passive adversaries.

Gennaro et al (2007) design a protocol in which a set of users jointly generate a public key $y = g^x$ and shares of the corresponding secret key x, in such a way that t or more servers can recover this secret key. The idea is that each user i plays the role of a dealer and shares a random value k_i among the servers. The secret key x will be the sum of some of these values. The protocol is as follows:

At the verification of valid share phase, those servers who cheat are detected and rejected. Let $P = \{B_1, B_2, ...B_n\}$. Define $F_0 = \{i \mid \text{server } i \text{ is not rejected at the verification phase}\}$. If the number of elements in the set is less than the threshold value then the key distribution process will be difficult, so the system will be restart with the authorized set of participants. All members

$i \in F_0$ that pass this phase have a valid shares s_{ij} corresponding to members j that form an authorized subset A. Each member $j \in P$ computes his share of the total secret as

$$x_j = \sum_{i \in F_0} s_{ij} \tag{2.10}$$

The total secret will be

$$x = \sum_{i \in F_0} x_i \in Z_p \tag{2.11}$$

The public value

$$y = g^x = \prod_{i \in F_0} g^{x_i} \in Z_p^* \tag{2.12}$$

After execution of this protocol, the public key y and its corresponding secret key x are generated where $y = g^x$, $x = \sum_{i \in F_0} x_i \in Z_p$ and $x_j = \sum_{i \in F_0} s_{ij}$ is the share of server j corresponding to the secret x.

2.6 PROCESS 4: GENERATION OF PROXY KEYS (PUBLIC KEY, SECRET KEY) BY B

The group of members constructs the proxy key pair (x_p, y_p) as follows. Let g is the base elements in Z_p. The original signer A generates a random number $k_A \in Z_p^*$ and computes

$$r_A = g^{k_A} \bmod p \tag{2.13}$$

and then calculates

$$s_A = x_A + k_A y_B \bmod p \tag{2.14}$$

where x_A is the secrete key of the original signer A and $y_A = g^{x_A}$ is the associated public key. Similarly x_B is the jointly generated secrete key of the group of proxy signers and $y_B = g^{x_B}$ is the associated public key. Using Shamir's secret sharing scheme (s_A, r_A) is distributed as a share to the group of members in B. By doing so, if any of the proxy servers (proxy signer) is busy, then the alternate servers may fulfill the job. Any subset of t members can reconstruct the values (s_A, r_A).A common server in B (TTP) checks whether $g^{s_A} = y_A . r_A^{y_B}$ holds or not. If it is true, then the TTP computes the proxy private key as

$$x_p = s_A + x_B y_A \bmod p \tag{2.15}$$

where x_B is the joint generation of shared secret key by the group of members in B using (VSS) verifiable secret sharing scheme. The proxy public key is computed by the TTP using

$$y_p = g^{x_p} \bmod p \tag{2.16}$$

2.7 PROCESS 5: SIGNATURE GENERATION BY B

The Client C gets the proxy blind distributed signature as follows. The TTP in B computes

$$r = g^k \bmod p \tag{2.17}$$

where $k \in Z_p^*$ is jointly generated by the group members in B, and sends r to the client C.The Client C computes

$$r^* = rg^{a+x_c} y_p^{-b} \bmod p \tag{2.18}$$

where $a, b \in Z_p^*$,

$$e^* = H(r^* \| m) \bmod p \tag{2.19}$$

where $H(\ )$ is a secured hash function, and then compute

$$e = e^* + b \bmod p \tag{2.20}$$

and sends e to B (TTP). TTP in B computes

$$s = k - ex_p \bmod p \tag{2.21}$$

and returns s to C. Upon receiving s, C computes

$$s^* = s + a \bmod p \tag{2.22}$$

The proxy blind distributed signature of message m is (r^*, s^*, e^*).

2.8 PROCESS 6: SIGNATURE VERIFICATION BY C

Verifier can verify the proxy blind distributed signature by checking whether

$$e^* = H(g^{s^*} y_p^{e^*} y_c \bmod p \| m) \tag{2.23}$$

holds. This is because $g^{s^*} y_p^{e^*} y_c \bmod p = g^{s+a} y_p^{e^*} g^{x_c} \bmod p = g^{s+a+x_c} y_p^{e^*} \bmod p$ $= g^{s+a+x_c} y_p^{e-b} \bmod p \quad = g^{k-ex_p+a+x_c} \left(g^{x_p}\right)^{e-b} \bmod p \quad = g^{k-ex_p+a+x_c} g^{ex_p-bx_p} \bmod p$ $= rg^{a+x_c} y_p^{-b} \bmod p = r^*$ and from (2.18) it is clear that (2.23) holds.

2.9 SECURITY ANALYSIS

Now, based on the above mathematical equations the following security analysis is carried out. 'Provable security '- approach is used for evaluating the security of the digital signature scheme. The security of this digital signature scheme is based on the intractability of the Discrete Log Problem.

Theorem 2.1

The attacker cannot construct a valid proxy private key x_p without the knowledge of the jointly generated shared secret key x_B by the group of members in B.

Proof

Since $r_A = g^{k_A} \bmod p$ and $s_A = x_A + k_A y_B \bmod p$, we know that $y_A = g^{x_A}$ and $y_B = g^{x_B}$ are the public keys of the user A and B, x_A and x_B are the corresponding secret keys. $k_A \in Z_p^*$ is the unique random number generated by the original signer A. Therefore the valid proxy key is constructed using the formulae

$$x_p = s_A + x_B y_A \bmod p = (x_A + k_A y_B + x_B y_A) \bmod p = (x_A + k_A g^{x_B} + x_B g^{x_A}) \bmod p.$$

Using the assumption that DLP is very hard to solve, an attacker cannot construct x_p without the knowledge of x_B which is generated by the group of servers in a secured way.

Theorem 2.2

The attacker cannot construct a valid proxy blind distributed signature for a selected message m^1 without the knowledge of x_c.

Proof

We know that the proxy blind distributed signature for the message m is (r^*, s^*, e^*), where $r^* = rg^{a+x_c} y_p^{-b} \bmod p$, $a, b \in Z_p^*$, $s^* = s + a \bmod q$, $s = k - ex_p \bmod p$ $k \in Z_p^*$ is jointly generated by the group members in B. $e^* = H(r^* \| m) \bmod p$ and $e = e^* + b \bmod p$.

Therefore $(r^*, s^*, e^*) = (rg^{a+x_c} y_p^{-b} \bmod p, s + a \bmod p, H(r^* \| m) \bmod p)$

Using the assumption that DLP is very hard to solve, an attacker cannot construct r^* without the knowledge of x_c which is the secret key of the client. If the attacker obtain $a^1 = s^* - s$ and $b^1 = e - e^*$, he cannot find a corresponding r by checking $r^* = rg^{a+x_c} y_p^{-b} \bmod p$ without x_c.Hence the attacker cannot construct a valid proxy blind distributed signature for a selected message m^1 without the knowledge of x_c.

Theorem 2.3

In the VSS if $(s_{ij}, s_{ij}^{'}) = (v_i.\psi(j), w_i.\psi(j))$ and $C_{im} = g^{v_i^{(m)}} h^{w_i^{(m)}}$ for $1 \le m \le t$ where h and g are the base elements in Z_p then $g^{s_{ij}} h^{s_{ij}^1} = \prod_{m=1}^{t} (C_{im})^{\psi(j)^{(m)}}$.

Proof

$$\prod_{m=1}^{t}\left(C_{im}\right)^{\psi(j)^{(m)}} = \left(C_{i1}\right)^{\psi(j)^{(1)}}.\left(C_{i2}\right)^{\psi(j)^{(2)}}....\left(C_{it}\right)^{\psi(j)^{(t)}}$$

$$= \left(g^{v_i^{(1)}} h^{w_i^{(1)}}\right)^{\psi(j)^{(1)}} ...\left(g^{v_i^{(t)}} h^{w_i^{(t)}}\right)^{\psi(j)^{(t)}}$$

$$= g^{v_i^{(1)}.\psi(j)^{(1)}+...v_i^{(t)}.\psi(j)^{(t)}}.h^{w_i^{(1)}.\psi(j)^{(1)}+...w_i^{(t)}.\psi(j)^{(t)}}$$

$$= g^{v_i.\psi(j)}.h^{w_i.\psi(j)} = g^{s_{ij}}.h^{s_{ij}^1}$$

2.10 NUMERICAL RESULTS

In this section, using Java programming the digital signature scheme has been coded and the following results were obtained. Let P= {1, 2, 3, 4, 5, 6, 7} be the set of servers with n = 7. Let A= {1, 3, 4, 5, 7} be the authorized subset of P with threshold value t=5. Let $\psi(i) = (1, i, i^2, i^3, i^4)$ and $\psi(D) = (1,0,0,0,0)$. This shares distributed to each servers is given in Table 2.1 and verification of secret shares by each server is given in Table 2.2.

Table 2.1 Shares distributed to each servers

$v_i = (v_i^{(1)},..., v_i^{(5)})$	$w_i = (w_i^{(1)},..., w_i^{(5)})$	$(s_{ij}, s_{ij}^{'}) = (v_i.\psi(j), w_i.\psi(j))$
(7,5,4,3,2)	(1,0,1,0,1)	{(21,3),(67,10),(81,14),(80,9),(74,13),(88,13),(74,9)}
(5,2,0,1,2)	(3,2,6,1,0)	{(10,12),(27,39),(24,68),(28,50),(37,35),(42,40),(27,49)}
(3,4,7,2,1)	(1,2,3,0,5)	{(17,11),(60,42),(92,54),(75,39),(61,65),(71,67),(73,45)}
(6,7,2,3,1)	(0,1,2,3,0)	{(19,6),(57,34),(64,36),(74,41),(68,23),(84,33),(74,23)}
(2,5,3,6,1)	(7,6,2,1,0)	{(17,16),(77,35),(78,48),(94,50),(69,47),(92,56),(67,61)}
(1,2,3,4,5)	(2,2,3,3,1)	{(15,11),(74,47),(74,54),(75,55),(81,42),(95,53),(53,40)}
(4,5,7,2,3)	(1,2,3,0,0)	{(21,6),(73,17),(104,34),(86,24),(85,20),(96,22),(87,30)}

Table 2.2 Verification of secret shares

Server	Commitment	Verification (1,2,3,4,5,6,7))	Individual Shares of total secret
1	{9,10,8,8,2}	(**3**,3,9,8,2,5,10) = (**3**,3,9,8,2,5,10)	10
2	{4,1,5,1,4}	(7,**3**,3,8,4,7,9) = (7,**3**,3,8,4,7,9)	6
3	{4,4,5,4,9}	(7,9,**3**,3,1,1,1) = (7,9,**3**,3,1,1,1)	0
4	{9,9,1,1,2}	(2,3,9,**8**,5,1,4) = (2,3,9,**8**,5,1,4)	6
5	{10,6,2,10,2}	(2,4,9,10,**4**,6,10) = (2,4,9,10,**4**,6,10)	2
6	{6,1,1,2,5}	(10,4,5,2,9,**4**,5) = (10,4,5,2,9,**4**,5)	7
7	{8,8,5,4,8}	(10,3,3,2,9,8,**7**) = (10,3,3,2,9,8,**7**)	4

Let the total secret = 2, $x_B = 2$, $y_B = 4$, secret key of A = $x_A = 3$, Random number generated by A is $k_A = 4, r_A = 4$, $s_A = 3$, secret key of C is $x_c = 3$, $a = 4, b = 5$ chosen by C. Signature of the message m =3 is (r^*, e^*, s^*) = (3, 3, 1) where H(3||m)= 3. Verification: LHS = $e^* = 3$ and the RHS = $H(3 \| m) = 3$, hence LHS = RHS

2.11 PERFORMANCE 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

- **SK** and **PK** are the number of secret and public keys respectively
- **B** is the trusted dealer
- **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
- **t** is the threshold value.

Here the time for performing modular addition/subtraction computation is ignored.

Table 2.3 gives the number of keys and computational complexity of the scheme and Table 2.4 gives a comparison of time in nanosecond for the presented scheme and Kim Lee's digital signature scheme for a set of values of the key size.

Table 2.3 Performance evaluation of the proposed scheme

		Our Signature Scheme
The number of Keys	SK(B)	1
	PK(B)	1
Computational Complexity	Key Generation	5E+4M+t[3E+2M]+I
	Signing	3E+3M+H
	Verify	2E+2M+H

Table 2.4 Comparison of time needed

Description	Key Size	Proposed Scheme Time (nano seconds)	Kim Lee's scheme Time (nano seconds)
Key Generation	4 bit	18216798	17016563
	8 bit	22564902	21242312
	16 bit	25767774	25091622
	32 bit	29846109	26913312
Signature Generation	4 bit	253893	285821
	8 bit	431318	491391
	16 bit	1264869	1426464
	32 bit	2569710	3045421
Signature Verification	4 bit	56749	61342
	8 bit	86155	89521
	16 bit	106863	124321
	32 bit	130761	162110
Overall Process	4 bit	18527440	17363726
	8 bit	23082375	21823224
	16 bit	27139506	26642407
	32 bit	32546580	30120843

2.12 INFERENCE AND CONCLUSION

Based on the numerical data, the performance for the overall processes in digital signature schemes viz. key generation, signature generation, signature verification and overall processes had been plotted in Figure 2.2. The Proxy blind distributed digital signature scheme is compared with the Kim Lee's digital signature scheme.

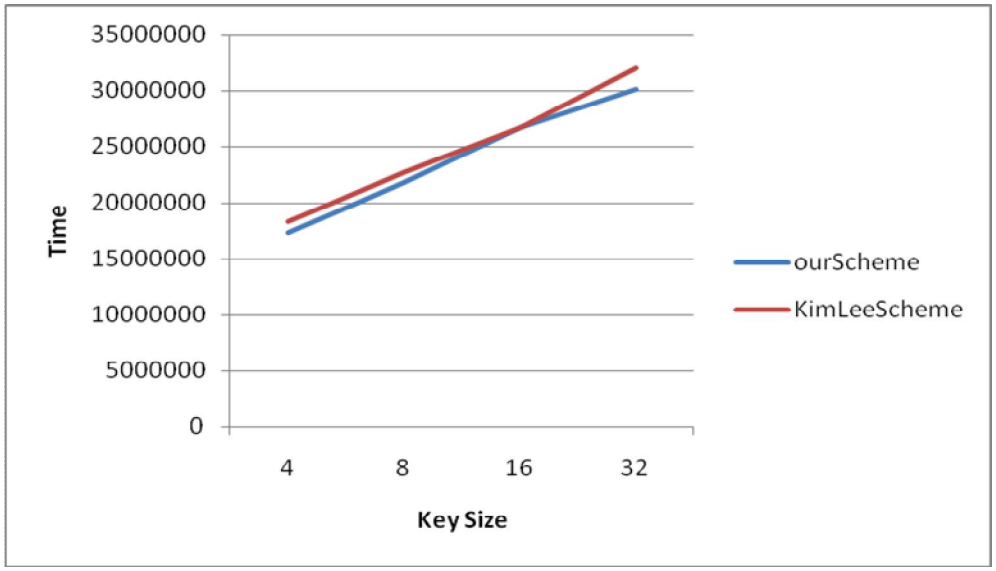

Figure 2.2 Comparison of the time needed for the proposed scheme with Kim-Lee's scheme

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

- The computation time for the key generation process is slightly higher than the Kim Lee's scheme, while for the signature generation and signature verification it is lesser.
- Though the system is distributed, the computation time for implementing the proxy blind distributed digital signature scheme is very similar to Kim Lee's signature scheme; therefore no additional time is required for processing the proposed digital signature scheme.
- This scheme prevents the original signer's forgery by Theorem 2.1.
- It prevents the recipient's universal forgery by Theorem 2.2.

- In the distributed key generation process, if an attacker or an intruder sends any false information to collapse the key generation process, this scheme will detect that person and eliminate him in the next run by newly defining the set of participants.

- Also this scheme take care that if the number of elements in the set F_0 is less than the threshold value t then the key distribution process will be difficult, so the system will be restart with the authorized set of participants.

- As this scheme delegates the signing authority to a group of systems with a threshold version. The system will output the correct result even if one or few system is busy or corrupted and hence the scheme is robust.