Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 790257, 24 pages
http://dx.doi.org/10.1155/2013/790257

Research Article

Cryptanalysis and Performance Evaluation of Enhanced Threshold Proxy Signature Scheme Based on RSA for Known Signers

Raman Kumar, Harsh Kumar Verma, and Renu Dhir

Department of Computer Science and Engineering, Dr. B. R. Ambedkar National Institute of Technology, Jalandhar, Punjab, India

Correspondence should be addressed to Raman Kumar; er.ramankumar@aol.in

Received 24 July 2012; Accepted 16 November 2012

Academic Editor: Hung Nguyen-Xuan

In these days there are plenty of signature schemes such as the (t, n) threshold proxy signature scheme (Kumar and Verma 2010). The network is a shared medium so that the weakness security attacks such as eavesdropping, replay attack, and modification attack. Thus, we have to establish a common key for encrypting/decrypting our communications over an insecure network. In this scheme, a (t, n) threshold proxy signature scheme based on RSA, any t or more proxy signers can cooperatively generate a proxy signature while $t - 1$ or fewer of them cannot do it. The threshold proxy signature scheme uses the RSA cryptosystem to generate the private and the public key of the signers (Rivest et al., 1978). Comparison is done on the basis of time complexity, space complexity, and communication overhead. We compare the performance of four schemes (Hwang et al. (2003), Kuo and Chen (2005), Yong-Jun et al. (2007), and Li et al. (2007), with the performance of a scheme that has been proposed earlier by the authors of this paper. In the proposed scheme, both the combiner and the secret share holder can verify the correctness of the information that they are receiving from each other. Therefore, the enhanced threshold proxy signature scheme is secure and efficient against notorious conspiracy attacks.

1. Introduction

Today Internet is an inseparable part of our life and millions of people will be using the Internet. Reading the news, chatting with friends, purchasing a new product, and researching for a paper, the number of uses of the Internet is endless. One of the attractions of the Internet is that one can do almost anything from the comfort of his/her own home and with a relative sense of anonymity.

Unfortunately, the data going across the Internet may not be as secure as we would like to think. It is not especially difficult for a person with the right technical skills to intercept the data going from one computer to another. Usually this is not a problem; people do not really care if someone knows that they went to http://www.google.com/ and started researching Number Theory. However, if the intercepted data contains a credit card number, password, social security number, or some other private information, it becomes a whole different story.

Online banking and a host of other services rely heavily upon the security of credit card numbers, PINs, and other private information as it goes across the network. But if it is easy to intercept these numbers, how do these services work? The answer is cryptography.

In today's commercial environment, establishing a framework for the authentication of computer-based information requires a familiarity with concepts and professional skills from both the legal and computer security fields. Combining these two disciplines is not an easy task because concepts from the information security field often correspond only loosely to concepts from the legal fields, even in situations where the terminology is similar. For example, from the information security point of view, "digital signature" means the result of applying to specific technical processes. The historical legal concept of "signature" is broader. It recognizes any mark made with the intention of authenticating the marked document [1].

In this research paper, we discuss threshold proxy signature scheme. In a (t, n) threshold proxy signature schemes, an original signer delegates a group of n proxy signers to sign message on behalf of him or her. When the proxy signature is created, t or more proxy signers cooperate to generate valid proxy signatures and less than t proxy signers cannot cooperatively produce valid proxy signatures. In essence, we have tested our enhanced threshold proxy signature scheme by undergoing some fruitful attacks.

2. Review of Threshold Proxy Signature Schemes

2.1. History of Threshold Proxy Signature Schemes. In the history of proxy signature technological development, the $(1, n)$ threshold proxy signature technique was the first to come. In $(1, n)$ proxy signature schemes, a legal proxy signature can be generated by a designated proxy signer by using a proxy signing key. The proxy signing key is computed from the original signer's private key, but the private key should not be computed from the proxy signing key in any way. In the eye of a modern user, such schemes are simple but not flexible. In order to extend proxy signature schemes to fit various practical situations, many (t, n) threshold proxy signature schemes have been proposed. For example, we have (t, n) threshold proxy signature schemes that allow any t or more proxy signers from a designated group of n members to cooperatively sign messages, while $t - 1$ or fewer members cannot generate the legal proxy signature. In practice, the original signer can flexibly choose the threshold t. The approach agrees with $(1, n)$, (t, n), and (n, n) threshold delegations.

Rivest et al. [1] and Blakley firstly proposed the (t, n) threshold secret sharing scheme based upon Lagrange interpolating polynomial and linear projective geometry, respectively, in 1979. In a (t, n) threshold secret sharing scheme, secret holder delivers the distinct secret values (called shares or shadows) to n participants. At least t or more participants can combine their shares and reconstruct the secret, but only $t - 1$ or fewer members cannot. Based on these properties, secret sharing is an important part of modern cryptography and has been used in many fields of modern cryptography. In 1996, Mambo et al. [12] proposed the concept of proxy signature. In their schemes, the original signer can delegate his/her right to the proxy signers who can sign the message instead of the original signer.

Recently, many threshold proxy signature schemes were proposed. The history of threshold proxy signature schemes is made up in Table 1.

The concept of threshold cryptosystems was also brought up by Denmedt and Frankel in 1991. They adapted the ElGamal public key cryptosystem and used the Lagrange interpolation or geometry to produce shadows.

To make proxy signature applicable to group-oriented situations, Sun [13] and Kim et al. [5] proposed a (t, n) threshold proxy signature in 1997, which is a variant of proxy signature by using the ideas of secret sharing and threshold cryptosystems. The basic strategy used in Kim et al.'s scheme is a random number generation.

2.2. Review of Kim et al.'s Scheme

2.2.1. The Random Number Generation Phase. This scheme requires a protocol to generate a random number among the group without the dealer. Let P_0 be the original signer, and let $P_1, P_2, \ldots, P_n$ be the n proxy signers of the proxy group.

(1) Each proxy signer P_i selects secret polynomial of degree $t - 1$ such that

$$f_i(x) = r_i + a_{i,1}x + a_{i,2}x^2 + \cdots + a_{i,t-1}x^{t-1} \bmod q, \quad (1)$$

where r_i, $a_{i,1}$, $a_{i,2}$, and $a_{i,t-1}$ are random numbers.

(2) Then, each P_i computes $f_i(j) \bmod q$ and sends it to P_j for all $1 \le j < n$ and $j \ne n$. Furthermore, P_i computes

$$g^{ri}, g^{ai,1}, g^{ai,2}, \ldots, g^{ai,t-1} \pmod p \quad (2)$$

and broadcasts them.

(3) After receiving $f_j(i)$ (for $j = 1, 2, \ldots, n$ and $j \ne i$), P_i confirms that the validity of $f_j(i)$ by checking whether or not $g^{fj(i)}$ satisfies the following equation:

$$g^{fj(i)} = g^{rj} \times \left(\left(g^{aj,1}\right)^i\right)^1 \times \cdots \times \left(\left(g^{aj,t-1}\right)^i\right)^{t-1} \bmod p. \quad (3)$$

(4) If the verifications in Step 3 hold, each P_i computes the secret share

$$S_i = \sum_{j=1}^{j=n} f_j(i) \quad (4)$$

and computes public outputs

$$\begin{aligned} r &= \prod_{j=1}^{j=n} r_j \bmod p, \\ g^{a1} &= \prod_{j=1}^{j=n} g^{j,1} \bmod p, \\ g^{a2} &= \prod_{j=1}^{j=n} g^{j,2} \bmod p, \\ &\vdots \\ g^{at-1} &= \prod_{j=1}^{j=n} g^{j,t-1} \bmod p. \end{aligned} \quad (5)$$

2.2.2. The Proxy Sharing Phase

(1) *Group Key Generation.* First, the proxy group must execute the above protocol to obtain the share s_i and the public outputs $y_G = g^{a0} \bmod p$, $A_j = g^{aj} \bmod p$, where

$$j = 1, 2, \ldots, t - 1. \quad (6)$$

Table 1: History of threshold proxy signature schemes.

Serial number	Scheme	Method
1	Rivest et al. [1]	The Lagrange interpolating polynomial and linear projective geometry
2	ElGamal [2]	Discrete logarithms
3	Desmedt and Frankel [3]	RSA and Lagrange coefficient
4	Zhang [4]	Discrete logarithms
5	Kim et al. [5]	Discrete logarithms
6	Sun et al. [6]	Discrete logarithms
7	Lee (2001)	Discrete logarithms
8	Hwang et al. [7]	RSA and Lagrange coefficient
9	Wang et al. [8]	RSA and Lagrange coefficient
10	Kuo and Chen [9]	RSA and Lagrange coefficient
11	H. Jiang (2007)	RSA and Lagrange coefficient
12	Fanyu (2007)	RSA and Lagrange coefficient
13	Li et al. [10]	RSA and Lagrange coefficient
14	Yong-Jun et al. [11]	RSA and Lagrange coefficient

(2) *Proxy Generation.* The original signer computes $K = g^k \bmod p$ and $e = h(m_w, K)$, where k is a random number, m_w is a warrant, and $h(\,)$ is one way hash function. After this, P_0 computes

$$\sigma = e \times x_0 + k \bmod q, \tag{7}$$

where x_0 is a private key of the original signer.

(3) *Proxy Sharing.* P_0 randomly chooses a polynomial such that

$$f'(x) = \sigma + b_1 x + b_2 x^2 + \cdots + b_{t-1} x^{t-1}, \tag{8}$$

where b_1, b_2, b_{t-1} are random numbers. Then, P_0 computes $f'(i)$ and sends it to each P_i in a secret manner; P_0 also computes

$$B_1 = g^{b1}, \qquad B_2 = g^{b2}, \ldots, B_{t-1} = g^{t-1} \pmod p \tag{9}$$

and publishes $m_w, K, B_1, B_2, \ldots, B_{t-1} \pmod p$.

(4) *Proxy Share Generation.* After receiving $f'(i)$, each P_i has to validate $f'(i)$ using

$$g^{f'(i)} = (y_0)^{h(mw,K)} K \prod_{j=1}^{t-1} \left(\left(B_j \right)^i \right)^j \bmod p, \tag{10}$$

where y_0 is the original signer's public key. If it holds, each proxy signer, P_i, computes the proxy sharing

$$\sigma_i' = f'(i) + s_i \times e \bmod q. \tag{11}$$

2.2.3. The Proxy Issuing and Verification Phase

(1) The t or more actual signers have to execute the random number generation phase to obtain the secret output s_i' and public outputs

$$y = g^{c0}, \qquad C_1 = g^{c1}, \qquad C_2 = g^{c2}, \ldots, C_{t-1} = g^{ct-1} \pmod p, \tag{12}$$

where

$$s_i' = f'(i) = c_0 + c_1^i + c_2^{i2} + \cdots + c_{t-1} i^{t-1}. \tag{13}$$

(2) Then each actual signer uses his proxy signature key to issue a partial proxy signature such that $e' = h(y, m)$ and $\gamma_i = s_i' + \sigma_i' \times e \bmod q$, where m is message. Then, each actual signer reveals γ_i.

(3) Everyone can verify the validity of γ_i by the following equation:

$$g^{\gamma i} = \left(y \prod_{i=1}^{t-1} (C_i)^j \right)^i \times (y_0)^{h(mw,K)} K \prod_{i=1}^{t-1} \left((B_i)^j \right)^i \bmod p \times y_G \prod_{i=1}^{t-1} \left(\left((A_i)^j \right)^{ih(mw,K)} \right)^{h(y,m)} \bmod p. \tag{14}$$

(4) If the previous verification holds, the signature, on m is (m, T, e', k, m_w), where $T = c_0 + \sigma \times e' = f'(0) + f(0) \times e'$ can be computed by applying the Lagrange formula.

(5) To verify the validity of the signature, anyone can examine the following equation:

$$y' = g^T \times \left((y_0)^{h(mw,K)} K \right)^{-ei} \bmod p, \qquad e' = h\left(y', m \right). \tag{15}$$

2.3. Security Analysis of Kim et al. and Related Schemes. The Kim et al.'s [13] scheme has been shown insecure by Sun et al. [6] using the public key updating attack. Kim et al. proposed two types of threshold proxy signature schemes, which were the proxy-protected scheme and the proxy-unprotected scheme. In the proxy-protected scheme, the original cannot impersonate a proxy signer to issue a

valid proxy signature. The proxy signing key combines the original signer's secret sharing key and a secret value among the t proxy signers. Therefore, the original signer cannot obtain the proxy signing keys. This property is called proxy-protected. One major drawback in Kim et al.'s scheme is that the actual signers cannot be identified. This can be very inconvenient for internal auditing. Kim et al.'s scheme does not satisfy the known signer's requirement, proxy protection requirement, and the time constrain requirement. It does not satisfy the known signer's requirement as the actual signer cannot be identified. Also, it is necessary for a verifier to use the public information to check the validity of proxy signature. If the pubic information is not authenticated, the original signer is able to execute the (t, n) threshold proxy signature scheme to generate a valid proxy signature key by himself; that is, he plays the roles of the original signer and the proxy signers simultaneously. This is because a verifier is unable to distinguish whether the public information is created by the legal proxy group or by others (a dishonest original signer or unauthorized group). Hence, it does not satisfy the proxy protection requirement. This scheme does not have the ability to put time constraints on the threshold delegation.

In order to remedy the problem of unknown signers, Sun et al. [6] revised Kim et al.'s [13] proxy-protected type threshold proxy signature scheme and made the actual signers able to be identified. Sun et al.'s scheme is also insecure since any $n - 1$ proxy signers in the group can conspire to obtain secret key needed by the remainder of the group. Also, the computational as well as communicational overhead of Sun's scheme is high. With t or more proxy signers, it issues a proxy signature which has to generate and share a random number among them. In essence, it requires several expansion modular exponential computations and communications.

Unfortunately, Zhang's scheme [13] has also shown to be insecure by Lee, Hwang, and Wang. They have shown a dishonest proxy signer can cheat to get a signature which is generated by the original signer on any message with the condition that a conventional digital signature scheme is a variation of ElGamal type signature.

In 1991, Desmedt and Frankel proposed a threshold RSA signature scheme. This technique allows t out of n individuals to generate a signature for a message. The signature is on the behalf of group of n members; hence, we also call it group signature. Rivest et al. [1] extended the concepts and principles from Desmedt and Frankel's threshold RSA signature to develop a threshold RSA proxy signature scheme.

In 1999, Okamoto et al. [14] also suggested an enhanced proxy signature scheme based on both the Mambo-Usuda-Okamoto and Kim-Park-Won schemes. Later on, Sun, Lee, and Hwang examined the security of the Sun-Hsieh scheme based on the Kim-Park-Won scheme and proved that the scheme is not nonrepudiable. And also, a slightly modified version was suggested by them.

Hwang et al. [7] have shown that Sun's scheme has a security weakness. An adversary can impersonate a legal proxy signer to generate a proxy signature and the real proxy signer cannot deny having signed the proxy signature.

Lee et al. [15] have proposed the generalization of the $(t_1/n_1 - t_2/n_2)$ proxy signature scheme based on factorization of the square root modulo of a composite number. They proposed three kinds of proxy signature schemes: the $(t/n - 1)$, the $(1 - t/n)$, and the $(1 - 1)$ proxy signature schemes. In this, the actual original signer cannot deny delegating the warrant or proxy signature either.

Tzeng et al. [16] have proposed a batch verification scheme for multiple proxy signature to reduce proxy verification time. The proposed scheme is not efficient because it does not verify each proxy signature separately, but somehow it is secure because it can detect forged multiple proxy signatures without failure.

Hwang et al. [17] have proposed a multiproxy multisignature scheme which allows any u or more proxy signers from a designated group of v proxy signers to sign messages on behalf of any t or more original signers from a group of n original signers in total. In this, they only make (u, v) proxy signers to sign on behalf of them.

Lu et al. [18] have proposed a proxy signature scheme which allows the original signer to revoke delegations whenever necessary. In this, the authentication server will not issue the time-stamp unless the delegation has not been revoked or the delegation period specified in the warrant has not expired.

Yang et al. [19] have proposed an improvement of Hsu et al.'s scheme that is somewhat efficient in terms of computational complexity and communication cost. Different from Hsu et al.'s, the original signer only computes a common proxy share and broadcasts it to the proxy group. As compared with Hsu et al.'s scheme, the secret shares calculations is not required.

Tzeng et al. [20] have proposed a threshold multiproxy multisignature scheme with (t_3, n_3) shared verification. They have proposed the security on the basis of one-way hash function and discrete logarithm problem. They considered only a few attacks.

Tzeng et al. [21] have presented security analysis of the Hwang-Lin-Lu scheme. The Hwang-Lin-Lu scheme is vulnerable to forge attack.

Hwang et al. [22] have presented a generalized version of proxy signature scheme. A generalization version of the $(t_1/n_1 - t_2/n_2)$ proxy signature scheme is based on the elliptic curve discrete logarithm problem only. In this paper, the actual original signer cannot deny delegating the warrant or the proxy signature.

In 2001, Hsu et al. proposed a nonrepudiable threshold proxy signature scheme. Tsai et al. [23] proposed a scheme to remedy the weakness of the Hsu-Wu-Wu scheme. In this, neither the original signer nor a malicious proxy signer can forge the legal proxy signature.

Li et al. [24] have presented a generalized version of proxy signature scheme. A generalization version of the $(t_1/n_1 - t_2/n_2)$ is based on the discrete logarithm problem only. They discussed three kinds of proxy signature schemes: the $(t/n - 1)$, the $(1 - t/n)$, and the $(1 - 1)$ proxy signature schemes. The actual original signer cannot deny delegating the warrant or the proxy signature.

Hwang et al. [25] have discussed Hwang and Shi's scheme without using a one-way hash function. In their scheme,

an original signer needs not send a proxy certificate to proxy signer through secure channels. This scheme is vulnerable to the public key substitution attacks.

Hwang et al. [26] have presented a cryptanalysis of Sun's threshold proxy signature scheme. They have shown only that the secret key can be compromised by collusion attack.

Raman Kumar et al. [27] have presented a new scheme which includes the features and benefits of four schemes: Hwang et al., Wen et al., Geng et al. and Feng et al. They have compared these schemes and optimized their own proposed scheme on the basis of different parameters.

2.4. Review of Hwang et al.'s Scheme. In the HLL scheme, Hwang et al. [7] proposed a practical and efficient (t, n) threshold proxy signature scheme based on the RSA cryptosystem. This scheme uses only an RSA digital signature scheme and a simple Lagrange formula to share the proxy signature key.

There are three types of participants in the scheme: the original signer, the n proxy signer, and the combiner. The original signer allows a group of n proxy signers to sign a message. The combiner can be the secretary of the original signer. The proposed threshold proxy signature scheme can be divided into three phases:

(1) The proxy sharing phase;

(2) The proxy issuing phase;

(3) The verification phase.

In the proxy generation phase, the original signer computes the partial proxy signing keys from his private key and sends them to each designated proxy signer. In the proxy signature issuing phase, the proxy signers cooperatively create a valid signature on a message M. In the verification phase, the verifier can identify not only the original signer, but also the actual signers. P_0 stands for the original signer and $P_1, P_2, \ldots, P_n$ stand for the n proxy signers. N_i is a public RSA modulus for P_i such that N_i is a public RSA modulus for P_i and $N_i = p_i \times q_i$, where p_i and q_i are two secret large primes, while d_i is the private key for P_i and its corresponding public key is e_i, such that $d_i \times e_i = 1 \bmod \Phi(N_i)$, where $\Phi(N_i) = (p_i - 1) \times (q_i - 1)$. The parameters e_i and N_i can be published. The parameters d_i and $\Phi(N_i)$ are kept secret by the holder. $[M]^{di} \bmod N_i$ represents M signed with P_i's private key d_i, and $[M]^{ei} \bmod N_i$ represents M encrypted with P_i's public key e_i using the ordinary RSA cryptosystem. The message m_w stands for a warrant that is minted by the original signer and it contains important information such as the validity period of the proxy key, the identities of the proxy signers, and the original signer. In the proposed scheme, let $N_0 < N_i$ $(i = 1, 2, \ldots, n)$.

2.4.1. The Proxy Sharing Phase. Assume that an original signer P_0 delegates the power to sign messages to n members during s stipulated period. The steps to generate the proxy key are as follows.

(1) *Proxy Generation.* P_0 produces the group proxy signing key D and proxy verification key E, where

$$D = d_0^{mw} \bmod \Phi(N_0),$$
$$E = e_0^{mw} \bmod \Phi(N_0), \tag{16}$$

where $m_w = (P + T + r) \bmod \Phi(N_0)$.

P is the validity period of proxy signatures, T is the sum of identities of $P_0, P_1, \ldots, P_n$, and r is a random number.

Then P_0 publishes $\{m_w, E, [m_w, E]^{d0} \bmod N_0\}$.

(2) *Proxy Sharing.* P_0 selects a $t - 1$ degree polynomial,

$$f(x) = D + a_1 x + \cdots + a_{t-1} x^{t-1} \bmod \Phi(N_0), \tag{17}$$

where $a_1, a_2, \ldots, a_{t-1}$ are random numbers. Meanwhile, P_0 calculates proxy signer P_i's partial proxy signing key $k_i = f(i)$ and sends $[[k_i]^{d0} \bmod N_0, k_i]^{ei} \bmod N_i$ to the proxy signer P_i.

2.4.2. Proxy Share Generation. When proxy signer P_i receives $[[k_i]^{d0} \bmod N_0, k_i]^{ei} \bmod N_i$, he or she can get $\{[k_i]^{d0} \bmod N_0, k_i\}$ by his or her secret key d_i. And then P_i confirms the validity of k_i and keeps it secret.

(1) *The Proxy Signature Issuing Phase.* Let T denote the group members including any t or more proxy signers who want to generate a proxy signature on message M on behalf of P_0 cooperatively. Each proxy signer P_i uses the partial proxy signing key k_i to generate the partial signature

$$s_i = M^{ki} \bmod N_0. \tag{18}$$

Then P_i sends $\{[s_i, i]^{di} \bmod N_i, s_i\}$ to the combiner.

When the combiner receives all partial signature s_i from P_i, firstly, he or she verifies the validity of the partial proxy signature by checking if $[s_i, i]^{di \cdot ei} \bmod N_i = (s_i, i)$ or not. If all partial signatures are valid, the combiner computes the value of

$$v = \prod_{\substack{ID_a, ID_b \in T \\ a>b}} (ID_a - ID_b),$$
$$vLi = \prod_{\substack{ID_a, ID_b \in T \\ a>b}} (ID_a - ID_b) \prod_{j=1, j\neq i}^{t} \left(-\frac{ID_j}{(ID_i - ID_j)} \right). \tag{19}$$

Here,

$$\prod_{j=1, j\neq i} (ID_i - ID_j) \text{ is a factor of } \prod_{\substack{ID_a, ID_b \in T \\ a>b}}^{t} (ID_a - ID_b). \tag{20}$$

So vLi is an integer and the combiner need not compute the inverse of

$$\prod_{j=1, j\neq i}^{t} (ID_i - ID_j). \tag{21}$$

Finally, the combiner generates the signature S as follows:

$$S = \prod_{i \in T} s_i^{vLi} \bmod N_0. \quad (22)$$

The result of proxy signature is $\{v, S\}$.

2.4.3. The Proxy Signature Verification Phase. The verifier can verify the signature signed on behalf of the original signer by the following equation:

$$S^E = M^v \bmod N_0. \quad (23)$$

The original signer can differentiate the actual signer from the signature $s_i^{diei} \bmod N_i = s_i$. Then the original signer can trace the actual signers by e_i.

2.5. Conclusions from the Threshold Proxy Signature Schemes. All analyses indicated that the scheme fails to satisfy all the requirements except one or two. So, an enhanced threshold proxy signature scheme must satisfy all of the following basic requirements which can be called proxy requirements [7, 9–11].

(1) *Secrecy.* The original signer's private key is very important. It must be kept secret. If it is discovered, the security of the system is ruined. Therefore, the system must ensure that the private key never gets derived from any information such as the sharing of the proxy signing key or the original signer's public key. Furthermore, no proxy signers should be able to cooperatively derive the original signer's private key.

(2) *Proxy Protected.* Only a delegated proxy signer can generate his partial proxy signature. Even the original signer cannot masquerade as a proxy signer to generate a partial proxy signature. This property protects the authority of the proxy signer.

(3) *Unforgeability.* A valid proxy signature can only be cooperatively generated by t or more proxy signers. Nondelegated signers have no capability to generate a valid proxy signature. Also, $(t-1)$ or less proxy signers have no capability of forging a valid proxy signature.

(4) *Nonrepudiation.* Any valid proxy signature must be generated by t or more proxy signers. The verifier can make sure that the signed message is a correct one by using the proxy signing keys. The original signer cannot deny having delegated the power of signing messages to the proxy signers. Furthermore, the proxy signers cannot deny that they have signed the message.

(5) *Time Constraint.* The proxy signing keys can be used only during a stipulated period. Once expired, proxy signing keys become invalid; as a result, the signing capability of the proxy signers disappears. However, the original signer's private key can be repeatedly used. This is more suitable for use in the real world.

(6) *Known Signers.* For internal auditing purposes, the system is able to identify the actual signers in the original signer's private key. The proxy signer has the capability to sign on the behalf of the original signer, but from the proxy signing key, the proxy signer cannot recover the original signer's private key.

3. Our Scheme

The concept of threshold cryptosystems was first proposed by Katzenbeisser [28]. They adapted the ElGamal [3] public key cryptosystem and used the Lagrange interpolation or geometry to produce the shadows. In the history of proxy signature technological development, the $(1, n)$ threshold proxy signature technique was the first to come [7]. In $(1, n)$ proxy signature schemes [12, 14, 29], a legal proxy signature can be generated by a designated proxy signer by using a proxy signing key. However, in a (t, n) threshold proxy signature scheme, the original signer delegates the power of signing messages to a designated proxy group of n members. Any t or more proxy signers of the group can cooperatively issue a proxy signature on the behalf of the original signer, but $(t-1)$ or fewer proxy signers cannot. Previously, all of the proposed threshold proxy signature schemes, for instance, Lee et al. [30], ElGamal [2], Sun [13], and Mambo et al. [12], have been based on the discrete logarithm problem. However, the recently proposed threshold proxy signature schemes are based on the RSA cryptosystem [1] and the Lagrange coefficient. In 2003, Hwang et al. [7] proposed a practical and efficient (t, n) threshold proxy signature scheme based on the RSA cryptosystem. This scheme uses only an RSA digit signature scheme and a simple Lagrange formula to share the proxy signature key. In 2004, Wang et al. [8] pointed out a problem on the correctness of the HLL scheme. In 2005, Kuo and Chen [9] also indicated two security weaknesses in the HLL scheme and proposed a new scheme to overcome these weaknesses.

We compare the performance of four schemes, Hwang et al. [7], Kuo and Chen [9], Yong-Jun et al. [11], and Li et al. [10], with the performance of a scheme that has been proposed by the authors of this paper earlier and proposed an enhanced secure threshold proxy signature scheme. In the proposed scheme, both the combiner and the secret share holder can verify the correctness of the information that they are receiving from each other. Therefore, the enhanced threshold proxy signature scheme is secure and efficient against notorious conspiracy attacks. Table 2 gives the comparison of threshold proxy signature schemes based on the proxy requirements of each scheme.

4. Security Analysis of the Proposed Scheme

4.1. Factorization of the RSA Module. Factoring n: the fastest known factoring algorithm developed by Pollard is the General Number Field Sieve [1], which has running time for factoring a large number of size n, of order

$$O\left(\exp\left(\left(\frac{64}{9}\log n\right)^{1/3}(\log\log n)^{2/3}\right)\right). \quad (24)$$

Table 2: A comparison of threshold proxy signature schemes based on proxy requirements.

Serial number	Proxy signature scheme/requirements	Kim et al.	Sun et al.	HLL	Wen et al.	Enhanced scheme
1	Secrecy	Yes	Yes	No	No	Yes
2	Proxy protection	No	No	No	No	Yes
3	Unforgeability	Yes	No	No	No	Yes
4	Nonrepudiation	Yes	Yes	No	No	Yes
5	Time-constraint	No	No	Yes	Yes	Yes
6	Known signers	No	Yes	No	No	Yes

Table 3: (a) The number of operations needed to factor n with GNFS method. (b) Computing $\phi(n)$ without Factoring "n."

(a)

Digits	Number of operations	Time
100	9.6×108	16 minutes
200	3.3×1012	38 days
300	1.3×1015	41 years
400	1.7×1017	5313 years
500	1.1×1019	3.5×105 years
1024	1.3×1026	4.2×1012 years
2048	1.5×1035	4.9×1021 years

(b)

$q-p$	$(q+p)^2 = 4n + (q-p)^2$	$q+p = \sqrt{4n+(q-p)^2}$
1	885	$29.7489\cdots$
2	888	$29.7993\cdots$
3	893	$29.8831\cdots$
4	900	30

The method relies upon the observation that if integers x and y are, such that $x \neq y \pmod n$ and $x^2 = y^2 \pmod n$, then $\gcd(x-y, n)$ and $\gcd(x+y, n)$ are nontrivial factors of n.

Tables 3(a) and 3(b) give the number of operations needed to factor n with the GNFS method and the time required if each operation uses one microsecond, for various lengths of the number n (in decimal digits).

Computing $\phi(n)$ without factoring "n":
assume that $n = p \times q$, $p < q$,
since $(q+p)^2 - (q-p)^2 = 4pq = 4n$,
then $(q+p)^2 = 4n + (q-p)^2$ so $q + p = \sqrt{4n + (q-p)^2}$;
guess $q - p$ and then find $q + p$, so $\varphi(n) = n - (p+q) + 1$.

Example 1. Suppose $n = 221 (4n = 884)$.

So, $q-p = 4$ and $q+p = 30$; then $\varphi(n) = 221-30+1 = 192$ and $p = 13$, $q = 17$, $n = 13 \times 17$.

4.2. Lattices and Lattice Reduction of RSA Module

4.2.1. Lattice-Based Attacks on RSA. The following attacks have been tested for RSA modules:

(i) Hastad's attack;

(ii) Franklin-Reiter attack;

(iii) extension to Wiener's attack.

4.2.2. Lattices and Lattice Reduction. Given a set of m linearly independent vectors, $\{b_1, \ldots, b_m\}$ in R^n, the set of all real linear combinations of these vectors $V = \{\sum_{i=1}^{m} a_i b_i : a_i \in R\}$ is a vector subspace.

Gram-Schmidt process [28] takes one basis $\{b_1, \ldots, b_m\}$ and produces a basis $\{b_1^*, \ldots, b_m^*\}$ which is pairwise orthogonal:

$$b_1^* = b_1,$$
$$\mu_{i,j} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}, \quad \text{for } 1 \leq j < i \leq n, \tag{25}$$
$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*.$$

Example 2. Consider

$$b_1 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}, \qquad b_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$
$$b_1^* = b_1 = \begin{pmatrix} 2 \\ 0 \end{pmatrix},$$

$$\mu_{2,1} = \frac{\langle b_2, b_1^* \rangle}{\langle b_1^*, b_1^* \rangle} = \frac{1}{2},$$
$$b_2^* = b_2 - \mu_{2,1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{26}$$

Given a set of basis vectors $\{b_1, \ldots, b_m\}$ in R_n and $m \leq n$, a lattice $L = \{\sum_{i=1}^{m} a_i b_i : a_i \in Z\}$ is a set of all integer linear combinations of the b_i.

Definition 3. A basis $\{b_1, \ldots, b_m\}$ is called LLL reduced if the associated Gram-Schmidt basis $\{b_1^*, \ldots, b_m^*\}$ satisfies

$$\left|\mu_{i,j}\right| \leq \frac{1}{2} \quad \text{for } 1 \leq j < i \leq m,$$
$$\left\|b_1^*\right\|^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right) \left\|b_{i-1}^*\right\|^2 \quad \text{for } 1 < i \leq m. \tag{27}$$

For all nonzero, $\|b_1\| \leq 2^{(m-1)/2}\|x\|$, we have

$$\left\|b_1\right\| \leq 2^{m/4} \Delta^{1/m}, \quad \Delta = \left|\det\left(B^T B\right)\right|^{1/2}. \tag{28}$$

4.2.3. Security Levels of the RSA Module on Different Platforms. The following are the creation of key in seconds for different security levels which can be used for encryption and decryption.

The fields in Tables 4(a) and 4(b) have been generated by varying the values of security levels for both the Pentium and AlphaStation, respectively. It shows the various parameters generated for different security levels.

4.2.4. A General Coalition Attack against Threshold Signature Schemes. Though our modification can withstand the forgery attack suffered by the said to be [7, 9–11] threshold group signature scheme, there is a general coalition attack against threshold signature schemes. In the ordinary threshold signature scheme, the group's secret key is $f(0)$, and each member U_i has the secret share $f(x_i)$. If t or more malicious members pool their secret shares together, they can recover $f(0)$ by applying the Lagrange interpolating polynomial. Then each one of them can alone compute valid signatures for new messages on behalf of the group afterwards without the cooperation of other signers and without being detected by verifiers. Obviously, this violates the group's signing policy. Otherwise, if such coalition is permissive, other signers would follow this kind of dishonesty. Thus, each user can also alone compute valid group signatures after one coalition. It is terrible for threshold signature schemes. This coalition attack is inherent in many threshold signature schemes using threshold secret share scheme, as long as the secret key can be recovered from secret shares.

4.2.5. The Probability of Catching a User. The probability of catching a user in enhanced threshold proxy signature scheme depends on the number of identity pairs used in it. The more the pairs used, the greater the chance of catching the anonymous user. The probability of catching the anonymous user is

$$1 - \left(\frac{1}{2}\right)^n, \tag{29}$$

where n is the number of pairs used.

For example, if $n = 5$, then the chance of catching a user is 0.97.

Table 4: (a) Security levels of the RSA module on a 90 MHz Pentium platform. (b) Security levels of the RSA module on a 255 MHz digital AlphaStation.

(a)

Security level	Encrypt (blks/sec)	Decrypt (blks/sec)	Create Key (sec)
512 bit	370	42	0.45
768 bit	189	15	1.5
1024 bit	116	7	3.8

(b)

Security level	Encrypt (blks/sec)	Decrypt (blks/sec)	Create Key (sec)
512 bit	1020	125	0.26
768 bit	588	42	0.59
1024 bit	385	23	1.28

4.2.6. An Algorithm to Check Primality of Any General Number Given. The algorithm is as follows.

Input: integer $n > 1$.

(1) If $n = a^b$ for integers $a > 0$ and $b > 1$, output *composite*.

(2) Find the smallest r such that $o_r(n) > \log^2(n)$.

(3) If $1 < \gcd(a, n) < n$ for some $a \leq r$, output *composite*.

(4) If $n \leq r$, output *prime*.

(5) For $a = 1$ to $\lfloor \sqrt{\varphi(r)} \log(n) \rfloor$ do
if $(X + a)^n \neq X^n + a \pmod{X^r - 1, n}$, output *composite*.

(6) Output *prime*.

Here $o_r(n)$ is the multiplicative order of n modulo r, log is the binary logarithm, and $\varphi(r)$ is Euler's totient function of r.

If n is a prime number, the algorithm will always return *prime*: since n is prime, Steps 1 and 3 will never return *composite*. Step 5 will also never return *composite* because Step 2 is true for all prime numbers n. Therefore, the algorithm will return *prime* either in Step 4 or in Step 6.

Conversely, if n is composite, the algorithm will always return *composite*: if the algorithm returns *prime*, then this will occur in either Step 4 or Step 6. In the first case, since $n \leq r$, n has a factor $a \leq r$ such that $1 < \gcd(a, n) < n$, which will return *composite*. The remaining possibility is that the algorithm returns *prime* in Step 6. The authors' paper proves that this will not happen because the multiple equalities tested in Step 5 are sufficient to guarantee that the output is *composite* [31].

4.2.7. An Algorithm to Determine If a Number Is a Probable Prime. The Fermat primality test is a probabilistic test to determine if a number is a probable prime. The algorithm is as follows.

Inputs: n: a value to test for primality; k: a parameter that determines the number of times to test for primality

Output: *composite* if n is composite, otherwise *probably prime*

repeat k times:

pick a randomly in the range $[1, n-1]$

if $a^{n-1} \not\equiv 1 \pmod{n}$, then return *composite*

Return Probably Prime. Using fast algorithms for modular exponentiation, the running time of this algorithm is $O(k \times \log^2 n \times \log\log n \times \log\log\log n)$, where k is the number of times we test a random a and n is the value we want to test for primality [32].

4.2.8. The Commonly Supported and Used Algorithms in Protocol. Here are some of the more commonly supported and used algorithms in protocol (see Table 5).

5. Performance Analysis of the Proposed Scheme

The analysis reports of the proposed hypothesis for enhanced threshold proxy signature scheme are given as the following.

5.1. Entropy. In this case, the value of entropy is the measure of the tendency of a process, to be entropically *favored* or to proceed in a particular direction. Moreover, entropy provides an indication for a specific encryption method. We have analyzed our hypothesis on the basis of entropy generated.

Figure 1 shows that entropy for enhanced threshold proxy signature scheme. Figure 2 shows the compression ratio required in each scheme. Table 6 lists the name and compression ratio required in each scheme.

5.2. Floating Frequencies/Intuitive Synthesis. Floating frequencies/intuitive synthesis in its completed three parts entirety takes full advantage of the time complexity, space complexity, and communication overhead provided by the digital medium. We have calculated floating frequency of the threshold proxy signature scheme. Figure 3 shows that floating frequencies/intuitive synthesis for the enhanced threshold proxy signature scheme.

5.3. ASCII Histogram. The ASCII histogram proved to be very useful since it helped enormously in debugging code involving probability calculations with simple print statements. Probabilistic simulations are extremely hard to test because the results of a given operation are never strictly the same. However, they should have the same probability distribution, so by looking at the rough shape of the histogram, it tells you if your calculations are going in the right direction. In this context, we have calculated ASCII histogram for our threshold proxy signature scheme. Figure 4 shows that ASCII Histogram for enhanced threshold proxy signature.

5.4. Autocorrelation. This is mathematical representation of the degree of similarity between a given time series and a lagged version of itself over successive time intervals. It is the same a calculating the correlation between two different time series, except that the same time series is used twice—once in its original form and once lagged one or more time periods. The term can also be referred to as "lagged correlation" or "serial correlation". In this, we have calculated autocorrelation for threshold proxy signature scheme. Figure 5 shows the Autocorrelation for the enhanced threshold proxy signature scheme.

5.5. Histogram Analysis. A histogram is a graphical representation showing a visual impression of the distribution of data. We have analyzed histogram of for all threshold proxy signature schemes.

Detailed View. The detailed view of the histogram analysis of all schemes can be represented as follows.

Experiment 1. Histogram analysis of ⟨1HLL⟩. File size 12581 bytes. Descending sorted on frequency.

Figure 6 shows that radar chart showing histogram analysis for HLL threshold proxy signature scheme. Table 7 lists the histogram analysis for the HLL threshold proxy signature scheme.

Experiment 2. Histogram analysis of ⟨2KUOCHEN⟩. File size 11733 bytes. Descending sorted on frequency.

Figure 7 shows that radar chart showing histogram Analysis for KUOCHEN threshold proxy signature scheme. Table 8 lists the histogram analysis for the KUOCHEN threshold proxy signature scheme.

Experiment 3. Histogram Analysis of ⟨3GENGVRF⟩. File size 11259 bytes. Descending sorted on frequency.

Figure 8 shows that radar chart showing histogram analysis for the GENGVRF threshold proxy signature scheme. Table 9 lists the histogram analysis for the GENGVRF threshold proxy signature scheme.

Experiment 4. Histogram ANALYSIS of ⟨4FNGVERF⟩. File size 12067 bytes. Descending sorted on frequency.

Figure 9 shows that radar chart showing histogram analysis for the FNGVRF threshold proxy signature scheme. Table 10 lists the histogram analysis for the FNGVRF threshold proxy signature scheme.

Experiment 5. Histogram analysis of ⟨5THRSPROX⟩. File size 16897 bytes. Descending sorted on frequency.

Table 5: The commonly supported and used algorithms used in protocol.

Diffie-Hellman key exchange algorithm	First published public key algorithm, can be used only for exchanging keys	Uses recipient's public key to generate a secret key; public data is then sent to recipient who can now generate the secret key
DSA (digital signature algorithm)	Does not encrypt data, but produces a signature that can be verified A public key (of 3 parts) is calculated from a private key	Signing: input is data to be signed, private key, a random number; output is a signature, comprising 2 numbers called r & s Verifying: input is data to be verified, public key & s; output is a number called v; if $v = r$ then signature is verified
SHA (secure hash algorithm)	US government standard produced by NIST Based on MD4 algorithm. Revised version of 1995 called SHA-1	Takes a message of less than 2^{64} bits and produces a message digest/fingerprint of 160 bits
DSS (digital signature standard)	US government standard method	Uses DSA to sign a message digest/fingerprint produced by SHA
ElGamal (T ElGamal's algorithm)	Variant of Diffie-Hellman for encryption and decryption as well as key exchanges	Sometimes known as Diffie-Hellman in earlier versions of PGP
RSA (Rivest-Shamir-Adleman algorithm)	First main, and still, the most widely used general purpose public-key encryption algorithm	Encrypt message with public key to obtain confidentiality Encrypt message with private key to obtain authentication, integrity, and nonrepudiation
3DES triple DES (data enrcyption standard)	DES was the first widespread symmetric key encryption algorithm 3DES applies the algorithm 3 times for additional security	DES is a 56-bit key, 64-bit block cypher using multiple rounds of permutations and substitutions Now considered necessary to apply 3 times with 3 keys
CAST-128 (developed by Carlisle Adams and Stafford Tavares)	Modern symmetric key encryption algorithm CAST-128 is an implementation of the CAST design procedure	Uses key sizes of 40 to 128 bits (in 8-bit increments) with 16 rounds of 64-bit blocks of plaintext Has been extensively reviewed by cryptologists
IDEA (international data encryption algorithm)	Modern symmetric key encryption algorithm, designed as a replacement for DES	128-bit key block cypher encrypting 64-bit blocks of plaintext
AES/Rijndael (Daemen and Rijmen's algorithm)	Selected for the new "Advanced Encryption Standard" by NIST to replace DES	High performance and very secure algorithm, using key sizes of 128, 192, and 256 bits
RSA	Selected for the new "Advanced Encryption Standard" by NIST	High performance and very secure algorithm

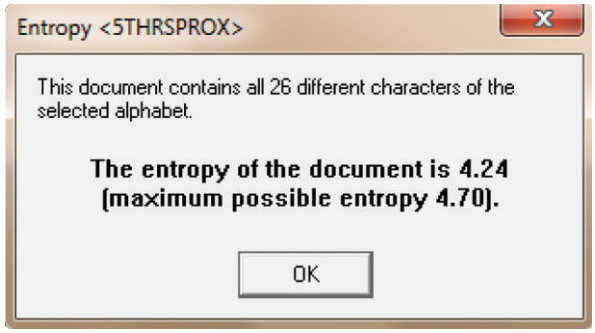

Figure 1: Entropy for the enhanced threshold proxy signature scheme.

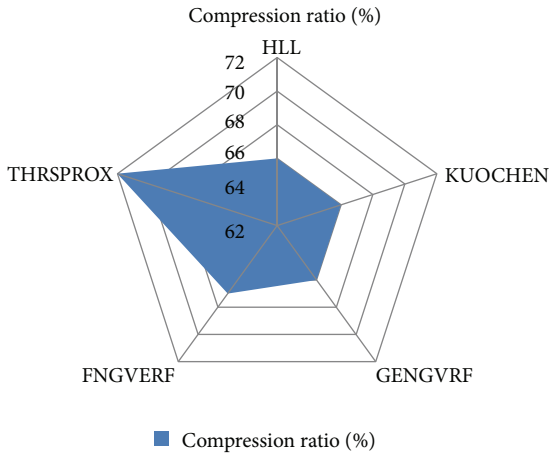

Figure 2: Radar chart showing compression ratio required in each scheme.

Table 6: Compression ratio (in %) for threshold proxy signature schemes.

Threshold proxy signature scheme	Compression ratio (in %)
HLL	66
KUOCHEN	66
GENGVRF	66
FNGVERF	67
THRSPROX	72

Figure 10 shows that Radar Chart showing histogram analysis for the enhanced threshold proxy signature scheme. Table 11 lists the histogram analysis for enhanced threshold proxy signature scheme.

Figure 11 shows that radar chart showing overall histogram analysis for all threshold proxy signature schemes. Table 17 lists the histogram analysis for the overall threshold proxy signature schemes.

5.6. Collusion Attack. The collusion attack is an action carried out by a given set of malicious users in possession of a copy of protected content that join together in order to obtain at the end of the attack procedure an unprotected asset. The attack is carried out by properly combining the protected copies of the multimedia documents collected by the colluders, according to the type of content and the kind of adopted protection system.

When the protection is assured by a data hiding algorithm, the collusion usually can take place in one of two different application frameworks: multimedia fingerprinting

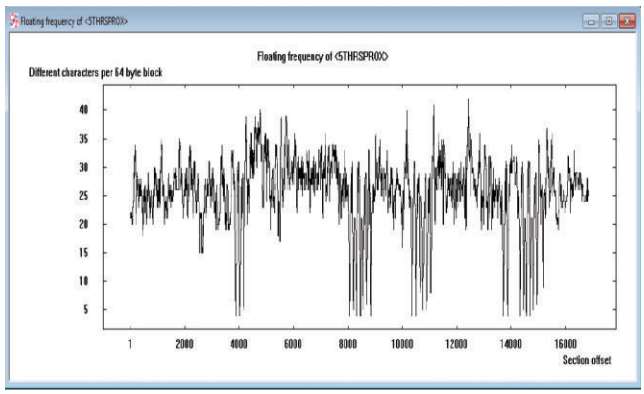

Figure 3: Floating frequencies/intuitive synthesis for the enhanced threshold proxy signature scheme.

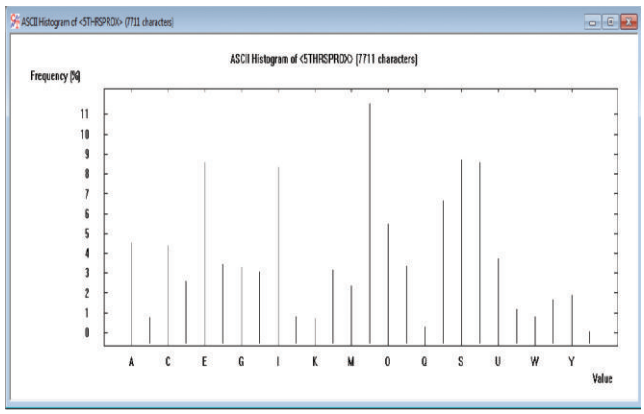

Figure 4: ASCII histogram for enhanced threshold proxy signature scheme.

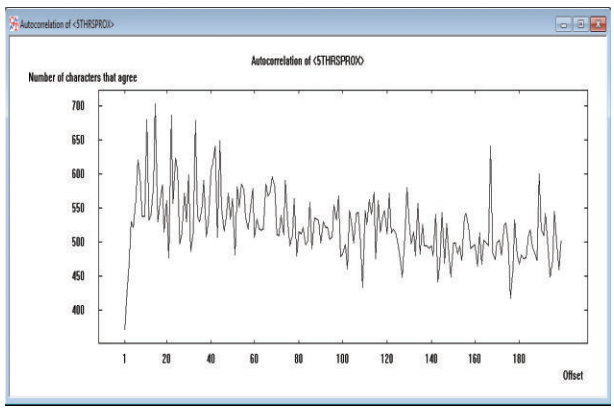

Figure 5: Autocorrelation for enhanced threshold proxy signature scheme.

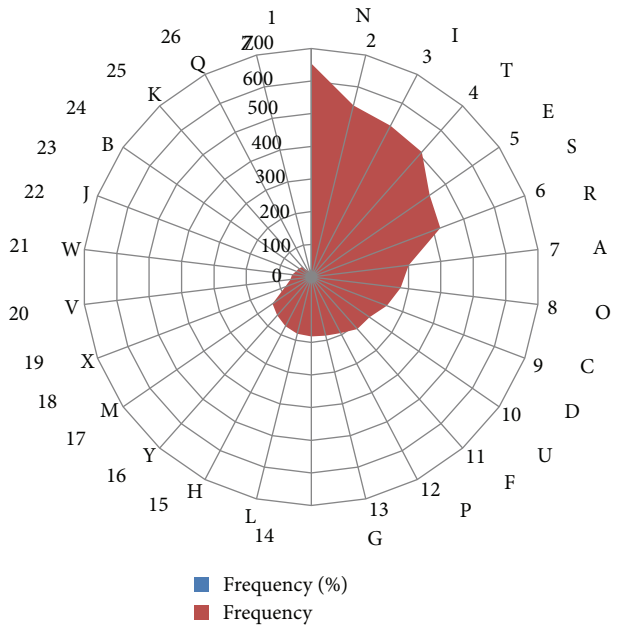

Figure 6: Radar chart showing histogram analysis for the HLL threshold proxy signature scheme.

Table 7: Histogram analysis for the HLL threshold proxy signature scheme.

Number	Substring	Frequency (in %)	Frequency
1	N	11.0343	654
2	I	9.1277	541
3	T	8.824	523
4	E	8.6216	511
5	S	7.4405	441
6	R	7.1368	423
7	A	5.0785	301
8	O	4.6567	276
9	C	4.1842	248
10	D	3.6612	217
11	U	3.5937	213
12	F	3.2732	194
13	P	3.1213	185
14	G	3.0707	182
15	L	3.0201	179
16	H	2.8682	170
17	Y	2.6489	157
18	M	2.4296	144
19	X	1.4341	85
20	V	1.0798	64
21	W	0.9954	59
22	J	0.8267	49
23	B	0.7761	46
24	K	0.6917	41
25	Q	0.3374	20
26	Z	0.0675	4

and ownership verification. In multimedia fingerprinting, a content owner, to protect his/her copyright, embeds a different code into each copy of the content distributed to each customer in order to be able to trace possible illegal usage of data and discover the source of the leakage of information; in this case, then, each colluder possesses a slightly different copy of the same multimedia content, and the attack consists in averaging all documents they have, trying to produce a new document in which the watermark is no longer present. If the number of averaged documents is large enough, the attack is very effective even without the introduction of perceptually significant degradation between the averaged multimedia document and the original one. In ownership verification, a content owner, to demonstrate that he/she is the authorized holder of the distributed content, embeds always the same code, linked to his/her identity, into different watermarked documents before they are distributed to the customers in such a way that the hidden code can be used to prove ownership in court if someone will infringe on his/her copyrights; in this case, then, each colluder possesses different multimedia documents, with the same hidden code, so that the attack is carried out by estimating the watermark by means of an average of all the different contents they have (this approach is suitable only for data-hiding systems in which the hidden watermark does not depend on the host data). Then the estimated watermark can be removed from all the documents hiding in it or even falsely inserted in other ones to generate fake watermarked documents [33].

One advantage of enhanced threshold proxy signature schemes is that they can prevent a "collusion attack" in which two key generation servers communicate with each other to get useful information about the user's private key. In essence, Figure 19 shows the overall communication architecture for the secure threshold proxy signature scheme based on RSA cryptosystem for known signers.

5.7. Security of RSA Signature

Existential forgery using a *key-only attack*:

Choose a random y
Compute $x = e_k(y)$
We have $y = \text{sig}_k(x)$, a valid signature of x.

Existential forgery using a known message *attack*:

Suppose $y = \text{sig}_k(x)$ and $y' = \text{sig}_k(x')$
Can check $e_k(y\,y' \bmod n) = x\,x' \bmod n$
So $y\,y' \bmod n = \text{sig}_k(x\,x' \bmod n)$.

Existential forgery using a *chosen message attack*:

To get a signature for x, find $x_1 x_2 = x \bmod n$
Query for signatures of x_1 and x_2.

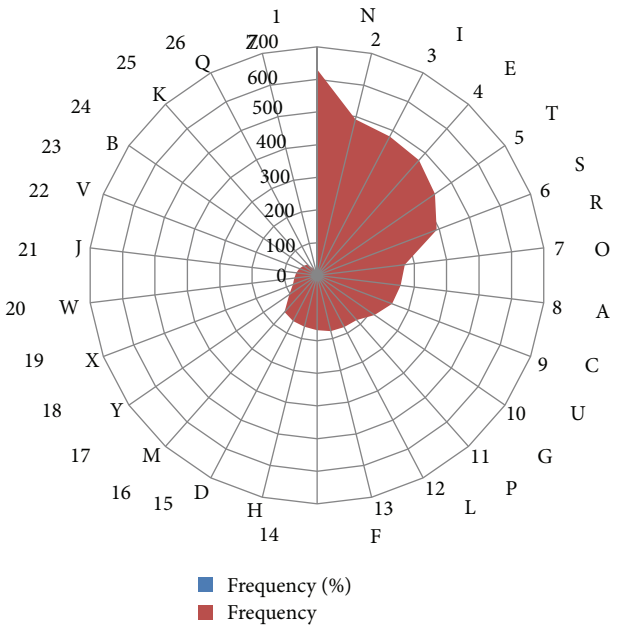

Figure 7: Radar chart showing histogram analysis for the KUOCHEN threshold proxy signature scheme.

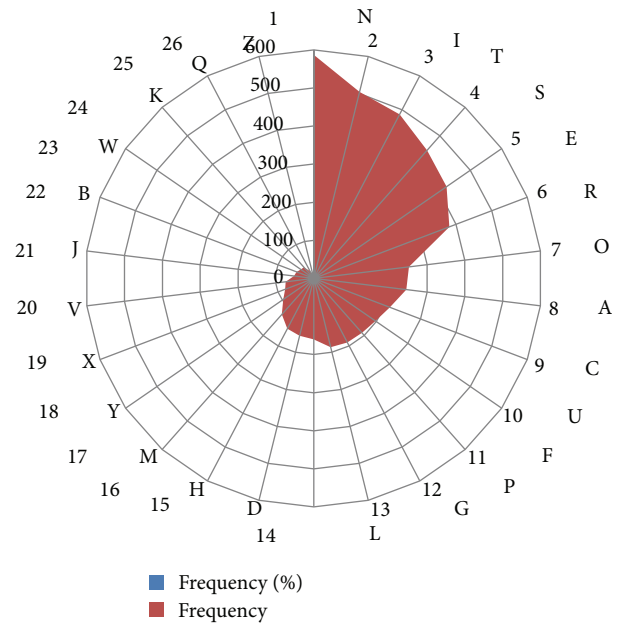

Figure 8: Radar chart showing histogram analysis for the GENGVRF threshold proxy signature scheme.

Table 8: Histogram analysis for the KUOCHEN threshold proxy signature schemes.

Number	Substring	Frequency (in %)	Frequency
1	N	11.3387	631
2	I	8.841	492
3	E	8.6253	480
4	T	8.4636	471
5	S	7.8886	439
6	R	7.044	392
7	O	4.8697	271
8	A	4.6361	258
9	C	4.4205	246
10	U	3.8455	214
11	G	3.2884	183
12	P	3.2165	179
13	L	3.1626	176
14	F	3.0189	168
15	H	2.9111	162
16	D	2.8392	158
17	M	2.6954	150
18	Y	1.8509	103
19	X	1.4196	79
20	W	1.2579	70
21	J	1.15	64
22	V	1.0422	58
23	B	0.9164	51
24	K	0.7907	44
25	Q	0.3953	22
26	Z	0.0719	4

Table 9: Histogram analysis for the GENGVRF threshold proxy signature schemes.

Number	Substring	Frequency (in %)	Frequency
1	N	10.9658	587
2	I	9.4153	504
3	T	9.079	486
4	S	8.3878	449
5	E	7.9208	424
6	R	7.1175	381
7	O	4.7076	252
8	A	4.5769	245
9	C	3.9978	214
10	U	3.6802	197
11	F	3.5681	191
12	P	3.5494	190
13	G	3.4747	186
14	L	2.989	160
15	D	2.8956	155
16	H	2.7835	149
17	M	2.3538	126
18	Y	1.8121	97
19	X	1.4945	80
20	V	1.3824	74
21	J	0.9714	52
22	B	0.8967	48
23	W	0.7659	41
24	K	0.7286	39
25	Q	0.411	22
26	Z	0.0747	4

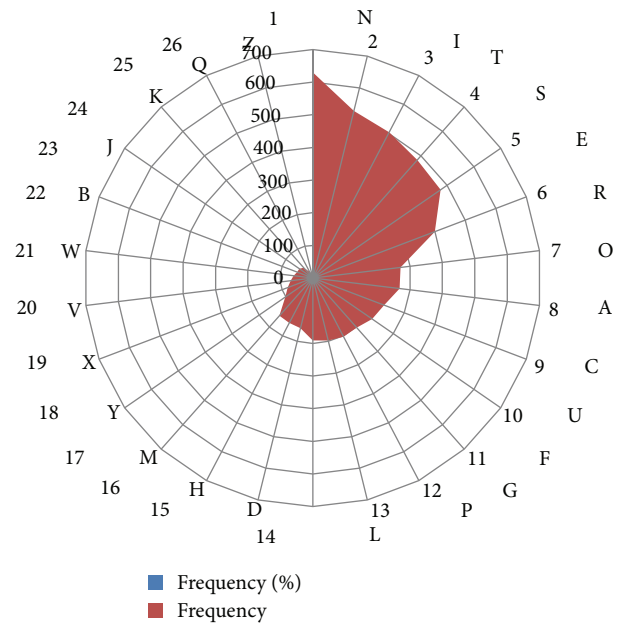

Figure 9: Radar chart showing histogram analysis for the FNGVRF threshold proxy signature scheme.

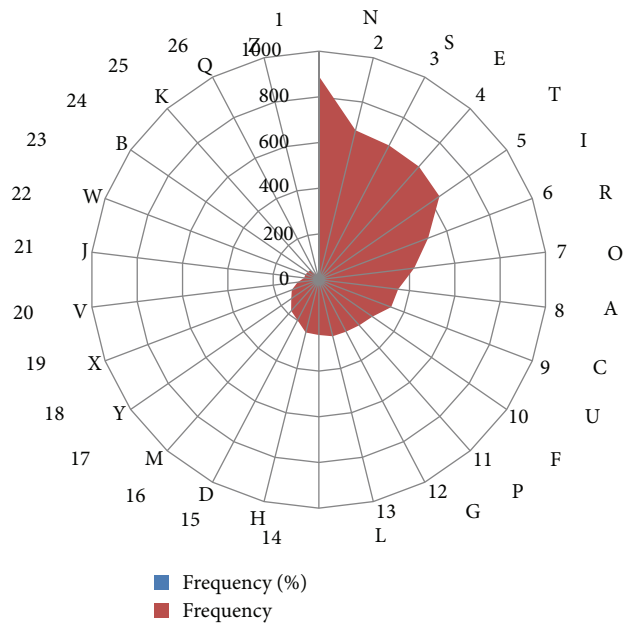

Figure 10: Radar chart showing histogram analysis for the enhanced threshold proxy signature scheme.

Table 10: Histogram analysis for the FNGVERF threshold proxy signature schemes.

Number	Substring	Frequency (in %)	Frequency
1	N	10.947	630
2	I	9.1573	527
3	T	8.7576	504
4	S	8.3927	483
5	E	8.2711	476
6	R	6.9505	400
7	O	4.7089	271
8	A	4.6568	268
9	C	4.066	234
10	U	3.8401	221
11	F	3.5274	203
12	G	3.5274	203
13	P	3.4231	197
14	L	3.3189	191
15	D	2.7454	158
16	H	2.6933	155
17	M	2.6759	154
18	Y	1.7724	102
19	X	1.4248	82
20	V	1.1816	68
21	W	1.0252	59
22	B	0.8862	51
23	J	0.8688	50
24	K	0.7298	42
25	Q	0.3823	22
26	Z	0.0695	4

Table 11: Histogram analysis for the enhanced threshold proxy signature schemes.

Number	Substring	Frequency (in %)	Frequency
1	N	11.5549	891
2	S	8.7278	673
3	E	8.5981	663
4	T	8.5722	661
5	I	8.3258	642
6	R	6.6399	512
7	O	5.4597	421
8	A	4.539	350
9	C	4.3963	339
10	U	3.696	285
11	F	3.4626	267
12	P	3.3329	257
13	G	3.307	255
14	L	3.1384	242
15	H	3.0865	238
16	D	2.5937	200
17	M	2.3603	182
18	Y	1.8934	146
19	X	1.634	126
20	V	1.2061	93
21	J	0.83	64
22	W	0.817	63
23	B	0.7522	58
24	K	0.7133	55
25	Q	0.3112	24
26	Z	0.0519	4

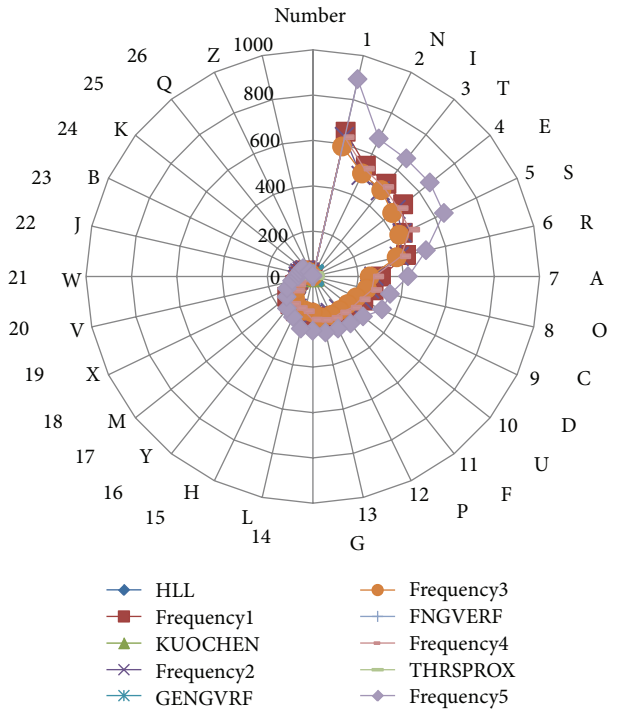

Figure 11: Radar chart showing overall analysis for all threshold proxy signature schemes.

5.8. Forgery Attack. RSA function is a multiplicative homomorphism; for all x, y,

$$f(xy \bmod n) = f(x) f(y) \bmod n,$$
$$f^{-1}(xy \bmod n) = f^{-1}(x) f^{-1}(y) \bmod n. \quad (30)$$

More generally,

$$f^{-1}\left(\prod x_i \bmod n\right) = \prod \left(f^{-1}(x_i)\right) \bmod n. \quad (31)$$

The property not only is exploited in most forgery attacks on RSA signatures, but also enhances recent security proofs.

5.9. Friedman Test. The Friedman test is a nonparametric statistical test developed by the US economist Milton Friedman. Similar to the parametric repeated measures ANOVA, it is used to detect differences in treatments across multiple test attempts. The procedure involves ranking each row (or block) together, then considering the values of ranks by columns. Applicable to complete block designs, it is thus a special case of the Durbin test. The Friedman test is used for one-way repeated measures analysis of variance by ranks. In its use of ranks, it is similar to the Kruskal-Wallis one-way analysis of variance by ranks. We have tested our hypothesis against the Friedman test [31].

Figure 12 shows the chart showing Friedman test for all threshold proxy signature schemes.

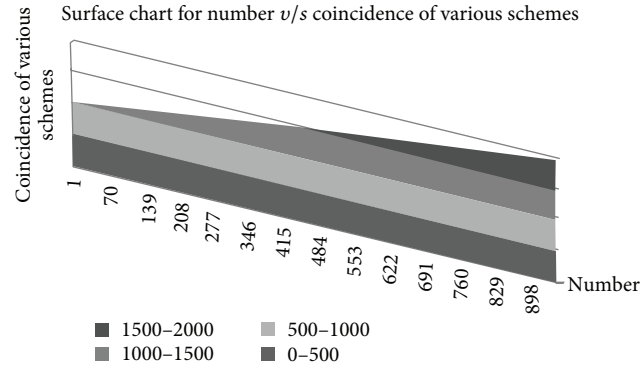

Figure 12: Chart showing Friedman test for all threshold proxy signature schemes.

6. Implementation and Comparison of the Proposed Scheme

The implementation of the schemes has been done.

6.1. Time Complexity. When determining the time complexity of an algorithm, we measure how fast the computing requirements grow as the size of the input grows.

6.1.1. Readings. Readings are taken for two scenarios described below. Each reading shown in Table 11 is the average of 30 readings.

Scenario 1: when number of threshold signers, $t = 1$.

Scenario 2: when number of threshold signers, $t = n$.

Table 12 shows the readings of execution time (in microseconds) when the number of threshold signers is equal to 11 and Table 13 shows the readings of execution time (in microseconds) and when the number of threshold signers is equal to number of proxy signers.

6.1.2. Graphs. We plot the graphs with the following:

(1) execution time (in microseconds) on Y-axis,

(2) number of proxy signers (N) on X-axis.

Scenario 1: when number of threshold signers, $t = 1$.

Scenario 2: when number of threshold signers, $t = n$.

6.1.3. Results. Figures 13(a), 13(b), 14(a), and 14(b) show that for both scenarios, Geng et al. scheme has the maximum time complexity and the proposed scheme has the minimum time complexity; also Fengying et al.'s time complexity falls very near the proposed scheme. We find that the proposed and Fengying et al.'s schemes have minimum time complexities. The time complexity of HLL scheme is more in case of scenario 1 (for $t = 1$) but less in case of scenario ($t = n$). This is because it has an extra constant overhead to verify the validity of the stipulated period. As this overhead is constant, hence, for scenario 1, HLL scheme has more time complexity as compared to other schemes. The proposed scheme has less time complexity as compared to other schemes because it need not compute the inverse when we compute the value

Table 12: Variation of time with number of signers for scenario 1 ($t = 1$).

Number of signers N	Time (in microseconds)				
	HLL	KC	Geng	Fengying	Proposed
1	128.0267	118.5867	123.6233	100.5567	98.52333
2	135.92	135.7567	135.46	107.42	103.9967
3	145.9433	142.23	146.5733	110.98	112.6033
4	163.4033	152.1533	168.17	115.01	116.68
5	164.8167	157.0067	172.7933	128.6167	122.1067
6	167.7767	175.15	194.65	130.22	132.5533
7	176.0667	178.7533	196.7933	135.69	135.77
8	205.11	188.2633	213.01	138.0067	137.9133
9	207.7133	189.5567	224.61	155.59	146.92
10	214.1167	191.36	226.2533	159.64	158.2633

Table 13: Variation of time with number of signers for scenario 2 ($t = n$).

Number of signers N	Time (in microseconds)				
	HLL	KC	Geng	Fengying	Proposed
1	128.0267	118.5867	123.6233	100.5567	98.52333
2	138.9933	147.0233	152.16	115.6033	117.08
3	163.9733	175.71	191.2133	138.89	139.5033
4	177.2367	206.82	225.47	166.2267	165.08
5	196.0533	216.4767	254.1533	190.6533	179.8767
6	217.9833	257.7233	276.2533	214.4267	209.7767
7	252.38	282.3267	305.4967	249.96	233.82
8	261.4233	313.67	337.4967	283.1467	256.4967
9	304.9767	346.7833	375.7	315.74	270.6067
10	324.8367	387.2833	415.05	350.9433	303.99

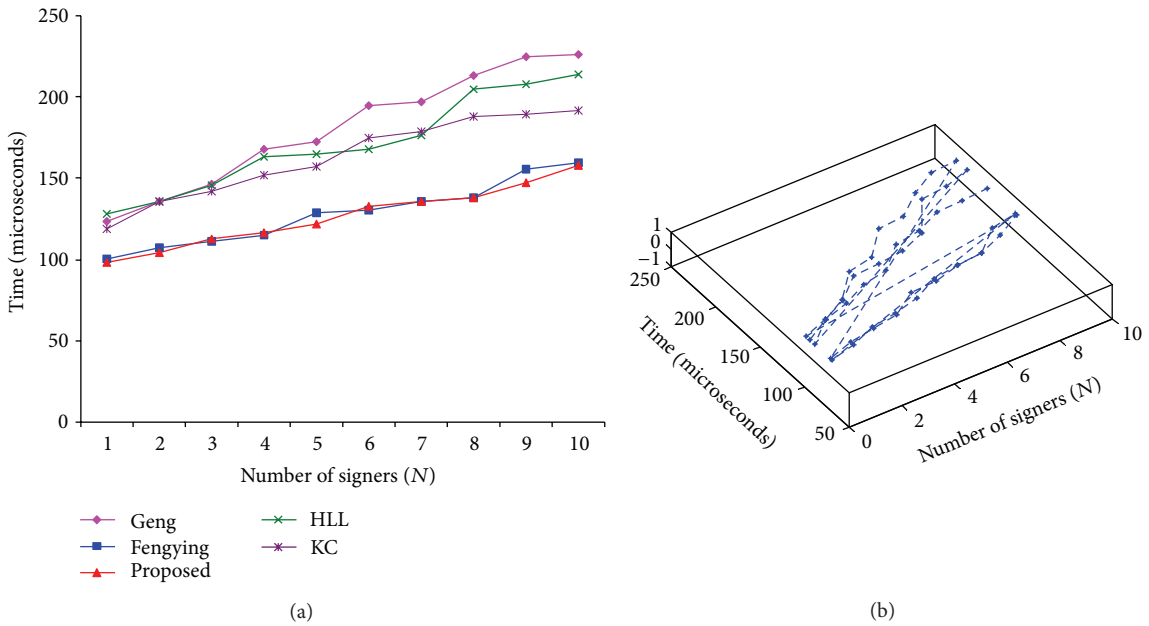

Figure 13: (a) Comparison of schemes based on time complexity when $t = 1$. (b) Three-dimensional view for comparison of schemes based on time complexity when $t = 1$.

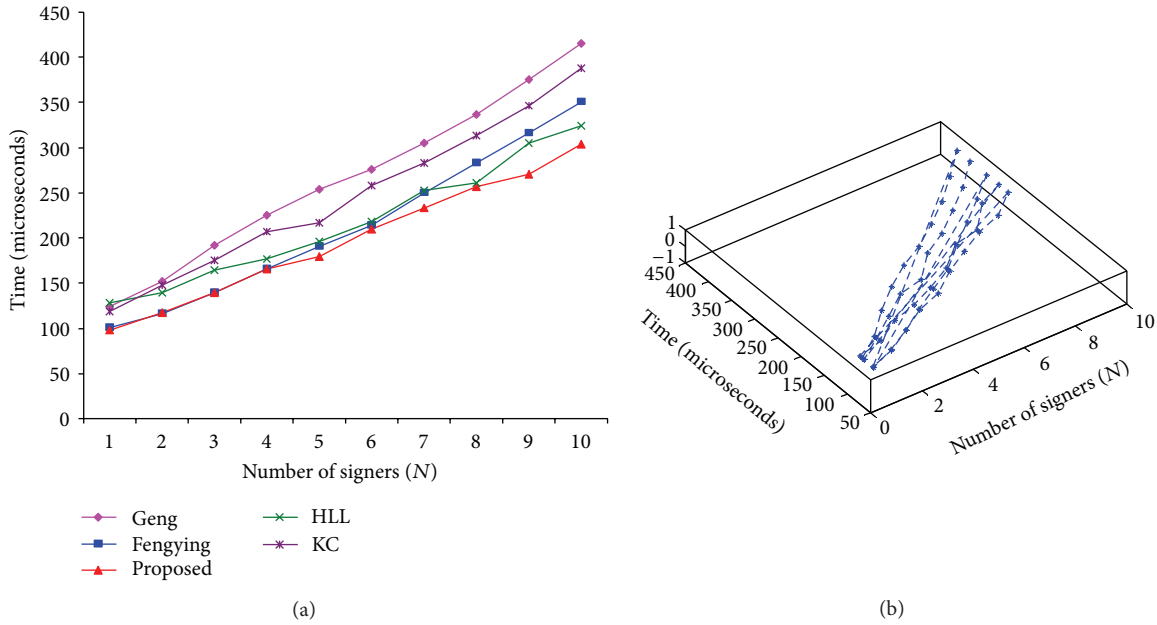

FIGURE 14: (a) comparison of schemes based on time complexity when $t = n$. (b) Three-dimensional view for comparison of schemes based on time complexity when $t = n$.

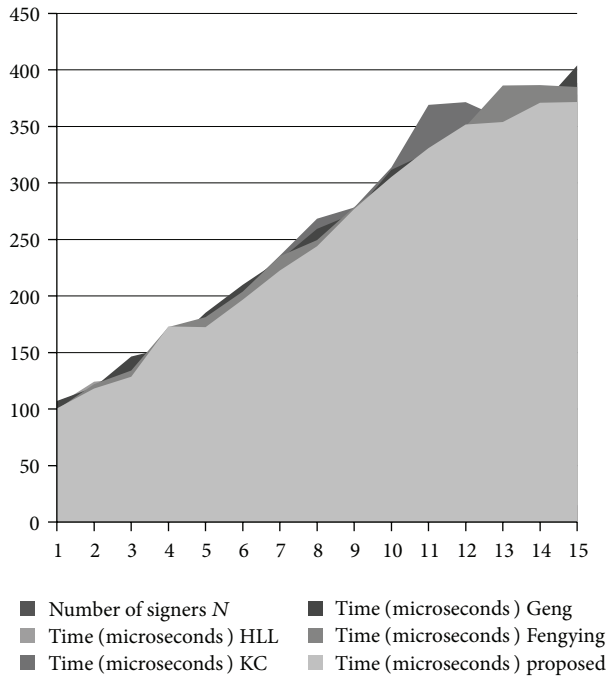

FIGURE 15: Comparison of schemes based on time complexity when $t = 1$ and n.

of Lagrange coefficient (Figure 15), which makes the scheme more efficient [11]. Also, we do not verify message warrant, which saves the extra overhead, that was incurred in case of KC scheme.

Figures 13(a), 13(b), 14(a), and 14(b) show that for both scenarios, the execution time increases with the increase in the number of proxy signers. This is because as the number of proxy signers increases, the number of computations also increases, due to, which the execution time increases. The number of computations increases because for each proxy signers, following parameters have to be computed.

(1) RSA modulus, N_i.

(2) Private key, d_i.

(3) Public key, e_i.

(4) p_i and q_i.

(5) $\Phi(N_i)$.

(6) K_i.

The more the number of proxy signers, the more the computations. The graphs generated also confirm this point. Also, we observe that in the graphs in Figures 14(a) and 14(b) the execution time is more in case of $t = n$ than in case of $t = 1$. It is because the number of computations increases with the number of threshold signers as well. For each threshold signer, the following computations have to done.

(1) Partial proxy signatures, s_i.

(2) The Lagrange coefficient, L_i.

The more the number of threshold proxy signers, the more the computations as confirmed by the graphs generated.

Scenario 3: when number of threshold signers, $t = 1$ and n.

Table 14 shows the readings of execution time (in microseconds) when the number of threshold signers is equal to 1 and is equal to number of proxy signers, where number of Signers $N = 15$, days = 80, and threshold number of Signers $N = 15$.

Scenario 3: when number of threshold signers, $t = 1$ and n.

Table 14: Variation of time with number of signers for scenario 3.

Number of signers N	Time (in microseconds)				
	HLL	KC	Geng	Fengying	Proposed
1	103.296	106.593	107.142	99.45	100.549
2	124.175	118.132	119.230	121.978	118.132
3	127.473	137.362	146.450	134.066	128.571
4	154.945	156.044	154.395	172.527	173.076
5	179.670	173.626	185.165	181.268	172.428
6	195.054	203.846	209.890	203.846	196.703
7	216.043	235.714	231.868	235.164	222.527
8	256.593	268.329	259.340	249.400	243.956
9	265.187	278.219	272.967	278.571	276.923
10	286.264	313.626	311.538	304.285	305.274
11	347.472	369.010	326.923	318.241	330.769
12	364.325	371.428	345.114	350	351.648
13	357.143	356.593	368.021	386.153	353.846
14	373.143	364.286	367.142	386.505	370.879
15	378.021	388.461	403.890	384.835	371.538

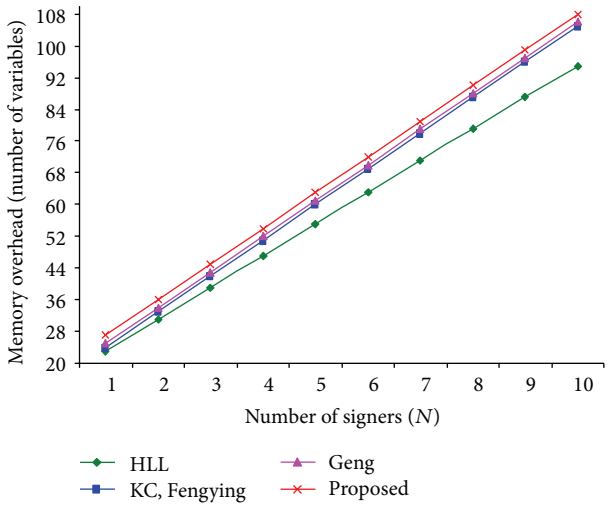

Figure 16: Comparison of schemes based on space complexity when $t = 1$.

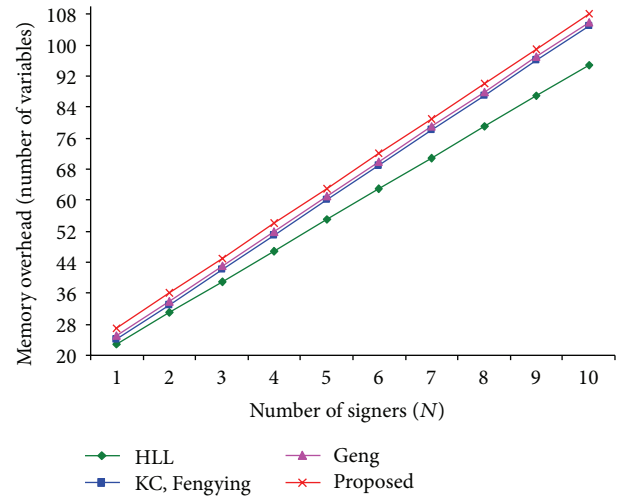

Figure 17: Comparison of schemes based on space complexity when $t = n$.

6.2. Space Complexity. The space complexity of a program is the number of elementary objects that this program needs to store during its execution. We generate graphs to analyze the space complexity of the schemes.

6.2.1. Estimation Based on Number of Variables. Table 14 lists the name and number of variables required in each scheme. Beside the variables shown in Table 14, a character string is required to store the message M which has to be signed.

6.2.2. Graphs. We plot the graphs with the following:

(1) memory overhead (in terms of number of variables) on y-axis,

(2) Number of proxy signers (N) on x-axis.

The graphs have been shown in Figures 16 and 17.

6.2.3. Results. Figures 16 and 17 have been generated by varying the values of n for both scenarios in Table 15, which shows the memory overhead in terms of memory variables required for each scheme. In Figures 16 and 17, we see that for both scenarios, the proposed scheme has the maximum space complexity and the HLL scheme has the minimum space complexity. The complexity of KC, Geng et al., and Fengying et al. scheme falls between the two schemes' space complexity with KC and Fengying et al. schemes having the same space complexity. We note that in Table 4, the proposed scheme requires only $(n + 3)$ more variables than the HLL scheme, which has the minimum space complexity. For example, the proposed scheme requires only 4 extra variables than the HLL

Table 15: Space Requirements of the Schemes.

Serial no.	Scheme									
	HLL		KC		Geng		Fengying		Proposed	
	Variables	Quantity	Variables	Quantity	Variables	Quantity	Variables	Quantity	Variables	Quantity
1.	$N_i, d_i, e_i, \phi(N_i), p_i, q_i$	$6 \times (n+1)$	$N_i, d_i, e_i, \phi(N_i), p_i, q_i$	$6 \times (n+1)$	$N_i, d_i, e_i, \phi(N_i), p_i, q_i$	$6 \times (n+1)$	$N_i, d_i, e_i, \phi(N_i), p_i, q_i$	$6 \times (n+1)$	$N_i, d_i, e_i, \phi(N_i), p_i, q_i$	$6 \times (n+1)$
2.	K_i, i	$2 \times n$	K_i, i, ID_i	$3 \times n$	K_i, i, ID_i	$3 \times n$	K_i, i, ID_i	$3 \times n$	K_i, i, ID_i	$3 \times n$
3.	s_i, L_i	$2 \times t$	s_i, L_i	$2 \times t$	s_i, L_i	$2 \times t$	s_i, L_i	$2 \times t$	s_i, L_i	$2 \times t$
4.	a_i	$t-1$	a_i	$t-1$	a_i	$t-1$	a_i	$t-1$	a_i	$t-1$
5.	D, E, S, P, m_w, t, n	5×1	D, E, S, P, m_w, t, n	5×1	D, E, S, P, m_w, t, n, v	6×1	D, E, S, P, m_w, t, n	5×1	$D, E, S, P, m_w, t, n, v, T, r$	8×1
Total	$8n + 3t + 12$		$9n + 3t + 12$		$9n + 3t + 13$		$9n + 3t + 12$		$9n + 3t + 15$	

Table 16: Comparison of communication overheads of the threshold proxy signature schemes.

Schemes	HLL		KC		Geng		Fengying		Proposed	
Number of Communications	PS	PSIV	PS	PSIV	PS	PSIV	PS	PSIV	PS	PSIV
Number of transmissions	n	t	n	t	n	t	n	t	n	t
Number of broadcasts	3	0	3	0	3	0	3	0	3	0
Total	$n+t+3$		$n+t+3$		$n+t+3$		$n+t+3$		$n+t+3$	

PS: proxy sharing phase.
PSIV: proxy signature issuing and verification phase.

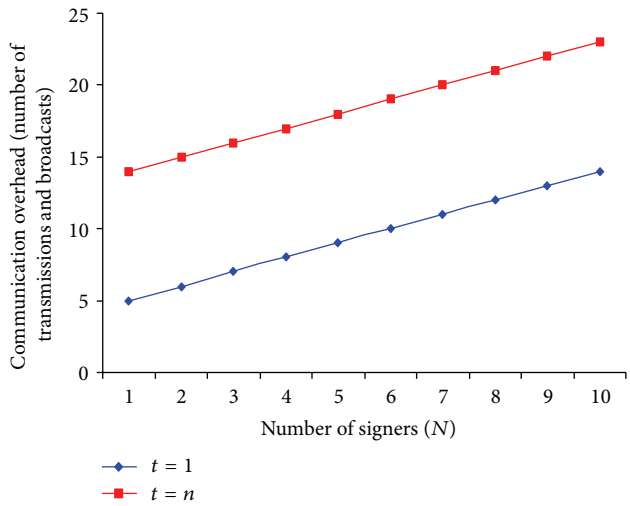

Figure 18: Comparison of schemes based on communication overhead.

scheme for $N = 1$, which has the minimum space complexity, and for $N = n$, the proposed scheme requires only 13 more variables than the HLL scheme. Also, this difference can be easily accommodated with the development of very large storage devices. Hence, we can see that the space complexities of the schemes are almost equal and the performance of the schemes in the case of the space complexity is almost the same.

Scenario 1: when number of threshold signers, $t = 1$.

Scenario 2: when number of threshold signers, $t = n$.

Figures 16 and 17 also show that for both the scenarios, the space complexity increases with the increase in the number of proxy signers. This is because as the number of proxy signers increases, the number of variables also increases due to which the memory requirements increase. Memory overhead increases because for each proxy signers, the following parameters are required.

(1) RSA modulus, N_i.

(2) Private key, d_i.

(3) Public key, e_i.

(4) p_i and q_i.

(5) $\Phi(N_i)$.

(6) K_i.

The more the number of proxy signers, the more the memory requirements. The figures generated also confirm this point.

Also, we observe in the graphs that memory overhead is more in the case of $t = n$ than in the case of $t = 1$. It is because the number of variables increases with the number of threshold signers as well. For each threshold signer, the following variables have to be declared.

(1) Partial proxy signatures, s_i.

(2) The Lagrange coefficient, L_i.

The more the number of threshold proxy signers, the more the memory requirements as confirmed by the graphs generated.

6.3. Communication Overhead. The communication overhead includes two types of communication in the schemes:

(1) number of transmissions;

(2) number of broadcasts.

6.3.1. Comparison of the Schemes. Table 6 shows the communication overhead for each scheme for various phases. In Figure 6, we find that the communication overhead of all the schemes is the same.

6.3.2. Results. Figure 18 has been generated from Table 15, which can be referred to to find the number of transmissions and broadcasts to compute the communication overhead for all the schemes. Table 16 can be referred to to find the comparison between threshold proxy signature schemes based upon proxy requirements. As evident from Figure 18, the communication overhead of all the schemes is the same.

Figure 19 shows the overall communication architecture for the secure threshold proxy signature scheme based on RSA cryptosystem for known signers. It sends a message after logging in to the system. It also sends a message to the other client using the secure threshold proxy signature scheme based on the RSA cryptosystem for known signers. In this, the delegates can send the message to the n proxy signers with security. After sending the message, it logs out from the system. We have improved the results after improving the performance of four schemes: Hwang et al. [7], Kuo and Chen [9], Yong-Jun et al. [11], and Li et al. [10], with the performance of a scheme that has been proposed by the authors of this paper earlier. In this paper, we compared the various threshold proxy signature schemes based on how they violate proxy signature requirements. Table 16 summarizes the comparison of four threshold proxy signature schemes: Kim et al.'s scheme, Sun's scheme, HLL

Table 17

*	Schemes	HLL	Frequency 1	KUOCHEN	Frequency 2	GENGVRF	Frequency 3	FNGVERF	Frequency 4	THRSPROX	Frequency 5
No.	Substring	Frequency (in %)	Frequency	Frequency (in %)	Frequency	Frequency (in %)	Frequency	Frequency (in %)	Frequency	Frequency (in %)	Frequency
1	N	11.0343	654	11.3387	631	10.9658	587	10.947	630	11.5549	891
2	I	9.1277	541	8.841	492	9.4153	504	9.1573	527	8.7278	673
3	T	8.824	523	8.6253	480	9.079	486	8.7576	504	8.5981	663
4	E	8.6216	511	8.4636	471	8.3878	449	8.3927	483	8.5722	661
5	S	7.4405	441	7.8886	439	7.9208	424	8.2711	476	8.3258	642
6	R	7.1368	423	7.044	392	7.1175	381	6.9505	400	6.6399	512
7	A	5.0785	301	4.8697	271	4.7076	252	4.7089	271	5.4597	421
8	O	4.6567	276	4.6361	258	4.5769	245	4.6568	268	4.539	350
9	C	4.1842	248	4.4205	246	3.9978	214	4.066	234	4.3963	339
10	D	3.6612	217	3.8455	214	3.6802	197	3.8401	221	3.696	285
11	U	3.5937	213	3.2884	183	3.5681	191	3.5274	203	3.4626	267
12	F	3.2732	194	3.2165	179	3.5494	190	3.5274	203	3.3329	257
13	P	3.1213	185	3.1626	176	3.4747	186	3.4231	197	3.307	255
14	G	3.0707	182	3.0189	168	2.989	160	3.3189	191	3.1384	242
15	L	3.0201	179	2.9111	162	2.8956	155	2.7454	158	3.0865	238
16	H	2.8682	170	2.8392	158	2.7835	149	2.6933	155	2.5937	200
17	Y	2.6489	157	2.6954	150	2.3538	126	2.6759	154	2.3603	182
18	M	2.4296	144	1.8509	103	1.8121	97	1.7724	102	1.8934	146
19	X	1.4341	85	1.4196	79	1.4945	80	1.4248	82	1.634	126
20	V	1.0798	64	1.2579	70	1.3824	74	1.1816	68	1.2061	93
21	W	0.9954	59	1.15	64	0.9714	52	1.0252	59	0.83	64
22	J	0.8267	49	1.0422	58	0.8967	48	0.8862	51	0.817	63
23	B	0.7761	46	0.9164	51	0.7659	41	0.8688	50	0.7522	58
24	K	0.6917	41	0.7907	44	0.7286	39	0.7298	42	0.7133	55
25	Q	0.3374	20	0.3953	22	0.411	22	0.3823	22	0.3112	24
26	Z	0.0675	4	0.0719	4	0.0747	4	0.0695	4	0.0519	4

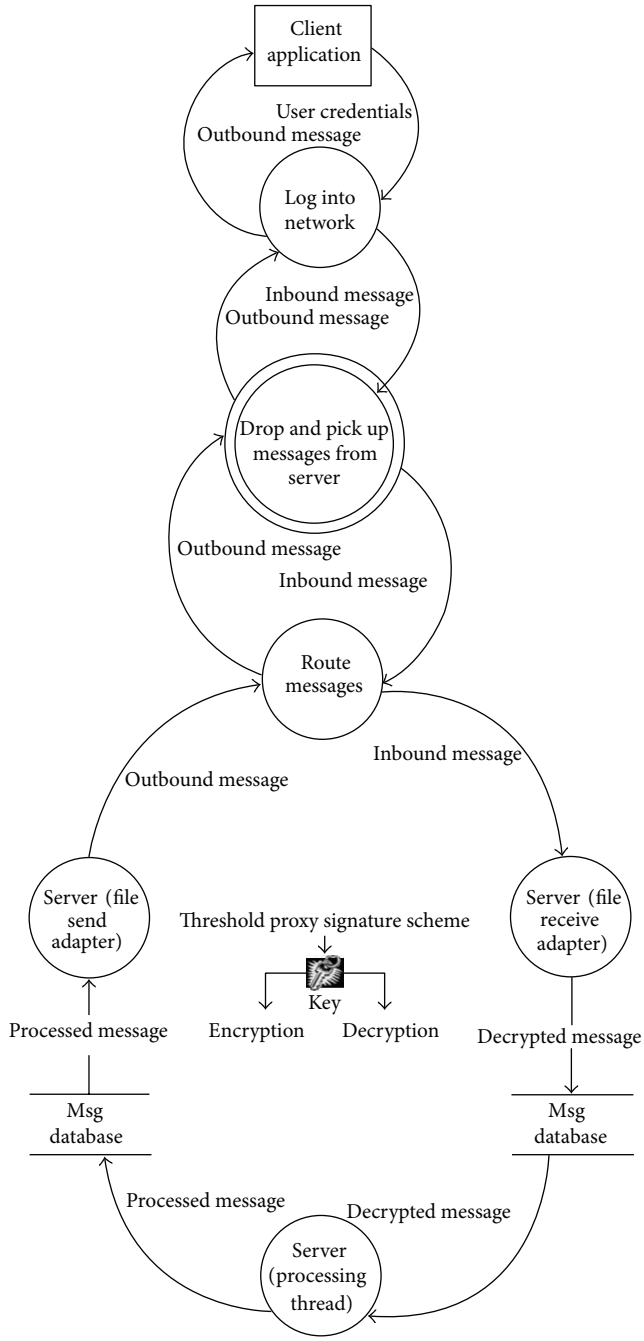

Figure 19: Overall communication architecture for secure threshold proxy signature scheme based on RSA cryptosystem.

scheme, and Wen et al.'s scheme. We also propose a new scheme which includes the features and benefits of the two schemes: Fengying et al. and Geng et al. The main advantages of our proposed scheme are as follows.

(i) It achieved the property of nonrepudiation.

(ii) Anyone cannot forge the legal proxy signature.

(iii) The verifier can identify the actual proxy signer of its group.

(iv) It also provides more refined result against its time complexity, space complexity, and communication overhead.

(v) The proposed scheme is secure and efficient against notorious conspiracy attacks.

7. Conclusion

As the proxy signature is the solution to the delegation of signing capabilities in any electronic environment, in this

paper, various threshold proxy signature schemes have been compared based on whether they fulfill the proxy signature requirements or not and proposed an enhanced secure threshold proxy signature scheme. Some of these schemes are based on an RSA cryptosystem for known signers, as the RSA cryptosystem is a popular security technique. In this, we also propose a new scheme which includes the features and benefits of the two schemes: Fengying et al. and Geng et al. The implementation of the encryption and decryption functions justifies the real-life applicability of the proposed scheme. In this, we have analyzed our enhanced threshold proxy signature scheme for various parameters**.** In essence, the results show that the enhanced threshold proxy signature scheme is an efficient one and it can provide great capabilities for various applications.

Acknowledgment

The authors wish to thank many anonymous referees for their suggestions to improve this paper.

References

[1] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the Association for Computing Machinery*, vol. 21, no. 2, pp. 120–126, 1978.

[2] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[3] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," in *Proceedings of the Advances in Cryptology (Crypto'89)*, pp. 307–315, 1989.

[4] K. Zhang, "Threshold Proxy Signature Schemes," in *Proceedings of the Information Security Workshop*, pp. 191–197, 1997.

[5] S. Kim, S. Park, and D. Won, "Proxy signatures, revisited," in *Proceedings of the 1st International Conference on Information and Communication Security (ICICS'97)*, vol. 1334 of *Lecture Notes in Computer Science*, pp. 223–232, 1997.

[6] H. M. Sun, N. Y. Lee, and T. Hwang, "Threshold proxy signatures," *IEE Proceedings: Computers and Digital Techniques*, vol. 146, no. 5, pp. 259–263, 1999.

[7] M. S. Hwang, E. J. L. Lu, and I. C. Lin, "A practical (t, n) threshold proxy signature scheme based on the RSA cryptosystem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 6, pp. 1552–1560, 2003.

[8] G. Wang, F. Bao, J. Zhou, and R. H. Deng, "Comments on 'A practical (t, n) threshold proxy signature scheme based on the RSA cryptosystem'" *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 10, pp. 1309–1311, 2004.

[9] W. C. Kuo and M. Y. Chen, "A modified (t, n) threshold proxy signature scheme based on the RSA cryptosystem," in *Proceedings of the 3rd International Conference on Information Technology and Applications (ICITA'05)*, vol. 2, pp. 576–579, July 2005.

[10] F. Li, Q. Xue, and Z. Cao, "Crypanalysis of Kuo and Chen's threshold proxy signature scheme based on the RSA," in *Proceedings of the 4th International Conference on Information Technology-New Generations (ITNG'07)*, pp. 815–818, Las Vegas, Nev, USA, April 2007.

[11] G. Yong-Jun, T. Hui, and H. Fan, "A modified and practical threshold proxy signature scheme based on RSA," in *Proceedings of the 9th International Conference on Advanced Communication Technology (ICACT'07)*, pp. 1958–1960, Gangwon-Do, South Korea, February 2007.

[12] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures: delegation of the power to sign messages," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 79, no. 9, pp. 1338–1353, 1996.

[13] H. M. Sun, "Efficient nonrepudiable threshold proxy signature scheme with known signers," *Computer Communications*, vol. 22, no. 8, pp. 717–722, 1999.

[14] T. Okamoto, T. Mitsuru, and E. Okamoto, *Extended Proxy Signature For Smart Cards*, Lecture Notes in Computer Science, Springer, New York, NY, USA, 1999.

[15] C. C. Lee, T. C. Lin, S. F. Tzeng, and M. S. Hwang, "Generalization of proxy signature based on factorization," *International Journal of Innovative Computing, Information and Control*, vol. 7, no. 3, pp. 1039–1054, 2011.

[16] S. F. Tzeng, C. C. Lee, and M. S. Hwang, "A batch verification for multiple proxy signature," *Parallel Processing Letters*, vol. 21, no. 1, pp. 77–84, 2011.

[17] M. S. Hwang, S. F. Tzeng, and S. F. Chiou, "A non-repudiable multi-proxy multi-signature scheme," *Innovative Computing, Information and Control Express Letters*, vol. 3, no. 3, pp. 259–264, 2009.

[18] E. J. L. Lu, M. S. Hwang, and C. J. Huang, "A new proxy signature scheme with revocation," *Applied Mathematics and Computation*, vol. 161, no. 3, pp. 799–806, 2005.

[19] C. Y. Yang, S. F. Tzeng, and M. S. Hwang, "On the efficiency of nonrepudiable threshold proxy signature scheme with known signers," *Journal of Systems and Software*, vol. 73, no. 3, pp. 507–514, 2004.

[20] S. F. Tzeng, C. Y. Yang, and M. S. Hwang, "A nonrepudiable threshold multi-proxy multi-signature scheme with shared verification," *Future Generation Computer Systems*, vol. 20, no. 5, pp. 887–893, 2004.

[21] S. F. Tzeng, M. S. Hwang, and C. Y. Yang, "An improvement of nonrepudiable threshold proxy signature scheme with known signers," *Computers and Security*, vol. 23, no. 2, pp. 174–178, 2004.

[22] M. S. Hwang, S. F. Tzeng, and C. S. Tsai, "Generalization of proxy signature based on elliptic curves," *Computer Standards and Interfaces*, vol. 26, no. 2, pp. 73–84, 2004.

[23] C. S. Tsai, S. F. Tzeng, and M. S. Hwang, "Improved nonrepudiable threshold proxy signature scheme with known signers," *Informatica*, vol. 14, no. 3, pp. 393–402, 2003.

[24] L. H. Li, S. F. Tzeng, and M. S. Hwang, "Generalization of proxy signature-based on discrete logarithms," *Computers and Security*, vol. 22, no. 3, pp. 245–255, 2003.

[25] M. S. Hwang, C. C. Lee, and S. J. Hwang, "Cryptanalysis of the Hwang-Shi proxy signature scheme," *Fundamenta Informaticae*, vol. 53, no. 2, pp. 131–134, 2002.

[26] M. S. Hwang, I. C. Lin, and E. J. L. Lu, "A secure nonrepudiable threshold proxy signature scheme with known signers," *Informatica*, vol. 11, no. 2, pp. 137–144, 2000.

[27] R. Kumar and H. K. Verma, "Secure threshold proxy signature scheme based on RSA for known signers," *Journal of Information Assurance and Security*, vol. 5, no. 4, pp. 319–326, 2010.

[28] S. Katzenbeisser, *Recent Advances in RSA Cryptography*, Springer, New York, NY, USA, 2001.

[29] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures for delegating signing operation," in *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, pp. 48–56, March 1996.

[30] N. Y. Lee, T. Hwang, C. H. Wang, and O. Zhang, "Nonrepudiable proxy signature schemes," in *Proceedings of the Australasian Conference on Information Security and Privacy (ACISP'8)*, pp. 415–422, 1998.

[31] M. Agrawal, N. Kayal, and N. Saxena, "PRIMES is in P," *Annals of Mathematics*, vol. 160, no. 2, pp. 781–793, 2004.

[32] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Section 31.8: primality testing," in *Introduction to Algorithms*, pp. 889–890, MIT Press, McGraw-Hill, Cambridge, Mass, USA, 2nd edition, 2001.

[33] C. T. Li, *Multimedia Foresics and Security*, IGI Global, 1st edition, 2008.