Key Management Using Certificateless Public Key Cryptography in Ad Hoc Networks

Fagen Li[1,2,3], Masaaki Shirase[1], and Tsuyoshi Takagi[1]

[1] School of Systems Information Science,
Future University-Hakodate, Hakodate 041-8655, Japan
[2] School of Computer Science and Engineering,
University of Electronic Science and Technology of China, Chengdu 610054, China
[3] Key Laboratory of Computer Networks and Information Security,
Xidian University, Xi'an 710071, China
fagenli@uestc.edu.cn

Abstract. As various applications of wireless ad hoc network have been proposed, security has become one of the big research challenges and is receiving increasing attention. In this paper, we propose a distributed key management approach by using the recently developed concepts of certificateless public key cryptography and threshold secret sharing schemes. Without any assumption of prefixed trust relationship between nodes, the ad hoc network works in a self-organizing way to provide the key generation and key management services using threshold secret sharing schemes, which effectively solves the problem of single point of failure. Certificateless public key cryptography is applied here not only to eliminate the need for certificates, but also to retain the desirable properties of identity-based key management approaches without the inherent key escrow problem.

Keywords: Ad hoc network, network security, key management, certificateless public key cryptography.

1 Introduction

An ad hoc network is a collection of autonomous nodes that communicate with each other by forming a multi-hop wireless network. The property of not relying on the support from any fixed infrastructure makes it useful for a wide range of applications, such as instant consultation between mobile users in the battlefields, emergency, and disaster situations, where geographical or terrestrial constraints demand totally distributed networks. While ad hoc network provides a great flexibility for establishing communications, it also brings a lot of research challenges. One of the important issues is the security due to all the characteristics of these networks, such as the vulnerability of the wireless links, the limited physical protection of each node and the dynamically changing topology. Key management service is a crucial security issue because it is the essential assumption of many other security services. For instance, many secure routing protocols,

such as ARAN [1] and SRP [2], assume that a pair of private and public keys and a certificate signed by a trusted third party have been assigned to nodes.

Because ad hoc networks are highly vulnerable to various security threats due to its inherent characteristics, such as open medium, absence of fixed central structure, dynamically changing topology and constrained resource, traditional key management approaches based on public key infrastructure (PKI) is not directly applicable to ad hoc networks. Designing an efficient key management solution should satisfy following characteristics:

Lightweight: Solutions must minimize the amount of computation and communication required to ensure the security services to accommodate the limited energy and computational resources of nodes.

Decentralized: Like ad hoc networks themselves, attempts to secure them must be ad hoc way: they must establish security without a priori knowledge to centralized or persistent entities. Instead, security solutions must utilize the cooperation of all trustworthy nodes in the network.

Reactive: Ad hoc networks are dynamic: nodes may enter and leave the network spontaneously and unannounced. Security solutions must react to changes in network state; they must seek to detect compromises and vulnerabilities.

Fault-Tolerant: Wireless transfer mediums are known to be unreliable; nodes are likely to leave or be compromised without warning. The security solutions should be designed with such faults in mind; they must not rely on message delivery or ordering.

Current research works in key management are mainly based on traditional PKI [3–6] and identity-based public key cryptography (ID-PKC) [7–9]. These approaches based on traditional PKI use a partially distributed or a fully distributed certificate authority (CA) to issue and manage public key certificates. However, the resource-constrained ad hoc networks might be unable to afford the rather complicated certificate management, including revocation, storage and distribution, and the computational costs of certificate verification. ID-PKC get rid of the public key certificates by allowing allowing the user's public key to be any binary string, such as an email address, IP address that can identify the user. ID-PKC have an advantage in the aspect of the key management compared with the traditional PKI. However, ID-PKC needs a trusted private key generator (PKG) which generates the private keys of the entities using their public keys and a master secret key. Therefore, the dependence on the PKG who know all user's private keys inevitably causes the key escrow problem to the ID-PKC systems. For example, the PKG can decrypt any ciphertext in an identity-based public key encryption scheme. Equally problematical, the PKG could forge any entity's signatures in an identity-based signature scheme.

In this paper, we propose a novel key management approach using certificateless public key cryptography (CL-PKC) [10]. The CL-PKC does not require the use of certificates and yet does not have the built-in key escrow feature of ID-PKC. It is a model for the use of public key cryptography that is intermediate between traditional PKI and ID-PKC. A CL-PKC system still makes use of a trusted third party which is called the key generating center (KGC). By way of

contrast to the PKG in ID-PKC, the KGC does not have access to the user's private key. Instead, the KGC supplies a user with a partial private key that the KGC computes from the user's identity and a master key. The user then combines the partial private key with some secret information to generate the actual private key. The system is not identity-based, because the public key is no longer computable from a user identity. When Alice wants to send a message to Bob in a CL-PKC system, she must obtain Bob's public key. However, no authentication of Bob's public key is necessary and no certificate is required.

The rest of this paper is organized as follows. In Section 2, we study the related work in the literature. Some preliminary works are given in Section 3. Our proposed key management approach is detailed described in Section 4. Finally, the conclusions are given in Section 5.

2 Related Work

In [3], Zhou and Haas focused on how to establish a secure key management service in an ad hoc networking environment. They proposed to apply the secret sharing technique [11] to distribute the CA's private key among a pre-selected subset of nodes, called servers. Then any combination of t servers can jointly issue public key certificates to mobile nodes. The focus of their work is to maximize the security of the shared secret in the presence of possible compromises of the secret share holders. It assumes a small group of servers with rich connectivity. Therefore, it is not suitable for purely ad hoc environments. [4] and [5] make an extension of [3] and provide a fully distributed CA scheme. In other words, each node holds a secret share, and k or more nodes in a local neighborhood jointly provide complete services. This solution has a good availability since all nodes are part of the CA service, it is easier for a node to locate k neighbor nodes and request the CA service. In [6], Hubaux et al. proposed a self-organized certificate chaining key management approach, which has similarity with PGP "web of trust" concept. Unlike the above publications, it does not require a trusted authority or any special nodes; instead, each node issues its own certificates to other nodes. Key authentication is performed via chains of certificates. Certificate chaining fits naturally with ad hoc networks where there is no physical infrastructure, relying on each mobile node to issue certificates to other nodes at their own discretion. However, certificate chaining requires a warm-up period to populate the certification graph, which completely depends on the individual node's behavior and mobility. Additionally, the validity of a certificate chain depends on the trustworthiness of all the mobile nodes in the chain, which may not be easy to ensure in open networks.

In [7], Khalili et al. provide a key distribution mechanism combining the use of ID-PKC and threshold cryptography. Their scheme avoids the need for users to generate their own public keys and distribute these keys throughout the network, since the user's identity acts as her public key. Besides that, users only need to propagate their identities instead of the certificates. This can lead to huge savings in bandwidth. However, the usage of ID-PKC instead of cer-

tificates also results in a few weaknesses. One major weakness is that the key escrow problem since distributed PKG know all user's private keys. The compromise of the PKG's master key could be disastrous in an ID-PKC system, and usually more severe than the compromise of a CA's signing key in a traditional public key cryptography. For these reasons, it seems that the use of ID-PKC may be restricted to small, closed groups or to applications with limited security requirements.

3 Preliminaries

3.1 Certificateless Public Key Cryptography

The idea of CL-PKC is proposed by Al-Riyami and Peterson [10] with the original motivation of eliminating the inherent key escrow problem of ID-PKC. Since then, different encryption and signature schemes were suggested [12–14]. In CL-PKC, the KGC supplies an user with a partial secret key which the KGC computes from the user's identity and a master key, and then the user combines its partial secret key and the KGC's public parameters with some secret information to generate its actual secret key and public key respectively. In this way, an user's secret key is not available to the KGC.

In more detail, an certificateless public key encryption scheme consists of the following algorithms:

- `Setup:` This algorithm takes security parameter k and returns the system parameters *params* and *master-key*. The system parameters includes a description of the message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. Usually, this algorithm is run by the KGC. The KGC publishes system parameters *params* and keeps the *master-key* secret.
- `Partial-Private-Key-Extract:` This algorithm takes *params*, *master-key* and an identity for entity A, $ID_A \in \{0,1\}^*$, as input. It returns a partial private key D_A. Usually this algorithm is run by the KGC and its output is transported to entity A over a confidential and authentic channel.
- `Set-Secret-Value:` This algorithm takes as inputs *params* and an entity A's identity ID_A as inputs and outputs A's secret value x_A.
- `Set-Private-Key:` This algorithm takes *params*, an entity A's partial private key D_A and A's secret value x_A as input. The value x_A is used to transform D_A into the (full) private key S_A. The algorithm returns S_A.
- `Set-Public-Key:` This algorithm takes *params* and entity A's secret value x_A as input and from these constructs the public key P_A for entity A.
- `Encrypt:` This algorithm takes as inputs *params*, a message $m \in \mathcal{M}$, and the public key P_A and identity ID_A of an entity A. It returns either a ciphertext $c \in \mathcal{C}$ or the null symbol $\perp$ indicating an encryption failure.
- `Decrypt:` This algorithm takes as inputs *params*, $c \in \mathcal{C}$, and a private key S_A. It returns a message $m \in \mathcal{M}$ or a message $\perp$ indicating a decryption failure.

3.2 Threshold Secret Sharing

Secret sharing allows a secret to be shared among a group of users (also called shareholders) in such a way that no single user can deduce the secret from his share alone. To construct the secret, one needs to combine a sufficient number of shares. (k, n) threshold secret sharing represents that the secret is distributed to n shareholders, and any k or more users can reconstruct the secret from their shares, but $k-1$ or fewer users cannot get any information about the secret. Here, k is the threshold parameter such that $1 \leq k \leq n$. The first threshold secret sharing scheme was proposed by Shamir [11] in 1979, which is based on polynomial interpolation. To distribute a secret S among n users, a trusted authority chooses a large prime q, and randomly selects a polynomial

$$f(x) = S + a_1 x + \cdots + a_{k-1} x^{k-1} (\text{mod } q),$$

where $a_1, \ldots a_{k-1} \in Z_q$. The trusted authority computes each user's share by $S_i = f(i)$ and securely sends the share S_i to user i. Then any k users can reconstruct the secret by computing

$$S = \sum_{i=1}^{k} S_i L_i (\text{mod } q),$$

where

$$L_i = \prod_{j=1, j \neq i}^{k} -j/(i-j) (\text{mod } q).$$

There are two weaknesses in the Shamir secret sharing scheme. On the one hand, his scheme does not detect the trusted authority distributes erroneous shares to some users and does not detect some compromised users provide false shares; on the other hand, his scheme needs a trusted authority to distribute a secret to users. To detect incorrect shares, a few verifiable secret sharing (VSS) schemes was proposed in [15–17]. A VSS scheme generates extra public information for each share using a one-way function. The public information can testify the correctness of the corresponding shares without disclosing them. To solve the second weakness of the Shamir secret sharing scheme, Pedersen [19] proposed a secret sharing scheme without having a trusted authority, which selects the secret and distributes it to users. In stead, these users choose the secret and distribute it among themselves.

In the secret sharing schemes described above, the secret is protected by distributing it among several users. However, given sufficiently long time an attacker could compromise k users and obtain their shares, thereby allowing him to reconstruct the secret. To defend against such attackers, proactive secret sharing schemes [18] use share refreshing, which enables users to compute new shares from old ones in collaboration without disclosing the shared secret to any user. The new shares constitute a new (k, n) sharing of the secret. After refreshing, users remove the old shares and only keep the new ones. Because the new shares are independent of the old ones, the adversary cannot combine old

shares with new shares to recover the secret. Thus, the attacker is challenged to compromise k users between periodic refreshing.

4 Proposed Security Solution

In this section, we first describe our assumptions about the network, and then give an overview of our key management approach using the threshold secret sharing schemes and certificateless public key cryptography. Finally, we describe our approach in detail.

4.1 Assumptions

Our key management approach does not rely on any assumption of underlying key management subsystem. That is, there is no trusted authority to generate and distribute the public/private keys and there is no pre-built trust association between nodes in the network. All the keys used are generated and maintained in a self-organizing way within the network.

We assume that each mobile node carries an IP address or an identity, which is unique and unchanged during its lifetime in the ad hoc network. The IP address or identity can be obtained through some dynamic address allocation and auto-configuration, only if the address is selected without any conflict with other nodes in the network. We also assume that each mobile node has a mechanism to discover its one-hop neighborhood and to get the identities of other nodes in the network.

4.2 Proposed Security Scheme

4.2.1 Overview

Consider that an ad hoc network has n nodes in the initial phase. The network has a public/private key pair, called master key $\langle PK, SK \rangle$, which is used to provide key generation service to all the nodes in the network. The master key pair is generated in such a manner that the master public key PK is well known to all the nodes in the network, and the master private key SK is shared by all the nodes in a (k, n) threshold fashion. Any k or more nodes can reconstruct the master private key SK from their shares, but $k-1$ or fewer nodes cannot get any information about the SK. Before utilizing any network service, each node will have to obtain its partial private key corresponding to its identity and distribute its public key throughout the network. This partial private key can be computed by obtaining k shares of its key from the original nodes in the network. Note that the distributed key generation service in a (k, n) threshold fashion requires an adversary to corrupt at least k nodes in order to obtain a user's partial private key. Furthermore, honest nodes need only contact any k nodes in order to obtain their own partial private keys, thus making the protocol resilient to temporary loss of connectivity with other nodes in the network.

Our solution has the following good characteristics: (i) It does not need a trusted authority to select and to distribute the master private key to nodes. Nodes choose the secret and distributes it among themselves. (ii) It does not need public key certificates, saving network bandwidth and computational power of nodes. (iii) The using of the CL-PKC make our solution eliminate the key escrow problem of the ID-PKC key management approaches [7–9].

In the following, we describe the basic operations of our key management approach: master public/private key generation, partial private key generation service, key agreement, new master private key share creation, and master private key share refreshing of nodes.

4.2.2 Master Key Generation

Our master key generation mechanism uses the Pedersen's threshold secret sharing scheme without a trusted authority [19]. Therefore, our approach does not need the support of the trusted authority to compute a master private key, separate it into multiple shares and then distribute the shares to shareholders. Instead, the master key pair is computed collaboratively by the initial network nodes. The detailed scheme is as follows.

1. Each node C_i randomly chooses a secret x_i and a polynomial $f_i(x)$ over Z_q of degree $k-1$ such that $f_i(0) = x_i$. Let

$$f_i(x) = a_{i0} + a_{i1}x + \cdots + a_{i,k-1}x^{k-1},$$

   where $a_{i0} = x_i$.
2. Each node C_i computes $w_{ij} = g^{a_{ij}}$ for $j = 0, \ldots, k-1$ and broadcasts $\{w_{ij}\}_{j=0,\ldots,k-1}$.
3. When everybody has sent these $k-1$ values, C_i sends $s_{ij} = f_i(j)$ securely to C_j for $j = 1, \ldots, n$ (in particular C_i keeps s_{ii}).
4. C_i verifies the correctness of s_{ji} from C_j by checking

$$g^{s_{ji}} = \prod_{l=0}^{k-1} w_{jl}^{i^l}.$$

   If this fails, C_i broadcasts that an error has been found, publishes s_{ji} and then stops.
5. C_i can compute its share of master private key as $S_i = \sum_{j=1}^{n} s_{ji}$. That is, the master private key share of node C_i is combined by the subshares from all the nodes, and each of them contributes one piece of that information.
6. Any coalition of k shareholders can jointly recover the secret as in basic secret sharing by computing $\sum_{i=1}^{k} S_i L_i (\text{mod } q)$, where

$$L_i = \prod_{l=1, l\neq i}^{k} -l/(i-l) (\text{mod } q).$$

It is easy to see that the jointly generated master private key

$$SK = \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} f_i(0).$$

Then, the master public key can be computed as

$$PK = SKP = \sum_{i=1}^{k} S_i L_i P,$$

where P is a common parameter used by the certificateless encryption scheme [10].

4.2.3 Distributed Partial Private Key Generation Service

Suppose that an entity A with identity ID_A needs to obtain its public key and corresponding private key. A choose a secret value x_A and set its public key as $P_A = \langle X_A, Y_A \rangle$, where $X_A = x_A P$ and $Y_A = x_A PK$. Then, A make its public key P_A is well known to all the nodes in the network. To obtain the private key, A contacts at least k neighbor nodes, present the identity and request partial private key generation service. These nodes that hold the master private key share can be the KGC service nodes. In our scheme, all the network nodes share the master private key, thus each of them can be the KGC service node. Each of the k KGC service nodes generates a secret share of the partial private key D_A and sends to A. To make sure the generated shares are securely transmitted, each of the KGC service nodes sends encrypted share to the node A using A's public key P_A. The process of generation of a share of the partial private key D_A can be represented by $D_{A_i} = S_i H_1(ID_A)$, where $S_i (i = 1, \ldots, k)$ is the share of the master private key of the KGC node, H_1 is a hash function used by the certificateless encryption scheme [10], and D_{A_i} is the generated partial private key share for the node A. A can verify the correctness of D_{A_i} by checking

$$\hat{e}(D_{A_i}, P) = \hat{e}(H_1(ID_A), W_i),$$

where $\hat{e}$ is a bilinear map defined in [10] and $W_i = S_i P$ is the i-th KGC's share commitments. If this fails, A broadcasts that an error has been found, publishes D_{A_i} and then stops. After obtaining k valid partial private key share, A calculate the complete partial private key as $D_A = D_{A_i} L_i$, where

$$L_i = \prod_{l=1, l \neq i}^{k} -l/(i-l) (\text{mod } q).$$

Then, A can sets its (full) private key $S_A = x_A D_A$.

4.2.4 Key Agreement

Suppose that entity A has its (full) private key S_A and corresponding public key $P_A = \langle X_A, Y_A \rangle$. Entity B has its (full) private key S_B and corresponding public

key $P_B = \langle X_B, Y_B \rangle$. If they want to set up a session key, A chooses a random values $a \in Z_q^*$ and sends $T_A = aP$ to B. B chooses a random values $b \in Z_q^*$ and sends $T_B = bP$ to A. After the above messages are exchanged, both entities check the validity of each other's public keys (A checks $\hat{e}(X_B, PK) = \hat{e}(Y_B, P)$ and B checks $\hat{e}(X_A, PK) = \hat{e}(Y_A, P)$). Then A computes

$$K_A = \hat{e}(H_1(ID_B), Y_B)^a \hat{e}(S_A, T_B)$$

and B computes

$$K_B = \hat{e}(H_1(ID_A), Y_A)^b \hat{e}(S_B, T_A).$$

It is easy to see that $K = K_A = K_B$ is a shared session key between A and B.

4.2.5 New Master Private Key Share Creation

When a new node C_p joins a network, it presents its identity, public key, and some other required physical proof to k neighbor nodes and requests the master public key and his share of the master private key. Each node in the coalition verifies the validity of the identity of the new node C_p. If the verification succeeds, the C_p's private key can be generated using the above method. To initialize the share of master private key for the requesting node, each coalition node C_i generates the partial share $s_{ip} = S_i L_i(p)$ for node C_p. Here, $L_i(p)$ is the Lagrange coefficient. It encrypts the partial share using C_p's public key and sends it to C_p. Node C_p obtains its new share by adding the partial shares as

$$S_p = \sum_{j=1}^{k} s_{p,j}.$$

Note that the partial shares may be shuffled before being sent to the joining node to protect the secrecy of the coalition nodes' secret shares [4]. After obtaining the share of the master private key, the new joining node is available to provide KGC service to other joining nodes.

4.2.6 Master Private Key Share Refreshing of Nodes

To protect against attackers that might compromise k or more nodes if there is enough time, a proactive secret sharing scheme is used to enable nodes of a region to compute new shares from old ones in collaboration without disclosing the master private key of the region. It relies on the homomorphic property. We notice that it is unnecessary to require all the nodes involved in the master private key share refreshing process. Instead, the task can be done by only k nodes, since we assume that, between any consecutive secret share updates, the number of adversaries who hold secret shares originated from the same secret key is less than k. To detect those incorrect subshares, the VSS scheme [15–17] is employed.

Details are shown as follows. To renew the master private key shares of all the n nodes in a region, k nodes are chosen from this region. Each node $C_i (1 \leq i \leq k)$ randomly generates $(S_{i1}, S_{i2}, \ldots, S_{in})$, a (k, n) sharing of 0. Then,

every subshares $S_{ij}(1 \leq j \leq n)$ is distributed to node C_j. When node C_j gets the subshares $S_{1j}, S_{2j}, \ldots, S_{kj}$, it can compute a new share from these subshares and its old share ($S'_j = S_j + \sum_{i=1}^{k} S_{ij}$). The new shares constitute a new (k, n) sharing of the master private key. After refreshing, nodes remove the old shares and use the new ones to provide the partial private key generation service. Because the new shares are independent of the old ones, the adversary cannot combine old shares with new shares to recover the master private key. Thus, the adversary is challenged to compromise k nodes in the same region between periodic refreshing.

5 Conclusions

Ad hoc networks are new paradigm in networking technologies. Key management is one of the most crucial technologies for security of ad hoc networks. This paper presents a new approach for key management using certificateless public key cryptography and threshold secret sharing schemes. Certificateless public key cryptography is applied here not only to eliminate the need for certificates, but also to retain the desirable properties of identity-based key management approaches without the inherent key escrow problem. In addition, we completely avoid a centralized certification authority or trusted third party to distribute the public keys and the certificates, thus enhance the tolerance of the network to compromised nodes and also efficiently save network bandwidth.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported by the National Natural Science Foundation of China (60673075), the National High Technology Research and Development Program of China (2006AA01Z428), the Key Laboratory of Computer Networks and Information Security of Xidian University (2008CNIS-02), and Youth Science and Technology Foundation of UESTC.

References

1. K. Sanzgiri, B. Dahill, B.N. Levine, C. Shields, and E.M. Belding-Royer. A secure routing protocol for ad hoc networks. In *Proceedings of 10th IEEE International Conference on Network Protocols*, Paris, France, pp. 78–87, 2002.
2. P. Papadimitratos and Z.J. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, San Antonio, TX, January 27–31, 2002.
3. L. Zhou and Z.J. Haas. Securing ad hoc networks. *IEEE Network*, Vol. 13, No. 6, pp. 24–30, 1999.
4. J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing robust and ubiquitous security support for mobile ad hoc networks. In *Proceedings of 2001 International Conference on Network Protocols*, Riverside, USA, pp. 251–260, 2001.

5. H. Luo, J. Kong, P. Zerfos, S. Lu, and L. Zhang. Self-securing ad hoc wireless networks. In *Proceedings of Seventh IEEE Symposium on Computers and Communications*, Taormina-Giardini Naxos, Italy, pp. 567–574, 2002.
6. J.P. Hubaux, L. Buttyan, and S. Capkun. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, Vol. 2, No. 1, pp. 52–64, 2003.
7. A. Khalili, J. Katz, and W.A. Arbaugh. Toward secure key distribution in truly ad hoc networks. In *Proceedings of 2003 Symposium on Applications and the Internet Workshops*, Orlando, FL, USA, pp. 342–364, 2003.
8. H. Deng, A. Mukherjee, and D. Agrawal. Threshold and identity-based key management and authentication for wireless ad hoc networks. In *Proceedings of International Conference on Information Technology: Coding and Computing*, Las Vegas, NV, USA, pp. 107–111, 2004.
9. H. Deng and D. Agrawal. TIDS: threshold and identity-based security scheme for wireless ad hoc networks. *Ad Hoc Networks*, Vol. 2, No. 3, pp. 291–307, 2004.
10. S.S. Al-Riyami and K.G. Peterson. Certificateless public key cryptography. In *Advances in Cryptology-ASIACRYPT 2003*, Lecture Notes in Computer Science, Vol. 2894, Springer-Verlag, Berlin, pp. 452–474, 2003.
11. A. Shamir. How to Share a Secret. *Communications of the ACM*, Vol. 22, No. 11, pp. 612–613, 1979.
12. D.H. Yum and P.J. Lee. Generic construction of certificateless encryption. In *Computational Science and Its Applications-ICCSA 2004*, Lecture Notes in Computer Science, Vol. 3043, Springer-Verlag, Berlin, pp. 802–811, 2004.
13. D.H. Yum and P.J. Lee. Generic construction of certificateless signature. In *Information Security and Privacy-ACISP 2004*, Lecture Notes in Computer Science, Vol. 3108, Springer-Verlag, Berlin, pp. 200–211, 2004.
14. S.S. Al-Riyami and K.G. Peterson. CBE from CL-PKE: a generic construction and efficient schemes. In *Public Key Cryptography-PKC 2005*, Lecture Notes in Computer Science, Vol. 3386, Springer-Verlag, Berlin, pp. 398–415, 2005.
15. B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proceedings of 26th IEEE Symposium on Foundations of Computer Science*, Portland, OR, USA, pp. 151–160, 1985.
16. P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of 28th IEEE Symposium on Foundations of Computer Science*, Los Angeles, CA, USA, pp. 427–437, 1987.
17. T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology-CRYPTO'91*, Lecture Notes in Computer Science, Vol. 576, Springer-Verlag, Berlin, 1991, pp. 129–140, 1991.
18. A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: how to cope with perpetual leakage. In *Advances in Cryptology-CRYPTO'95*, Lecture Notes in Computer Science, Vol. 963, Springer-Verlag, Berlin, pp. 457–469, 1995.
19. T. Pedersen. A threshold cryptosystem without a trusted party. In *Advances in Cryptology-EUROCRYPT'91*, Lecture Notes in Computer Science, Vol. 547, Springer-Verlag, Berlin, pp. 522–526, 1991.