A PUBLIC KEY MANAGEMENT SCHEME AND THRESHOLD-MULTISIGNATURE SCHEME FOR MOBILE AD HOC NETWORKS

Dawoud S. Dawoud[*], Adronis Niyonkuru
[*] Dean of Engineering, Victoria University, Uganda
Dawoud.shenouda@vu.ac.ug

Abstract: Mobile ad hoc networks offer communication over a shared wireless channel without any pre-existing infrastructure. Threshold digital signatures are an important cryptographic tool used in most existing key management schemes for mobile ad hoc networks. This paper proposes a *threshold-multisignature* scheme designed specifically for mobile ad hoc networks. The signature scheme allows a subset of shareholders with threshold *t*, to sign an arbitrary message on behalf of the group. The group signature is publicly verifiable and allows any outsider to establish the identity of the individual signers. The paper proposes a *self-certified public key issuing protocol* that allows negotiation between a single entity and a *distributed* certificate authority for an implicit self-certified public key.

Key words: Mobile ad hoc networks, security, pairwise key management, public key cryptography, self-certified public keys, self-certificates, group-oriented cryptography, threshold-multisignatures.

1. INTRODUCTION

Mobile ad hoc networks by definition differentiate themselves from existing networks by the fact that they do not rely on a fixed infrastructure [1] [2]. An ad hoc network of wireless nodes is a temporarily formed, spontaneous or unplanned network. Since it is created, operated and managed by the nodes themselves, ad hoc networks are solely dependent upon the cooperative and trusting nature of nodes [3]. The self-organized and infrastructure-less characteristics of mobile ad hoc networks make the design of feasible security mechanisms a challenging quest. Ad hoc network security research initially focused on secure routing protocols. All routing schemes however, neglect the crucial task of secure key management and assume pre-existence and pre-sharing of secret and/or private/public key pairs [2]. This left key management considerations in the ad hoc network security field as an open research area.

Fully self-organized ad hoc networks can be informally visualized as a group of strangers, people who have never met before, coming together for a common purpose. Since the network users are strangers, thus having no past relationships, the following issues can be identified:

1) As these strangers have to get to know each other in order to establish trust between them, bootstrapping of trust is required in the security mechanisms that protect their interests.
2) Nodes share no *a priori* keying material or a common *offline* trusted third party (TTP). With conventional public key infrastructure the offline TTP is responsible for the distribution of keying material (certificates) *prior* to network formation and can be seen as the root of trust or trust anchor [4]. Eliminating the need for an *offline* TTP or any *a priori* sharing of keying material, without making a security tradeoff, is the major challenge in providing key management services for ad hoc networks.
3) The identities of the nodes prior to network formation are unknown and therefore may be considered to be random or chosen by the nodes themselves as they join the network.

In ad hoc networks, to avoid a single point of vulnerability, the TTP has to be distributed [2]. The system secret is thus divided up into shares and securely stored by the entities forming the distributed cryptosystem. The main advantage of a distributed cryptosystem is that the secret is never computed, reconstructed, or stored in a single location, making the secret more difficult to compromise, which effectively enhances security [2] [5].
The principles of (*n, t*) threshold signature schemes and multisignature schemes are *combined* in schemes referred to as *threshold-multisignature* schemes [6] [7] or threshold signature schemes with *traceability* [8] [9] [10]. The threshold-multisignature schemes are based on variants of the *generalized ElGamal signatures* [11] [12]] extended to a *multiparty* setting. Threshold signature schemes with traceability guarantee the signature verifier that at least *t* members participated in the group signature and allow the signature verifier to establish the identities of the signers. Since individual signers are traceable they can be held accountable for their signatures.

An important component of any threshold digital signature scheme is the sharing of the group key [13]. A threshold-multisignature scheme that is suitable for ad hoc networks would require a *distributed group key generation* protocol that effectively eliminates the need for a TTP acting as a key distribution center. The first distributed key generation protocol, for discrete-log based threshold cryptosystems, was introduced in [14] and later extended in [5] [15] [16].

In threshold cryptosystems it is impractical to assume that an adversary cannot compromise more than *t* shareholders during the entire lifetime of the distributed secret [2] [17]. The secret share therefore has to be periodically updated to allow only a limited period *T* in which an *active* and *mobile* adversary must compromise a sufficient percentage of the shares in order to break the system [17]. Also assuming the same shareholders to be present at all times is unrealistic and the availability/security tradeoff (set by the total group members *n* and threshold *t*) may need to be changed as a function of system vulnerability, networking environment and current functionality of the cryptosystem. This is particularly true in mobile ad hoc networks where a node may move out of transmission range, turn off due to depleted batteries or be compromised due to poor physical security. The access structure $\Gamma_P^{(n,t)}$ of the initial share distribution will thus not necessarily remain constant and will have to be redistributed to a new access structure $\Gamma_{P'}^{(n',t')}$ using a *secret redistribution protocol* [18] [19].

The main objective of this paper is to propose a key management scheme that is suitable for mobile ad hoc networks: the **Ad Hoc P**ublic **K**ey **M**anagement (AdHocPKM) based on a *threshold-multisignature* . The proposed scheme allows a subset of shareholders with threshold *t*, to sign an arbitrary message on behalf of the group. The group signature is publicly verifiable and allows any outsider to establish the identity of the individual signers. Furthermore, the paper proposes a *self-certified public key issuing protocol* that allows negotiation between a single entity and a *distributed* certificate authority for an implicit self-certified public key. The resulting AdHocPKM integrates the advantages of distributed key generation, threshold-multisignatures, self-certified public keying and self-certificates to yield a secure, trustworthy key management service with a high availability feature.

The paper is organized as follows: Section-2 a novel *threshold-multisignature* scheme suitable for ad hoc networks is presented. Comments on this proposal are given in Section-3. A proposed cryptographic key issuing protocol, *threshold self-certified public keying*, is introduced in Section-4.

The proposed key management protocol, *Ad Hoc Public Key Management* (AdHocPKM) and the complete model of the system is the subject of another paper. Some conclusions are provided in Section-5.

2. PROPOSED THRESHOLD MULTISIGNATURE SCHEME FOR MOBILE AD HOC NETWORKS

In this section a new threshold-multisignature scheme without a trusted third party (TTP) is proposed.

The scheme consists of the following parts (*2.1-2.7).*

2.1 System parameter setup and individual self-certificate generation

The group members P_i, for $i = 1: n$ agree on and publish the following system parameters:

p, q	Two large primes, such that $q \mid (p - 1)$.
g	Generator of the cyclic subgroup of order *q* in $(Z)_p^*$.
$H(\cdot)$	One-way hash function.
(*n, t*)	Threshold parameter *t* and total number of group members *n*.
T	Threshold cryptosystem secret update period.

In the case of no existing public key infrastructure (PKI) or any shared information between potential protocol participants, the following procedure is followed: Each group member P_i, for $i = 1: n$ chooses a set of two random numbers $(SK_i, id_i) \in_R Z_q$. P_i uses SK_i as its private key and computes the corresponding public key as $PK_i = g^{SK_i}$. P_i also uses id_i to generate its public identity $ID_i = g^{id_i}$. P_i verifiably encrypts id_i with its public key PK_i using a *public verifiable encryption scheme* [15] [16], to generate $E_{PK_i}(id_i)$.

The main idea behind *non-interactive verifiable encryption* is that any outsider can perform a *validity test* which takes a public value (ID_i) as input and allows a prover to verify that a cipher text ($E_{PK_i}(id_i)$) encrypts id_i under a public key (PK_i) such that (ID_i, id_i) $\in R$, where *R* is a binary relationship ($ID_i = g^{id_i}$) [16].

Each P_i generates a self-certificate binding its identity ID_i to its public key PK_i. The self-certificate may be of the following structure: $SelfCert_{P_i} = PS_{SK_i}(ID_i \parallel E_{PK_i}(id_i) \parallel PK_i \parallel IssueDate \parallel CertPeriod \parallel Ext)$. *CertPeriod* specifies the self-certificate validity period and *Ext*, optional additional extension information governed by the key issuing policy.

Each P_i broadcasts its self-certificate to all P_j for $j = 1: n, j \neq i$.

Forged certificates will result in two or more certificates with the same identity. Assume a network participant receives two certificates with identity ID_i. This constitutes proof that the identity was stolen by either one of the participants. By performing a *validity test* (which takes ID_i, PK_i and $E_{PK_i}(id_i)$ as inputs) any network participant can determine which one of the two certificates is authentic. If both validity tests yield a positive result then it can be concluded that the legitimate owner of ID_i has been compromised and should reboot.

In the case of an existing PKI: Protocol participants P_i, for $i = 1 : n$ are assumed to have a private key SK_i and an authentic certificate, verifiable with the public key of a distributed *online* trusted third party. The certificate binds the public key $PK_i = g^{SK_i}$ to the user's identity ID_i. The certificates are distributed to all communication entities P_j, for $j = 1: n, j \neq i$.

2.2 Initial publicly verifiable distributed group key generation

The *publicly verifiable* distributed group key generation scheme due to Zhang *et al* [16] is used to realize initial secret sharing of the group private key. The protocol is a secure, round optimal, *initial* secret sharing scheme to an access structure $\Gamma_P^{(n,t)}$ without a TTP. Note that the setup phase has been changed for the case of no existing PKI or any shared information between potential protocol participants. In this case the system parameter setup phase is as given in Section-2.1.

For each protocol participant P_i, the key generation scheme gives as output a share x_i of the group private key x_Q and the corresponding group public key $y_Q = g^{x_Q}$. P_i, $i \in Q$ is the subset of honest players after individual share verification. Each P_i can calculate its public key as $y_i = g^{x_i}$ and broadcasts the *verifiably encrypted* private key $E_{PK_i}(x_i)$ as commitment to y_i.

2.3 Individual signature generation

Any subset β of t or more members can represent the group and sign an arbitrary message m. Each member P_i, $i \in \beta$ selects a random integer $k_i \in [1, q-1]$ and computes $r_i = g^{k_i} \bmod p$. Each member *verifiably encrypts* k_i with its own public key PK_i using a *verifiable encryption scheme* [15] [16] to generate $E_{PK_i}(k_i)$. Each member broadcasts its r_i and cipher text $E_{PK_i}(k_i)$ to all other members. This implies that each member commits to its public value r_i and provides a knowledge proof of its corresponding discrete logarithm, k_i.

After all committed r_i's are available the value R is calculated:

$$R = \prod_{i \in \beta} r_i \bmod p \qquad (1)$$

Each P_i uses public values y_i and ID_i to form the sets (ID_i, y_i), for $i \in \beta$. P_i uses these sets to construct a Lagrange polynomial function $h(y)$ as follows [9] [10]:

$$h(y) = \sum_{i \in \beta} \left(ID_i \prod_{j \in \beta, j \neq i} \frac{y - y_j}{y_i - y_j} \right) \qquad (2)$$

P_i uses secret keys x_i and k_i to compute individual signature s_i on message m from the following equation:

$$s_i = H(m, R, h(y))\, k_i + x_i \cdot L_{\beta i} \bmod q \qquad (3)$$

Where the Lagrange interpolating coefficient $L_{\beta i}$ is given by:

$$L_{\beta i} = \prod_{k \in \beta, k \neq i} \frac{-ID_k}{ID_i - ID_k} \bmod q \qquad (4)$$

The set (s_i, r_i) is the individual signature of P_i on message m, which is broadcast to all other group members.

Equation-3, which is used to calculate the individual signatures, is based on a variant of the ElGamal signature scheme. It can be shown that this variant is secure against forgery and more efficient to compute than the original ElGamal digital signature [11].

2.4 Individual signature verification

Upon reception of all the signatures (s_i, r_i), P_j, $j \in \beta$, performs the functionality of a clerk and uses the public key y_i to authenticate the individual signature of P_i by verifying whether the following equation holds:

$$g^{s_i} = r_i^{H(m,R,h(y))} y_i^{L_{\beta i}} \bmod p \qquad (5)$$

If Equation-3 fails to hold, the individual signature for message m is invalid. Participants are disqualified if their individual signatures are found to be invalid. The remaining honest participants form the set α and repeat the *individual signature generation* part from Equation-1 in Section-2.3 with $\beta = \alpha$.

2.5 Threshold signature generation

After P_j, $j \in \alpha$ has received and verified t or more individual signatures the group signature on message m can be obtained as (R, S) computed as:

$$R = \prod_{i \in \alpha} r_i \bmod p \qquad (6)$$

$$S = \sum_{i \in \alpha} s_i \bmod q \qquad (7)$$

The function $h(y)$ is attached to (R, S) and can be used later to trace the signers who collaborated to generate the threshold signature (R, S) on message m.

2.6 Threshold signature verification

Any outsider can use the group public key y_Q to verify the validity of the group signature (R, S) for a message m by checking if the following equation holds:

$$g^S = R^{H(m,R,h(y))} y_Q \bmod p \quad (8)$$

If Equation-8 holds, the group signature (R, S) for message m is valid.
Proof:

$$s_i = H(m,R,h(y))k_i + x_i \cdot L_{\alpha i} \bmod q \ (i \in \alpha)$$

$$\sum_{i\in\alpha} s_i = \sum_{i\in\alpha} H(m,R,h(y))k_i + \sum_{i\in\alpha}(x_i \cdot L_{\alpha i}) \bmod q$$

$$S = \sum_{i\in\alpha} H(m,R,h(y))k_i + x_Q \bmod q$$

$$g^S = g^{\sum_{i\in\alpha} H(m,R,h(y))k_i} g^{x_Q} \bmod p$$

$$g^S = R^{H(m,R,h(y))} y_Q \bmod p$$

2.7 Signer identification

Any outsider can determine the signers of the threshold signature (R, S) on message m by using member P_i's public values (ID_i, y_i) and solve the following equation:

$$ID_i = h(y_i) \quad (9)$$

If Equation-8 and Equation-9 hold, then ID_i is a signer of the threshold signature (R, S) for a message m.

3. DISCUSSION AND COMMENTS ON THE PROPOSED THRESHOLD-MULTISIGNATURE SCHEME

The proposed threshold-multisignature scheme is based on a variant of the ElGamal signature scheme [22] proposed by Yen *et al* [23] extended to a *multiparty* setting. The scheme can equally use any other secure and efficient signature generating variant of the ElGamal signature scheme. References [11] [12]] help in selecting such a variant. This implies that if an attack is found in future on the variant presented by Yen *et al* [23] used in the proposed threshold signature scheme, the variant can be replaced with a secure one.

Since the proposed *threshold-multisignature* scheme is a *natural* extension of the Yen *et al* ElGamal variant to a group setting, the security analysis is equivalent to those in [11] [12]] [23] and other similar threshold signature schemes [6] [7] [9] [10]. The remainder of the section will only briefly consider some new attacks on similar schemes and special features of the proposed scheme.

In [24], Wu *et al* present a *universal forgery* attack on the threshold signature scheme with traceable signers presented by Li *et al* in [9]. The attack allows adversaries to generate valid group signatures without detection. In [24] it is also argued that signers in [9] are in fact untraceable since the Lagrange polynomial function $h(y)$ used to identify the signers is not bounded to the group signature (R, S) on message m. This also applies to the threshold signature scheme proposed by Wang *et al* [10]. A similar *framing* attack is reported by Wang *et al* in [39] on the threshold-multisignature signature scheme proposed by Li *et al* in [7].

The threshold-multisignature scheme proposed in this paper eliminates universal forgery or framing attacks by using a *publicly verifiable* distributed key generation (DKG) protocol [16] to construct the threshold cryptosystem and *public verifiable* encryption [15] [16] [21] to commit shareholders to their public values. Utilizing a DKG protocol also prevents adversaries from controlling the group secret key. Including the Lagrange polynomial function $h(y)$ within the *individual signatures* (Equation-2, Section-2.3) binds $h(y)$ to the threshold signature (R, S) on message m. Note that $h(y)$ in contrast to [9] [10] cannot be manipulated without detection. By evaluating $h(y)$ the identities of the signers are traceable by any outsider.

A centralized combiner node is a specialized server which can be seen as a single point of vulnerability and should be distributed or replicated in ad hoc networks to ensure a correct combination [2]. The scheme proposed in this paper eliminates the need for a designated combiner/clerk as the construction of the threshold group signature is done by the shareholders themselves. This eliminates attacks relying on a corrupt clerk and mitigates the problem of ensuring the availability of a combiner node [2]. The scheme preserves the symmetric relationship between the shareholders by placing on each node the same computational, memory and communication overhead.

4. PROPOSED THRESHOLD SELF-CERTIFIED PUBLIC KEY ISSUING PROTOCOL

In this section a self-certified public key generating protocol, *threshold self-certified public keying*, is proposed by modifying the scheme given in [20] to make it suitable for ad hoc networks. The modification allows negotiation for a self-certified public/private key pair between a *distributed* certificate authority (DCA) and a single entity (party A) in contrast to the *centralized* certificated authority used in [20]. The scheme also eliminates the need for a designated combiner node.

In the proposed protocol a DCA and party A agree on an *implicitly* self-certified, public key for party A, without the DCA learning the corresponding private key. Party A then signs the self-certified public key y_A and certificate information CIA to yield a self-certificate $SelfCert_A$ that provides y_A with *explicit* authentication.

The DCA is constructed as explained in Section-2.1 and Section-2.2.

The proposed protocol is as follows:

1) Party *A* chooses a random number $a \in_R Z_q$ and identity ID_A, computes $r_A = g^a$ and transmits a certification request (CertReq) containing (ID_A, r_A) to *n* servers forming the DCA.

2) The DCA servers who received the certification request each prepares certification information CI_A depending on the DCA's certification policy. For example, the DCA can use party *A*'s identity, DCA's identity, party *A*'s public parameter *rA*, certificate serial number, date of issue, DCA's public key and other relevant extension information to yield: CI_A = [ID_A || ID_{DCA} || r_A || *CertNo* || *IssueDate* || *CertPeriod* || y_{DCA} || *Ext*].

Each of the DCA servers computes the signature parameter, $s_A = h(CI_A)$ and collaborate to generate a *threshold-multisignature* (*R, S, h*(*y*)) on s_A as explained in Section-3. Each server P_i, $i \in \alpha$ sends (s_i, r_i), (*R, S, h*(*y*)), s_A and CI_A to party *A*, which securely store all information.

3) Party *A* verifies the threshold-multisignature, (*R, S, h*(*y*)) on signature parameter, s_A. Party *A* then computes the private key $x_A = s_A + a$. The tuple (r_A, x_A) can be seen as the signature of the DCA on the certification information CI_A. Party *A* then calculates its corresponding public key using Equation-10:

$$y_A \equiv g^{x_A} = g^{h(CI_A)} \cdot r_A \qquad (10)$$

4) Party *A* signs (CI_A, (*R, S, h*(*y*)), y_A) with its private key x_A to generate the self-certificate of y_A, with the following structure: $SelfCert_A = PS_{x_A}(CI_A$ || (*R, S, h*(*y*)) || y_A || *UserIssueDate* || *Ext*).

Party *A* publishes $SelfCert_A$, the signature of which can be verified using y_A. The public key y_A of *A*, can be publicly verified by checking that the signature (*R, S, h*(*y*)) on $h(CI_A)$ is correct and evaluating Equation-10. In the remainder of the text the proposed algorithm will be referred to as *threshold self-certified public keying*.

4.1 Self-organized key renewal

The key pair (x_A, y_A) is the basic public/private key pair obtained by party *A* via the self-certified public key issuing protocol as given above in Section-2. In the self-certificate generation protocol due to Lee *et al* [25] users can renew their public/private key pair and self-certificate to generate a renewed key pair (x'_A, y'_A) and a renewed self certificate $SelfCert'_A = PS_{x'_A}(CI_A$ || (*R, S, h*(*y*)) || y_A || y'_A || *UserIssueDate'* || *Ext'*). This can also be referred to as *self-organized* key renewal.

To make the *user-controlled* key renewal procedure as given in [25] compatible with the proposed self-certified public key issuing protocol for a distributed certificate authority given in Section-5, the verification equation must be modified to the following:

$$y'_A = g^{h(CI_A)h(CI_A, r'_A)} r_A^{h(CI_A, r'_A)} r'_A \qquad (11)$$

Discussion on self-organized key renewal: Important advantages of the key renewal scheme are given [25]:

- Users can renew their own self-certified key pairs (x'_A, y'_A) without contacting the distributed certificate authority.
- An adversary not knowing party *A*'s private key x_A cannot forge a renewed key pair (x'_A, y'_A).
- If the basic key pair (x_A, y_A) is not used for any encryption (real communication) and only for key renewal, it is not vulnerable to any *shortcut* public key encryption scheme attacks (for example the adaptive chosen message attack) [4].
- In the case of an adversary compromising x'_A it is not possible for the adversary to gain any knowledge of any subsequent renewed key pairs (x''_A, y''_A).

The simplistic key renewal protocol therefore not only enhances security, but empowers the users to control the frequency of their key renewal. Since users are able to update their keys whenever they suspect that it has been compromised or lost, the trust in key pairs and authenticity of certificates are enhanced.

5. CONCLUSION

This paper addresses the issue of key management in mobile ad hoc networks. The proposed scheme is only operated by the end-users and does not require any *offline* trusted third party (TTP) or *a priori* sharing of keying material.

This solves the major problem of many existing key management proposals for mobile ad hoc networks that use a *distributed certificate authority* (DCA). In the existing proposals the DCA needs to be empowered with a common *offline* TTP, which is not guaranteed to be available in ad hoc networks.

6. REFERENCES

[1] [Capkun et el, 2003] S. Capkun, L. Buttyan, and J.-P. Hubaux, "Self-Organized Public-Key Management for Mobile Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 2, no. 1, pp. 52–64, 2003.

[2] [Zhou & Hass, 1999] L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks," *IEEE Network:*

special issue on network security, vol. 13, no. 6, pp. 24–30, 1999.

[3] [Buttyan & Hubaux, 2003] L. Buttyan and J. P. Hubaux, "Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks," *ACM Mobile Networks and Applications*, vol. 8, no. 5, pp. 579–592, 2003.

[4] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook in Applied Cryptography*. CRC Press, 1996.

[5] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems," in *proc. Advances in Cryptology - EUROCRYPT'99*, May, 2-6 1999.

[6] C.-M. Li, T. Hwang, and N.-Y. Lee, "Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders," in *proc. Advances in Cryptology - EUROCRYPT'94*, May, 9-12 1994.

[7] C.-M. Li, T. Hwang, N.-Y. Lee, and J.-J. Tsai, "(t, n) Threshold-multisignature schemes and generalized- multisignature scheme where suspected forgery implies traceability of adversarial shareholders," *Cryptologia*, vol. 24, no. 3, pp. 250–268, 2000.

[8] W.-B. Lee and C.-C. Chang, "(t, n) Threshold Digital Signature with Traceability Property," *Journal of information Science and Engineering*, vol. 15, no. 5, pp. 669–678, 1999.

[9] Z.-C. Li, J.-M. Zhang, J. Luo, W. Song, and Y.-Q. Dai, "Group-oriented (t, n) threshold digital signature schemes with traceable signers," in *proc. Topics in Electronic Commerce, Second International Symposium, ISEC 2001*, April, 26-28 2001.

[10] C.-T. Wang, C.-H. Lin, and C.-C. Chang, "Threshold signature schemes with traceable signers in group communications," *Computer Communications*, vol. 21, no. 8, pp. 771–776, 1998.

[11] P. Horster, M. Michels, and H. Petersen, "Generalized ElGamal signatures for one message block," in *proc. 2nd Int. Workshop on IT-Security*, September, 22-23 1994.

[12] L. Harn and Y. Xu, "Design of generalized ElGamal type digital signature schemes based on discrete logarithms." *Electronics Letters*, vol. 30, no. 24, pp. 2025–2026, 1994.

[13] Y. Desmedt, "Threshold Cryptography," *European Trans. on Telecommunications*, vol. 5, no. 4, pp. 449–457, 1994.

[14] T. P. Pedersen, "A Threshold Cryptosystem without a Trusted Party," in *proc. Advances in Cryptology - EUROCRYPT'91*, 1991.

[15] M. Stadler, "Publicly Verifiable Secret Sharing," in *proc. Advances in Cryptology - EUROCRYPT'96*, 1996.

[16] R. Zhang and H. Imai, "Round Optimal Distributed Key Generation of Threshold Cryptosystem Based on Discrete Logarithm Problem," in *proc. Applied Cryptography and Network Security (ACNS'03)*, October, 16-19 2003.

[17] A. Herzberg, S. Jaracki, H. Krawczyk, and M. Yung, "Proactive Secret Sharing Or: How to Cope With Perpetual Leakage," in *proc. Advances in Cryptology - CRYPTO '95*, 1995.

[18] Y. Desmedt and S. Jajodia, "Redistributing Secret Shares to New Access Structures and Its Applications," Department of Information and Software Engineering, School of Information Technology and Engineering, George Mason University, Technical Report ISSE-TR-97-01, July 1997.

[19] T. M. Wong, C. Wang, and J. M. Wing, "Verifiable Secret Redistribution for Archive System," in *proc. First International IEEE Security in Storage Workshop*, December, 11 2002.

[20] H. Petersen and P. Horster, "Self-certified keys - Concepts and Application," in *proc. Third Conf. on Communication and Multimedia Security*, September 22-23 1997.

[21] J. Camenisch and I. Damgrd, "Verifiable Encryption, Group Encryption, and their Applications to Separable Group Signatures and Signature Sharing Schemes," in *proc. Advances in Cryptology - ASIACRYPT'00*, 2000.

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

[23] S.-M. Yen and C.-S. Laih, "New Digital Signature Scheme based on Discrete Logarithm," *Electronics Letters*, vol. 29, no. 12, pp. 1120–1121, 1993.

[24] T.-S. Wu and C.-L. Hsu, "Cryptanalysis of group-oriented (t, n) threshold digital signature schemes with traceable signers," *Computer Standards and Interfaces*, vol. 26, no. 5, pp. 477–481, 2004.

[25] B. Lee and K. Kim, "Self-certificate: PKI using Self-certified Key," in *proc. Conf. on Information Security and Cryptology (CISC'00)*, November, 25 2000.