Public Key Cryptography Sans Certificates in Ad Hoc Networks

Nitesh Saxena

School of Information and Computer Science,
University of California, Irvine
nitesh@ics.uci.edu

Abstract. Several researchers have proposed the use of threshold cryptographic model to enable secure communication in ad hoc networks without the need of a trusted center. In this model, the system remains secure even in the presence of a certain threshold t of corrupted/malicious nodes.

In this paper, we show how to perform necessary public key operations without node-specific certificates in ad hoc networks. These operations include pair-wise key establishment, signing, and encryption. We achieve this by using Feldman's verifiable polynomial secret sharing (VSS) as a key distribution scheme and treating the *secret shares as the private keys*. Unlike in the standard public key cryptography, where entities have independent private/public key pairs, in the proposed scheme the private keys are *related* (they are points on a polynomial of degree t) and each public key can be computed from the public VSS information and node identifier. We show that such related keys can still be securely used for standard signature and encryption operations (using resp. Schnorr signatures and ElGamal encryption) and for pairwise key establishment, as long as there are no more that t collusions/corruptions in the system.

The proposed usage of shares as private keys can also be viewed as a threshold-tolerant identity-based cryptosystem under standard (discrete logarithm based) assumptions.

1 Introduction

Securing communication in so-called ad hoc networks, such as mobile ad hoc networks and sensor networks, is a challenging problem due to the lack of a trusted centralized authority. Starting with the seminal proposal by Zhou and Haas [1], several researchers have proposed the use of a threshold cryptographic model to distribute trust among the nodes of the network (see [2, 3, 4, 5, 6, 7, 8, 9]), towards solving this problem. Such a model tolerates a threshold t of corruptions/collusions in the network, and at the same time, allows any set of $t+1$ nodes to make distributed decisions (for example, regarding admission of new nodes to the network). This is achieved by (t, n) polynomial secret sharing scheme of Shamir [10] that splits up the network-wide secret among n nodes using a polynomial of degree t. More specifically, if p, q be large primes s.t. q divides $p - 1$

J. Zhou, M. Yung, and F. Bao (Eds.): ACNS 2006, LNCS 3989, pp. 375–389, 2006.

then each player/node P_i receives a secret share x_i equal to a value $f(i) \bmod q$ of some t-degree polynomial f. In order to ensure the robustness of the secret sharing and secret reconstruction protocols in the presence of malicious nodes, Feldman's verifiable secret sharing (VSS) [11] is employed. Additionally, Feldman's VSS creates an $O(t * |p|)$-size public file (which is nothing but the commitments to the polynomial coefficients) from which everyone can compute and verify $y_i = g^{x_i} \bmod p$ for every $i = 1, \ldots, n$.

The above-mentioned proposals on ad hoc network security required that each node be issued a certificate and also a secret share in a distributed manner. Most recently, [12] shows that as long as each node is able to obtain an updated VSS information, there is no need for node-specific certificates. However, [12] focuses mainly on how to efficiently admit new nodes, i.e., how to create new secret shares in a distributed manner. In this work, we are concerned with the problem of how to enable secure communication among the nodes once they have been admitted. In particular, we show that the secret shares created by Feldman's VSS can be securely and efficiently used as private keys in many standard discrete-log based public-key cryptosystems, namely in a Schnorr signature scheme, in an ElGamal encryption, and in a non-interactive version of the Diffie-Hellman pairwise key establishment protocol. Note that if the VSS share x_i is treated as P_i's private key, the Feldman's VSS public information allows everyone to compute the corresponding public key y_i.

Motivation. The motivation for establishing pairwise keys is straight-forward – it is needed to secure communication between any pair of nodes, e.g., as required in various secure routing protocols, such as Ariadne [13]. Signing is required in cases when *non-repudiation* is needed, e.g., as in ARAN secure routing protocol [14]. Encryption is suitable for scenarios where an authorized node outside the network needs to send a *private* query to a node inside. An example scenario is in a wireless sensor network, where a base station sends a maintenance query to a particular sensor node (e.g., to obtain its reading of nuclear activity in the environment). However, sending the query in clear would leak critical information to an adversary who might be interested in knowing what the sensor network is installed for (e.g., for detecting a nuclear attack [15]).

Related vs. Independent Keys. It is not obvious whether the proposed usage of secret shares as private keys is safe. The reason is simple – unlike in the standard public-key cryptosystems where every user gets an *independently created* private/public key pair, here the private keys of all parties are related by being values of a t-degree polynomial (Note, for example, that any set of $t + 1$ such values determines all the others). Recall, for example, that the "text-book RSA" is not secure when public keys of two users are related [16].

Our Contributions. We show that indeed such use of the *secret shares as private keys* is just as secure as the standard discrete-log based signatures, encryption, and key establishment, as long as no more than t of the players in the group collude or are corrupted by an attacker. Note that this is the best that one can hope for because if the private keys are shares in a secret sharing with t-degree

privacy threshold, any collection of $t + 1$ such keys enables reconstruction of the whole secret-sharing and hence also all the other private keys. Our proposal renders necessary public key operations efficiently feasible in ad hoc networks, without the need of certificates.

Threshold-tolerant ID-based Cryptography. The proposed scheme is essentially equivalent to an identity-based cryptosystem that tolerates upto a threshold of corruptions/collusions. However, as compared to well-known ID-based cryptographic mechanisms, such as IBE [17] and other related schemes, our approach is more efficient and is also based on standard cryptographic assumptions.

Paper Organization. Section 2 describes some preliminaries followed by Section 3, which presents our new scheme. Finally, in Section 4, we compare our proposal to prior identity-based cryptosystems. In the rest of the paper, we use the terms group/network/system and member/node/player/user interchangeably.

2 Preliminaries

2.1 Computation, Communication and Adversarial Model

We work in the standard model of threshold cryptography and distributed algorithms known as synchronous, reliable broadcast, static adversary model. This model involves nodes equipped with synchronized clocks. We assume some nomenclature system that provides each node in the network with a unique identifier, and also that it's computationally hard for an adversary to forge identities.

We assume the existence of an on-line trusted public repository where the network-wide or group public key is published. The nodes (both within and outside the network) are connected by weakly synchronous communication network offering point-to-point channels and a reliable broadcast. To interact with a node in the network, an outsider must first be able to retrieve the group public key from the repository.

We consider the presence of the so-called "static" adversary, modeled by a probabilistic polynomial time algorithm, who can *statically*, i.e., at the beginning of the life time of the scheme, schedule up to $t < n/2$ arbitrarily malicious faults among n users in the group. Such an adversary is said to break our scheme if it is able to break the underlying key establishment, signature and encryption schemes against the standard notions of security.

2.2 Discrete Logarithm Setting and Underlying Assumptions

In this paper, we work in the standard discrete logarithm setting: p, q are large primes s.t. q divides $p - 1$ and g denotes a generator of subgroup G_q of order q in $\mathbb{Z}_p^*$. For definitional convenience we'll denote by $DL\text{-}INST(k)$ any set of instances of this discrete-log setting, i.e. of triples (p, q, g) which satisfy the above constraints, but where q is a k-bit prime and p is $poly(k)$-bit prime, long enough to fend off known attacks on the discrete logarithm.

We call function f *negligible* if for every polynomial $P(.)$, $f(k) \leq 1/P(k)$ for all sufficiently large k. We say that some event occurs with a negligible probability if the probability of this event is a negligible function of the security parameter k.

Assumption 1 (Discrete Logarithm (DL) Assumption). For every probabilistic polynomial time algorithm I, for every (p, q, g) in $DL\text{-}INST(k)$, probability $Pr[x \leftarrow \mathbb{Z}_q; I(p, q, g, g^x) = x]$ is negligible.

Assumption 2 (Computational Diffie-Hellman (CDH) Assumption). For every probabilistic polynomial time algorithm I, for every (p, q, g) in $DL\text{-}INST(k)$, probability $Pr[x \leftarrow \mathbb{Z}_q; y \leftarrow \mathbb{Z}_q; I(p, q, g, g^x, g^y) = g^{xy}]$ is negligible.

Assumption 3 (Square Computational Diffie-Hellman (SCDH) Assumption). For every probabilistic polynomial time algorithm I, for every (p, q, g) in $DL\text{-}INST(k)$, probability $Pr[x \leftarrow \mathbb{Z}_q; I(p, q, g, g^x) = g^{x^2}]$ is negligible.

2.3 Random Oracle Model (ROM)

Our proofs of security are in the so-called Random Oracle Model [19], i.e. we model hash functions like MD5 or SHA1 as ideal random oracles. Doing security analysis in the ROM model effectively means that our proofs will consider only such attacks on the cryptographic schemes we propose whose success does not change if the fixed hash function like MD5 or SHA in these schemes are replaced with truly random functions. Of course, since functions like MD5 or SHA are not truly random functions, the security analysis in the ROM model provides only a heuristic argument for the security of the actual scheme. However, such heuristic seems the best we can currently hope for. Indeed, the ROM heuristic arguments are currently the only security arguments for most practical cryptographic schemes including OAEP RSA encryption [19] and full-domain hash RSA signatures [20], as well as the two fundamental discrete-log-based cryptosystems, the hashed ElGamal encryption [21] and Schnorr signature scheme [22, 23], the two schemes which we extend to a threshold setting in this paper.

2.4 Feldman's Verifiable Secret Sharing (VSS)

The idea of secret sharing [16] is to divide a secret x into pieces or *shares* which are distributed among n players such that pooled shares of a threshold $t + 1$ number players allow reconstruction of the secret x. We use Shamir's secret sharing scheme [10] which is based on polynomial interpolation. To distribute shares among n users, a trusted dealer chooses a large prime q, and selects a polynomial $f(z)$ over $\mathbb{Z}_q$ of degree t such that $f(0) = x$. The dealer computes each user's share x_i such that $x_i = f(id_i) \mod q$, and securely transfers x_i to user M_i. Then, any group G of $t + 1$ players who have their shares can recover the secret using the Lagrange interpolation formula:

$$x = \sum_{i \in G} x_i \; l_i^G(0) \pmod{q}$$

where $l_i^G(0) = \prod_{j \in G, j \neq i} \frac{-j}{i-j} \pmod{q}$.

Feldman's *Verifiable Secret Sharing* (VSS) [11] allows players to validate the correctness of the received shares. VSS setup involves two large primes p and q, and an element $g \in \mathbb{Z}_p^*$ chosen in a way that q divides $p-1$ and g is an element of $\mathbb{Z}_p^*$ which has order q. The dealer computes commitment to the coefficients a_i $(i = 0, \cdots, t)$ of the secret sharing polynomial in the form of witnesses w_i $(i = 0, \cdots, t)$, such that $w_i = g^{a_i} \pmod{p}$, and publishes these w_i-s in some public domain (e.g., a directory server). The secret share x_i can be validated by checking that

$$g^{x_i} \stackrel{?}{=} \prod_{j=0}^{t} (w_j)^{id_i{}^j} \pmod{p}$$

2.5 Schnorr's Signature

The private key is x, chosen at random in $\mathbb{Z}_q$. The public key is $y = g^x \pmod{p}$. A Schnorr's signature [22] on message m is computed as follows. The signer picks a one-time secret k at random in $\mathbb{Z}q$, and computes the signature on m as a pair (c, s) where $s = k + cx \pmod{q}, c = H(m, r)$, and $r = g^k \pmod{p}$. Signature (c, s) can be publicly verified by computing $r = g^s y^{-c} \pmod{p}$ and then checking if $c = H(m, r)$. The Schnorr's signature scheme is proven secure against chosen message attack [24, 25] in ROM [23].

2.6 ElGamal Encryption

We use a variant of ElGamal Encryption scheme, called *Hashed* ElGamal [21], which is semantically secure under the CDH assumption in ROM. For a private key, public key pair $(x, y = g^x)$, the encryptor chooses a random $r \in \mathbb{Z}_q$ and computes the ciphertext (c_1, c_2) where $c_1 = g^r \pmod{p}$ a $c_2 = m \oplus H(y_i{}^r)$ ($\oplus$ denotes the bit-wise XOR operator). The plaintext can be obtained by computing $c_2 \oplus H(c_1^{x_i})$ from the ciphertext (c_1, c_2).

3 Our Proposal: "Secret-Shares-as-Private-Keys"

In this section we present our proposal on using secret VSS shares as private keys that renders public key operations efficiently feasible in ad hoc networks. We begin by providing a brief overview of the scheme.

3.1 Overview

The idea of the scheme is very simple. Basically, we use Feldman's VSS (summarized in Section 2.4), to build our scheme. A dealer (or a set of founding nodes in an ad hoc network) chooses a secret sharing polynomial $f(z) = a_0 + a_1 z + \cdots a_t z^t$

in $\mathbb{Z}_q$, where a_0 (also denoted as x) is the group secret key. The dealer also publishes commitments to the coefficients of the polynomial, as $w_i = g^{a_i} \pmod p$, for $i = 0, \cdots, t$. These witnesses constitute the public key of the group. To join the group, a user M_i with a unique identifier (such as an email address) id_i, receives from the dealer (or a set of $t+1$ or more nodes distributedly [12]) a secret share $x_i = f(id_i) \pmod q$ over a secure channel. The public key $y_i = g^{x_i} \pmod p$ of M_i can be computed using the public key of the group and its identifier id_i as

$$y_i = \prod_{j=0}^{t} (w_j)^{{id_i}^j} \pmod p$$

Now, any user (within or outside) the group, can send encrypted messages to M_i using its public key y_i, which M_i can decrypt using its secret key x_i. Similarly, M_i can use x_i to sign messages, which can be publicly verified using y_i. Moreover, any two users M_i and M_j can establish pairwise keys in a non-interactive manner: M_i and M_j compute $k_{ij} = (y_j)^{x_i} \pmod p$, and $k_{ji} = {y_i}^{x_j} \pmod p$, respectively. Since $K_{ij} = k_{ij} = k_{ji}$, a hash of K_{ij} can be used as session keys for secure communication between M_i and M_j.

We call these secret sharing based pairwise key establishment, signature and encryption procedures as *SS-KE*, *SS-Sig* and *SS-Enc*, respectively. *SS-Sig* is realized using the Schnorr's signature scheme, and *SS-Enc* using ElGamal encryption.

3.2 Setup and Joining

In order to setup the system, a dealer (or a set of co-founding members) first chooses appropriate parameters (p, q, g) for the group, and selects a polynomial $f(z) = a_0 + a_1 z + \cdots + a_t z^t$ in $\mathbb{Z}_q$, where a_0 (also denoted as x) is the group secret. The dealer keeps the polynomial secret and publishes commitments to the coefficients of the polynomial, as $w_i = g^{a_i} \pmod p$, for $i = 0, \cdots, t$. These witnesses constitute the public key of the group.

To join the group, a user M_i sends its unique identifier id_i to the dealer, who issues it its secret share $x_i = f(id_i) \pmod q$. (We assume there exists some kind of a unique nomenclature system for the users in the group, and that its computationally hard for anyone to forge the identities.) In an ad hoc network, the setup and joining are performed in a distributed manner. Refer to [12] for these decentralized setup and admission processes.

3.3 SS-KE: Secret Sharing Based Pairwise Key Establishment

Any pair of users M_i and M_j in the group can establish shared keys with each other using their secret keys and the group public key. M_i computes the public key y_j of M_j (knowing its identifier id_j only) as

$$y_j = \prod_{i=0}^{t} (w_i)^{{id_j}^i} \pmod p$$

M_i then exponentiates y_j to its own secret key x_i, to get $k_{ij} = {y_j}^{x_i} = g^{x_j x_i} \pmod p$. Similarly, M_j computes public key y_i of M_i as

$$y_i = \prod_{j=0}^{t} (w_j)^{id_i{}^j} \pmod p,$$

and exponentiates it to its own secret key x_j, to get $k_{ji} = {y_i}^{x_j} = g^{x_i x_j} \pmod p$. Since, k_{ij} equals k_{ji}, M_i and M_j can use $K_{ij} = H(k_{ij}) = H(k_{ji})$, as a session key for secure communication with each other.

Computational Complexity. Each party needs to compute the other party's public key via interpolation, and one exponentiation only. Using the well-known scheme of multi-exponentiation (or Shamir's trick) [26], the cost of interpolation is $O(log(n^t))$ squarings and $O(log(n^t))$ multiplications, where n denotes the total number of parties. For reasonable threshold values and network sizes, the interpolation is fairly efficient.

Next, we present the security argument for the above *SS-KE* procedure. Basically we show that an adversary, who corrupts t users, can not distinguish a key K_{IJ} for some uncorrupted user pair (M_I, M_J) from random *even* if he learns all other session keys K_{ij} for $(i,j) \neq (I,J)$.

Theorem 1 (Security of *SS-KE*). *Under the CDH Assumption in ROM, there exists no probabilistic polynomial time adversary A, which on inputs of secret keys of t corrupted users, and shared keys K_{ij} between every user pair except K_{IJ} $\{(i,j) \neq (I,J)\}$, is able to distinguish with a non-negligible probability K_{IJ} from a random value.*

Proof. We prove the above claim by contradiction, i.e, we prove that if a polynomial time adversarial algorithm A, which on inputs of secret keys of t corrupted users, and shared keys K_{ij} between every user pair except K_{IJ} $\{(i,j) \neq (I,J)\}$, is able to distinguish with a non-negligible probability K_{IJ} from a random value, then there exists a polynomial time algorithm B, which is able to break the CDH assumption in the random oracle model.

In order to construct the algorithm B which breaks the CDH assumption, we first construct a polynomial time algorithm C, which breaks the SCDH assumption. The algorithm C runs on input of an SCDH instance $y = g^x \pmod p$, and would translate the adversarial algorithm A into outputting $g^{x^2} \pmod p$.

Without loss of generality, we first assume that the adversary A corrupts t players denoted by $M_1, M_2, \cdots, M_t$. Now, the algorithm C runs as follows:

As in the simulation of Feldman's VSS, C picks $x_1, x_2, \cdots, x_t$ values corresponding to the secret keys of corrupted users, uniformly at random from $\mathbb{Z}_q$. It then sets $x_i = F(id_i)$, and employs appropriate Lagrange interpolation coefficients in the exponent to compute the public witnesses $g^{A_1}, \cdots, g^{A_t} \pmod p$, where $F(z) = x + A_1 z + \cdots + A_t z^t \pmod q$.

Corresponding to the shared keys K_{ij} between every user pair, C picks a random value R_{ij}, and runs the algorithm A on $x_1, \cdots, x_t$ and $R_{i,j}$ values. Note

that the values $x_1, \cdots, x_t$ and the witnesses have an identical distribution to an actual run of the Feldman's secret sharing protocol, and therefore A can not see the difference between C's inputs and actual protocol run. Also, since the K_{ij} values for $(i,j) \neq (I,J)$ are obtained by hashing $g^{x_i x_j}$, the only way A can tell the difference, except with negligible probability, between $K_{i,j}$ and $R_{i,j}$ for $(i,j) \neq (I,J)$, is by querying the random oracle on at least one appropriate $g^{x_i x_j}$ value. If A does tell the difference, then C records $R = g^{x_i x_j}$, and use the following equations to compute g^{x^2},

$$x = \sum_{k=1}^{t} x_k l_k^i + x_i l_i^i \pmod{q}$$

$$x = \sum_{k=1}^{t} x_k l_k^j + x_j l_j^j \pmod{q}$$

(l_k^i denotes the lagrange coefficient $l_k^G(0)$, where $G = \{1, \cdots, t, i\}$).
Multiplying above two equations, we get

$$x^2 = (\sum_{k=1}^{t} x_k l_k^i)(\sum_{k=1}^{t} x_k l_k^j) + x_i x_j l_i^i l_j^j \pmod{q}$$

This implies,

$$g^{x^2} = g^{(\sum_{k=1}^{t} x_k l_k^i)(\sum_{k=1}^{t} x_k l_k^j)} R^{l_i^i l_j^j} \pmod{p}$$

If A doesn't tell the difference between $K_{i,j}$ and $R_{i,j}$ for $(i,j) \neq (I,J)$, then it must tell the difference between $K_{I,J}$ and $R_{I,J}$. However, as above, this is only possible, except with negligible probability, if A queries $g^{x_I x_J}$ to the random oracle. Them C records this value (say K) and computes g^{x^2} similarly as above, using the following equation

$$g^{x^2} = g^{(\sum_{k=1}^{t} x_k l_k^I)(\sum_{k=1}^{t} x_k l_k^J)} K^{l_I^I l_J^J} \pmod{p}$$

Now, we will use C to construct B to break a CDH instance (g^u, g^v). This is very simple as outlined in [27]: B runs C on input g^u, then on g^v, and finally on $g^{u+v} = g^u g^v$, and receives $g^{u^2}, g^{v^2}, g^{(u+v)^2}$, respectively. Now, since $(u+v)^2 = u^2 + v^2 + 2uv \pmod{q}$, B can easily compute g^{uv} from the outputs of C.

Clearly, $Pr(B) = Pr(C)^3$, where $Pr(B), Pr(C)$, denote the probabilities of success of B and C respectively.

3.4 SS-Sig: Secret Sharing Based Signatures

As mentioned previously, we realize *SS-Sig* using the Schnorr's signature scheme.

Signing. To sign a message m , M_i (having secret key x_i), picks a random secret $k \in Z_q$ and computes $r = g^k \pmod{p}$. It then outputs the signature as a pair (c, s), where $c = H(m, r)$ and $s = k + r x_i \pmod{q}$.

Verification. In order to verify the above signature (c, s), a recipient first computes the public key y_i of the signer M_i using its identity id_i as $y_i = \prod_{j=0}^{t} (w_j)^{id_i{}^j} \pmod p$, and then verifies whether $c = H(m, r)$, where $r = g^s y_i{}^{-c} \pmod p$.

Computational Complexity. The signer needs to compute only one exponentiation, while the verifier requires one interpolation operation, two exponentiations and and two multiplications.

In the following theorem, we argue the security of *SS-Sig*. More precisely, we argue that *SS-Sig* remains secure against existential forgery under chosen message attack (CMA) [24] in ROM as long as the discrete logarithm assumption holds. Notice that *SS-Sig* is different from regular signatures in the sense that the users generate signatures with related (and not independent) secret keys, and the adversary knows at most t of these secret keys.

For clarity of our argument, we first recall the argument for security of the underlying Schnorr's signature scheme against CMA attack in ROM and discrete logarithm assumption; the simulator algorithm, on input $y = g^x$, can produce Schnorr's signatures on any m by picking s and c at random in $\mathbb{Z}_q$, computing $r = g^s y^{-c} \pmod p$ and setting $H(m, r) = c$. This simulator can also translate the adversary's forgery into computing $dlog_g y$ as follows. It runs the adversary until the adversary outputs a forgery (c, s) on some message m. Note that because H is a random function, except for negligible probability, the adversary must ask to H a query (m, r) where $r = g^s y^{-c} \pmod p$, because otherwise it could not have guessed the value of $c = H(m, r)$. The simulator then rewinds the adversary, runs it again by giving the same answers to queries to H until the query (m, r), which it now answers with new randomness c'. If the adversary forges a signature on m in this run, then, except for negligible probability, it produces s' s.t. $r = g^{s'} y^{-c'} \pmod p$, and hence the simulator can now compute $dlog_g y = (s - s')/(c' - c) \pmod q$. One can show that if the adversary's probability of forgery is ϵ, this simulation succeeds with probability $\epsilon^2/4q$: $O(\epsilon)$ probability that the adversary forges in the first run times the $O(\epsilon/qH)$ probability that it will forge on the second run and that it will choose to forge on the same (m, r) query out of its q queries to H. We refer to [23] for the full proof.

Theorem 2 (Security of *SS-Sig*). *Under the DL assumption in ROM, as long as the adversary corrupts no more than t users,* SS-Sig *is secure against the chosen-message attack for every remaining uncorrupted user*

Proof. We prove the following claim: if there exists a polynomial time algorithm A, which on inputs the secret keys of t corrupted users, is able to create an existential forgery in CMA model corresponding to an uncorrupted user, then there exists a polynomial time algorithm B, which can break the DL assumption in ROM.

We construct an algorithm B, which runs on input of a DL instance $y = g^x \pmod p$, and would translate the adversarial algorithm A into outputting x. We first assume that the adversary A corrupts t players denoted by $M_1, M_2, \cdots, M_t$, w.l.o.g.

Note that in our multiple user scenario, the adversary A can request the signature oracle to sign chosen messages corresponding to any honest player. In other words, when A sends (m, id_i) to the signature oracle, the oracle responds with a signature on message m signed with x_i.

B picks $x_1, x_2, \cdots, x_t$ values corresponding to the secret keys of corrupted users, uniformly at random from $\mathbb{Z}_q$. It then sets $x_i = F(id_i)$, and employs appropriate Lagrange interpolation coefficients in the exponent to compute the public witnesses $g^{A_1}, \cdots, g^{A_t} \pmod p$, where $F(z) = x + A_1 z + \cdots + A_t z^t \pmod q$. Since, $x = \sum_{k=1}^{t} x_k l_k^i + x_i l_i^i \pmod q$, B can compute the public key y_i, corresponding to an honest player M_i $(i \geq t+1)$ as

$$y_i = (y / g^{\sum_{k=1}^{t} x_k l_k^i})^{1/l_i^i} \pmod p \tag{1}$$

B now runs A on inputs $x_1, x_2, \cdots, x_t$ and simulates the signature oracle on A's query (m, id_i), by picking s and c at random in $\mathbb{Z}_q$, computing $r = g^s {y_i}^{-c} \pmod p$ and setting $H(m, r) = c$. A then outputs a forgery (C, S) on some message M corresponding to user M_i. Note that because H is a random function, except for negligible probability, A must have asked to H a query (M, R) where $R = g^S {y_i}^{-C} \pmod p$, because otherwise it could not have guessed the value of $C = H(M, R)$. B then reruns A by giving the same answers to queries to H until the query (M, R), which it now answers with new randomness C'. If A outputs the forgery on the same message M, but this time for a different user M_j $(i \neq j)$ then, except for negligible probability, it produces S' s.t. $R = g^{S'} {y_j}^{-C'} \pmod p$. B can now (using equation 1) compute

$$x = (S - S' + (C/l_i^i) \sum_{k=1}^{t} x_k l_k^i - (C'/l_j^j) \sum_{k=1}^{t} x_k l_k^j) / (C/l_i^i - C'/l_j^j) \pmod q$$

As in the security proof of Schnorr's Signatures, the probability of success of B would be $\epsilon^2/4q$, where ϵ represents the success probability of A and q is the total number of queries to $H()$.

3.5 SS-Enc: Secret Sharing Based Encryption

We use Hashed ElGamal encryption scheme in the *SS-Enc* procedure.

Encryption. In order to encrypt a message m for a user M_i in the group, the encryptor computes the public key of M_i as $y_i = \prod_{j=0}^{t} (w_j)^{{id_i}^j} \pmod p$, chooses a random $r \in \mathbb{Z}_q$ and then sends a pair (c_1, c_2) to M_i, where $c_1 = g^r \pmod p$ and $c_2 = m \oplus H({y_i}^r)$ ($\oplus$ denotes the bit-wise XOR operator).

Decryption. M_i recovers the message by computing $c_2 \oplus H(c_1^{x_i})$ from the ciphertext (c_1, c_2).

Computational Complexity. In the above procedure, the encryptor performs one interpolation and two exponentiation. The decryptor, on the other hand, needs to compute only a single exponentiation.

Before presenting the security argument for *SS-Enc*, we briefly discuss the indistinguishability notion [28]. Indistinguishability is defined as the following game: the adversary is first run on input of the public key and outputs two messages to be challenged upon. Next, one of these messages is encrypted and given to the adversary. The adversary is said to win this game if he can output which message was encrypted with non-negligible probability greater than half.

The above notion of indistinguishability was designed for a single user scenario, where multiple messages are being encrypted for one user. However, to capture the security of *SS-Enc*, where there are multiple users in the group and the messages are encrypted using *related* keys, we adopt the *multi-user* indistinguishability notion of Baudron et al. [29] and Bellare et al. [30]. In this notion, the adversarial game is as follows: first the adversary is given as input n public keys $(pk_1, \cdots, pk_n)$ of all the users. The adversary then outputs two vectors of n messages $M_0 = \{m_{01}, \cdots, m_{0n}\}$ and $M_1 = \{m_{11}, \cdots, m_{1n}\}$, which might be related or same, to be challenged upon. One of the message vectors M_b (b is 0 or 1) is then encrypted with n public keys (the order of the encryption is preserved, i.e., m_{bi} is encrypted with pk_i). The adversary is said to win the game if he can, with probability non-negligibly greater than half, output which message was encrypted. It has been shown in [30, 29] that an encryption scheme secure in the sense of single-user indistinguishabilty is also secure in the sense of multi-user indistinguishability.

Following is the security argument for *SS-Enc* based on a slightly modified multi-user indistinguishability notion, as described above (Basically, the adversary is only challenged for the encryptions of $n - t$ honest users in the group).

Theorem 3 (Security of *SS-Enc*). *Under the CDH assumption in ROM, as long as the adversary corrupts no more than t users,* SS-Enc *is secure in the sense of multi-user indistinguishability notion.*

Proof. As usual, the proof goes by contradiction, i.e., we proof that if there exists a polynomial time algorithm A, which on inputs the secret keys of t corrupted users, is able to break the multi-user indistinguishability notion, then there exists a polynomial time algorithm B, which can break the CDH assumption in ROM.

We construct an algorithm B, which running on input of a CDH instance $U = g^u, V = g^v$, translates the algorithm A into outputting g^{uv}. As usual, we first assume that the adversary A corrupts t players denoted by $M_1, M_2, \cdots, M_t$, w.l.o.g.

As in the security proof of *SS-Sig*, B picks $x_1, x_2, \cdots, x_t$ values corresponding to the secret keys of corrupted users, uniformly at random from $\mathbb{Z}_q$. It then sets $x_i = F(id_i)$, and employs appropriate Lagrange interpolation coefficients in the exponent to compute the public witnesses $g^{A_1}, \cdots, g^{A_t} \pmod p$, where $F(z) = u + A_1 z + \cdots + A_t z^t \pmod q$. Since, $u = \sum_{k=1}^{t} x_k l_k^i + x_i l_i^i \pmod q$, B can compute the public key y_i, corresponding to an honest player M_i ($i \geq t+1$) using Equation 1.

To help the reader understand the construction of our translator algorithm B, we first recall the how the translator works in the security proof (under CDH and ROM) of single-user hashed ElGamal. The translator works as follows: on input of a CDH instance $(U = g^u, V = g^v)$, it first runs the adversary on input g^u. The adversary outputs two messages m_0, m_1. The translator picks one message m_b ($b = 0$ or 1) at random, and sends the encryption (c_1, c_2) to the adversary, where $c_1 = V * g^r \pmod p$ and $c_2 = R$ (r is a random value in $\mathbb{Z}_q$ and R is a random pad of same length as the message). In the random oracle model, the only way the adversary can distinguish this encryption is by querying the random oracle on value $O = c_1^u = U^{r+v}$, which will be recorded by the translator, and used to compute $g^{uv} = OU^{-r}$. If there are a total of q queries being made to the oracle, this means that the probability of success of translator would be $1/q$ times the probability of success of the adversary.

Now, we are ready to describe the translation based on our multi-user setting: B runs A on inputs the secret keys $x_1, \cdots, x_t$ corresponding to the corrupted users, and the public keys $y_{t+1}, \cdots, y_n$ of all honest ones. A outputs two vectors of $n - t$ messages $M_0 = \{m_{0i}\}$ and $M_1 = \{m_{1i}\}$, where $i = t + 1, \cdots, n$, to be challenged upon. B then picks M_b (b is 0 or 1) and sends to A the vector $\{(V * g^{r_i}, R_i)\}$, where r_i is a random value in $\mathbb{Z}_q$, and R_i is a random pad equally long as the message m_{bi}, for $i = t+1, \cdots, n$. The only possibility for A to win this game, is by querying the random oracle on at least one of the value $O = (V * g^{r_j})^{x_j}$, for some $j \in \{t+1, \cdots, n\}$. B records this value, and assuming that it corresponds to M_j, it computes g^{uv} as follows:

$$u = \sum_{k=1}^{t} x_k l_k^j + x_j l_j^j \pmod q$$

This implies that

$$g^{uv} = g^{v \sum_{k=1}^{t} x_k l_k^j} g^{v x_j l_j^j} \pmod q$$

and

$$g^{uv} = V^{\sum_{k=1}^{t} x_k l_k^j} V^{x_j l_j^j} \pmod p$$

Since, $O = (V * g^{r_j})^{x_j}$, this means $V^{x_j} = O {y_j}^{-r_j}$, and therefore,

$$g^{uv} = V^{\sum_{k=1}^{t} x_k l_k^j} O {y_j}^{-r_j l_j^j} \pmod p$$

Given that there are a total of q queries to the random oracle, the probability of success of B would be probability of success of A times $1/q(n - t)$, as only one query will yield correct g^{uv} value and each query might correspond to one j value in $\{t + 1, n\}$.

Remark: *Extension to Chosen Ciphertext Security.* The hybrid encryption techniques for extending standard hashed ElGamal to chosen ciphertext security (refer to [31], [32]) can be used to achieve chosen ciphertext security for the *SS-Enc* scheme.

4 Comparison with ID-Based Cryptography

As previously pointed out in the introduction section, our proposed scheme can be viewed as an identity-based cryptosystem based on threshold assumption. Basically, a trusted center provides each user with a secret value (VSS share in our case) derived from the unique identifier of the user, and publishes the VSS information as its public key. Knowing the identifier of a particular user and also the public key of the trusted center, one can send encrypted messages and verify signatures. This is equivalent to IBE [17], and ID-based signatures [33], apart from the fact that our scheme becomes insecure if there are more than a threshold of collusions or corruptions. However, unlike other ID-based schemes, our proposal is based on standard (discrete logarithm) assumptions. Moreover, for reasonable group sizes and threshold values, our scheme is much more efficient than these prior ID-based schemes, which require costly computations (such as scalar point multiplications, map-to-point operations and bilinear mappings [17]) in elliptic-curves. For example, for a group size of around 100, and threshold of 10 (10% of group size), the encryption in our scheme would require less than 70 squarings, less than 70 modular multiplications, and only 2 modular exponentiations. The decryption would just require 1 exponentiation. On the other hand, IBE requires 1 map-to-point operation, 2 scalar point multiplications, and 1 bilinear mapping, for encryption, and 1 bilinear mapping for decryption. It is well-known that for appropriate security parameters, the IBE computations are extremely costly (e.g., a bilinear mapping takes around 80ms, scalar point multiplication costs around 30 ms, while a single modular exponentiation is only a few milliseconds on fast processors). Refer to, e.g., [8] for details regarding these cost comparisons.

Acknowledgments

The author is thankful to Stanisław Jarecki for his help with the paper, and anonymous reviewers for their useful comments.

References

1. Zhou, L., Haas, Z.J.: Securing Ad Hoc Networks. IEEE Network Magazine **13** (1999) 24–30
2. Luo, H., Lu, S.: Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks. Technical Report TR-200030, Dept. of Computer Science, UCLA (2000) Available online at http://citeseer.ist.psu.edu/luo00ubiquitous.html.
3. Kong, J., Zerfos, P., Luo, H., Lu, S., Zhang, L.: Providing Robust and Ubiquitous Security Support for MANET. In: IEEE 9th International Conference on Network Protocols (ICNP). (2001) 251–260
4. Kong, J., Luo, H., Xu, K., Gu, D.L., Gerla, M., Lu, S.: Adaptive Security for Multi-level Ad-hoc Networks. In: Journal of Wireless Communications and Mobile Computing (WCMC). Volume 2. (2002) 533–547

5. Luo, H., Zerfos, P., Kong, J., Lu, S., Zhang, L.: Self-securing Ad Hoc Wireless Networks. In: Seventh IEEE Symposium on Computers and Communications (ISCC '02). (2002)
6. Narasimha, M., Tsudik, G., Yi, J.H.: On the Utility of Distributed Cryptography in P2P and MANETs: The Case of Membership Control. In: IEEE International Conference on Network Protocol (ICNP). (2003) 336–345
7. Saxena, N., Tsudik, G., Yi, J.H.: Admission Control in Peer-to-Peer: Design and Performance Evaluation. In: ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN). (2003) 104–114
8. Saxena, N., Tsudik, G., Yi, J.H.: Identity-based Access Control for Ad-Hoc Groups. In: International Conference on Information Security and Cryptology (ICISC). (2004)
9. Liu, D., Ning, P.: Establishing Pairwise Keys in Distributed Sensor Networks. In: ACM Conference on Computers and Communication Security. (2003) 52–61
10. Shamir, A.: How to Share a Secret. Communications of the ACM **22** (1979) 612–613
11. Feldman, P.: A Practical Scheme for Non-interactive Verifiable Secret Sharing. In: 28th Symposium on Foundations of Computer Science (FOCS). (1987) 427–437
12. Saxena, N., Tsudik, G., Yi, J.H.: Efficient node admission for short-lived mobile ad hoc networks. In: International Conference on Networking Protocols (ICNP). (2005)
13. Hu, Y.C., Perrig, A., Johnson, D.B.: Ariadne: A secure on-demand routing protocol for ad hoc networks. In: Proceedings of the Eighth ACM International Conference on Mobile Computing and Networking (Mobicom 2002). (2002)
14. Dahill, B., Levine, B., Royer, E., Shields, C.: A secure routing protocol for ad hoc networks. Technical Report UM-CS-2001-037, University of Massachusetts (2001)
15. Hills, R.: Sensing for danger. Science Technology Report (2001) Available at http://www.llnl.gov/str/JulAug01/Hills.html.
16. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography. CRC Press series on discrete mathematics and its applications. (1997) ISBN 0-8493-8523-7.
17. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: CRYPTO. (2001) 213–229
18. Bazzi, R.A., Konjevod, G.: On the establishment of distinct identities in overlay networks. In: Principles of Distributed Computing (PODC). (2005)
19. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: ACM Conference on Computer and Communications Security. (1993) 62–73
20. Bellare, M., Rogaway, P.: The exact security of digital signatures — how to sign with RSA and Rabin. In: EUROCRYPT'96. Volume 1070 of LNCS. (1996) 399–416
21. ElGamal: A public-key cryptosystem and a signature scheme based on discrete logarithms. In: IEEE Transactions in Information Theory (IT-31). (1999) 469–472
22. Schnorr, C.P.: Efficient Signature Generation by Smart Cards. Journal of Cryptology **4** (1991) 161–174
23. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: EUROCRYPT'96. Volume 1070 of LNCS. (1996) 387–398
24. Goldwasser, S., Micali, S., Rivest, R.L.: A "paradoxical" solution to the signature problem. In: IEEE Annual Symposium of Foundations of Computer Science (FOCS'84). (1984) 441–448
25. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing **17** (1988) 281–308

26. Möller, B.: Algorithms for multi-exponentiation. In: Selected Areas in Cryptography. (2001) 165–180
27. Maurer, U.M., Wolf, S.: Diffie-Hellman oracles. In: CRYPTO'96. Volume 1109 of LNCS. (1996) 268–282
28. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences, **28** (1989) 270–299
29. Baudron, O., Pointcheval, D., Stern, J.: Extended notions of security for multicast public key cryptosystems. In: International Colloquium on Automata, Languages and Programming (ICALP'00). Volume 1853 of LNCS. (2000) 499–511
30. Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: Security proofs and improvements. In: EUROCRYPT'00. Volume 1807 of LNCS. (2000) 259–274
31. Bellare, M., Boldyreva, A., Palacio, A.: An uninstantiable random-oracle-model scheme for a hybrid encryption problem. In: EUROCRYPT'04. Volume 3027 of LNCS. (2004) 171–188
32. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: CRYPTO'99. Volume 1666 of LNCS. (1999) 537–554
33. Cha, J., Cheon, J.: An ID-based signature from Gap-Diffie-Hellman Groups. In: Proceedings of International Workshop on Practice and Theory in Public Key Cryptography. Volume 2567 of LNCS. (2003) 18–30