Non-Interactive Self-Certification for Long-Lived Mobile Ad Hoc Networks

Nitesh Saxena, *Member, IEEE* and Jeong Hyun Yi, *Member, IEEE*

***Abstract*— Mobile ad hoc networks (MANETs) have many well-known applications in military settings as well as in emergency and rescue operations. However, lack of infrastructure and centralized control make MANETs inherently insecure, and therefore specialized security services are needed for their deployment. *Self-certification* is an essential and fundamental security service in MANETs. It is needed to securely cope up with dynamic membership and topology, and to bootstrap other important security primitives and services (such as secure routing and group key management) without the assistance of any centralized trusted authority. An ideal protocol must involve minimal interaction among the MANET nodes, since connectivity can be unstable. Also, since MANETs are often composed of weak or resource-limited devices, self-certification protocol must be efficient in terms of computation and communication.**

In this paper, we propose a power-aware and fully non-interactive self-certification protocol based on bi-variate polynomial secret sharing and a non-interactive threshold signature scheme. In contrast with prior work, our techniques do not require any interaction and do not involve any costly reliable broadcast communication among MANET nodes. We thoroughly analyze our proposal and show that it compares favorably to previous mechanisms.

Keywords: Security protocol, self-configuration, threshold cryptography, authentication, key management, ad hoc networks

I. Introduction

Unlike cellular networks whose infrastructure includes base stations or access points, routers and switches that are fixed and wired together, mobile ad hoc networks (MANETs) are infrastructure-less and mobile nodes act as wireless routers. Lack of infrastructure and lack of centralized control, coupled with a dynamic network topology, result in vulnerabilities that do not exist in wired networks, and therefore specialized security services are needed prior to deployment of MANETs.

Self-certification (or self-configurable authentication) is a fundamental security service in MANETs; it is required to ascertain membership eligibility and to bootstrap other important security services, such as secure routing (e.g., [15], [14]) and secure group communication (e.g., [39], [38]). An example application of self-certification is in a MANET used for rescue and disaster recovery. In such an application, a member of one disaster recovery team (e.g., a policeman) might need to be certified by another team (e.g., that of fire-fighters) before secure communication between the two can be established.

Node certification in MANETs cannot be performed centrally. Requiring constant presence (availability) of a central fixed entity is not realistic for many types of MANETs due to a number of reasons. First, such an entity is a single point of failure. Second, it represents an attractive and high-payoff target for attacks. Third, topology changes due to mobility and node outages may cause the central entity to be unreachable and thus unable to perform its duties in the parts of a MANET not connected to it. This motivates us to investigate self-configurable authentication techniques that function in a distributed or decentralized manner. Since our emphasis is on security, the natural technology to consider is threshold cryptography.

The concept of threshold cryptography involves distributing cryptographic primitives (such as decryption or digital signatures) in order to secure them against corruption of a certain number of parties, i.e., a threshold. For example, a (t, n) threshold signature scheme [8] allows, in a group of n parties, to share the ability to digitally sign messages in such a way that any t parties can do so jointly, whereas, no coalition of up to $(t-1)$ parties can. Such a threshold signature scheme is resilient against the so-called *static adversary* who corrupts at most $(t-1)$ parties in the entire lifetime of the system.

More advanced *proactive* cryptographic schemes [13] offer improved resistance against corruptions. Time is divided into *update rounds*, and the proactive scheme offers the same combination of security and robustness even in the presence of so-called *mobile adversaries* [31], which can potentially corrupt a set of up to $(t-1)$ parties in each update round (e.g, every day). This is done by the *proactive update* procedure which involves parties randomly re-sharing the shared secret at the start of each update round.

Two features of MANETs make self-certification a very challenging problem. First, MANET devices often have very weak computational facilities and battery power. Second, MANET nodes usually function in an asynchronous (on/off) manner, often becoming temporarily unavailable to one another. Therefore, an ideal solution must be efficient in terms of both computation and communication[1]. It must also involve minimal (ideally, *none* at all) interaction among the nodes of the network which requires synchronous communication.

In this paper, we distinguish between *"long-lived"* and *"short-lived"* MANETs. Long-lived MANETs are formed for

Nitesh Saxena is with Computer Science and Engineering Department, Polytechnic Institute of New York University, USA. E-mail: nsaxena@poly.edu.

Jeong Hyun Yi is with School of Computing, Soongsil University, Korea. E-mail: jhyi@ssu.ac.kr.

[1]Communication is directly related to the consumption of battery power in MANET devices [1].

the long haul and require strong robustness/resilience. They need to be protected against powerful *mobile adversaries* through periodic updates of the secret shares possessed by the nodes [13]. Short-lived MANETs, on the other hand, are ephemeral and need to be resilient against weaker *static adversaries*. A MANET formed for the duration of a conference program committee meeting (typically, one day) is one example of a short-lived MANET. Another example is a temporary MANET formed by a group of soldiers on a battlefield as they stay in close proximity to each other. A squadron of military aircraft flying in formation also represents a short-lived MANET. Whereas, a MANET formed by a group of college students taking part in a semester-long project course is an example of a *long-lived* MANET. Another example of a long-lived MANET is a flotilla of merchant vessels or military ships sailing together, say, across the Pacific Ocean.

A number of self-certification techniques have been proposed in recent years [20], [19], [23], [28], [34], [35] (We review them in the following section). The focus of these schemes is on long-lived MANETs. They are based on (t, n) threshold cryptography and allow any set of t-out-of-n nodes (called sponsors) to certify a new node by issuing it:

(1) a share of a group secret (to be used in future admissions) through a distributed secret share issuance protocol, and
(2) a membership certificate or token (to be used for future secure communication) through a threshold signing protocol

Unfortunately, all previous schemes are far from ideal. They are **heavily interactive** among the sponsors as far as either (1) or (2) is concerned. Furthermore, they are computationally very expensive in performing (2). This severely limits their practicality.

We observed in [36] that self-certification for short-lived MANETs can be realized by only issuing node-specific secret shares (item (1) above) and thus obviating the need for membership certificate issuance[2] in short-lived MANETs. The nodes can use their secret shares (and/or the group public key) for the purpose of secure communication with each other. We constructed an efficient and fully non-interactive self-certification technique based on bi-variate secret sharing and evaluated it in the context of short-lived MANETs in [36]. In long-lived MANETs, on the other hand, both node-specific secret shares as well as certificates are needed. This makes self-certification process more challenging.

Contributions: In this paper, we extend our approach of [36] for long-lived MANETs. In particular, we present **fully non-interactive** self-certification protocol for long-lived MANETs based on bi-variate polynomial secret sharing and threshold version [4] of the so-called BLS signature scheme [6]. In contrast with prior work for long-lived MANETs [20], [19], [23], [28], [34], [35], our protocol does not require any interaction and do not involve any costly reliable broadcast communication among MANET nodes. We thoroughly analyze our proposal via experiments and show that it compares favorably to previous mechanisms

The scope of this paper is only limited to self-certification and admission of a new node in a MANET setting. The complementary problem of distributed node revocation is of independent interest, which has been addressed in our prior work [35].

Organization: The rest of the paper is organized as follows: we first review prior work in Section II. Some cryptographic background is provided in Section III, followed by the system and security model in Section IV. We then describe, in Section V, the proposed self-certification mechanism. The detailed performance results, analysis and comparison are presented in Section VI.

II. Related Work

We now review relevant prior work for robust self-certification in MANETs

A. *Interactive Threshold Signing*

Zhou and Haas [44] first suggested the use of threshold cryptography to secure mobile ad hoc networks. They proposed a distributed certification authority (CA) which issues certificates (using some threshold signature [8] protocol) to nodes joining the network. The proposed approach is hierarchical in the sense that only a selected set of nodes can serve as parts of the certification authority, i.e., take part in admission decisions. Moreover, contacting distributed CA nodes in a multi-hop and ever-changing MANET might not always be possible. Although quite attractive, this idea is not directly applicable for the purposes of self-configuration in MANETs.

Kong, et al. considered the same problem in series of papers [20], [19], [23], [22] and proposed a set of protocols for providing ubiquitous and robust security services for MANETs. They adapted the model of Zhou and Haas so that any node can participate in decision of new node admission, thus maintaining the true "peer" nature of a MANET and providing increased availability. The security of their mechanism relies upon a specific variant of the proactive threshold RSA signature scheme. Unfortunately, this scheme is neither robust [28] (i.e., it can not tolerate malicious nodes) nor secure [16]. Note that all previously known provably secure threshold/proactive RSA signature schemes fail to yield self-configuration for MANETs.

Narasimha, et al. [28] and Saxena, et al. [35] proposed similar protocols based on threshold DSA [10]. While provably secure, the solution is quite inefficient since it is heavily interactive among sponsoring nodes.

B. *Non-Interactive Threshold Signing*

Out of all the known discrete-logarithm based threshold signature schemes, i.e., threshold DSA [10], threshold Schnorr [40], and threshold BLS [4], only the latter is non-interactive. In [35], Saxena, et al. proposed the self-certificate protocol that

[2]In short-lived MANETs, since there is no need for proactive updates, the polynomial used for sharing the group secret *remains constant* throughout the lifetime of the MANET and the commitment to this polynomial becomes a part of the group public key. The commitment to each node's secret share is derivable from (and thus automatically bound to) the group public key. Therefore, node-specific membership tokens are not needed.

uses uni-variate polynomial secret sharing [37] and threshold BLS for certificate issuance. Although this scheme is non-interactive when issuing a certificate for new node, its distributed secret share issuance (which we review in Section III-A) still requires interaction due to the Lagrange interpolation of uni-variate secret sharing. In the rest of the paper we refer to this protocol as *Univariate polynomial based threshold BLS* or U-BLS.

The self-certification technique developed in this paper has completely non-interactive distributed share issuance and certificate issuance. The proposed technique uses secret sharing based on so-called bi-variate polynomials. Bivariate polynomials have previously been employed for related purposes in the literature [2], [27], [3]. In particular, [21] presents a key pre-distribution scheme for sensor networks using bi-variate polynomials [3] *in the presence of a centralized authority*. The protocol we propose is fully distributed and allows nodes in a MANET to readily and efficiently share pairwise secret keys without any centralized support.

III. Preliminaries

Notation used in the rest of paper is summarized in Table I.

TABLE I
NOTATION USED IN THE REST OF THIS PAPER.

P_i	network node i
id_i	identity for P_i
t	admission threshold
n	total number of network nodes
$\mathbb{G}$	cyclic group in finite fields
$\mathbb{G}_1, \mathbb{G}_2$	cyclic GDH groups of order q
P	generator of group $\mathbb{G}_1$
$\hat{e}$	bilinear map s.t. $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$
H	hash function such as SHA-1 or MD5
H_1	special hash function s.t. $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$
x_i	secret share of P_i
$x_i^{(j)}$	partial share for P_i by P_j
T_i	membership token for P_i
PK_i	temporary public key of P_i
$S_i(m)$	P_i's signature on message m
$K_{i,j}$	pairwise key between P_i and P_j
$E_{K_{i,j}}$	encryption with $K_{i,j}$

A. Threshold Secret Sharing

We review Shamir's secret sharing scheme [37] which is based on uni-variate polynomial interpolation. We will refer to it as TSS. To distribute shares of a secret x among n entities, a trusted dealer *TD* chooses a polynomial $f(z)$ over $\mathbb{Z}_q$ of degree $(t-1)$: $f(z) = \sum_{i=0}^{t-1} a_i z^i \pmod q$ where the constant term a_0 is set to the network secret x; $f(0) = a_0 = x$. *TD* computes each entity's share x_i such that $x_i = f(id_i)$, where id_i is an identifier of entity P_i, and securely transfers x_i to P_i. Note that after distributing at least t secret shares, the dealer is no longer required.

Then, any set of t entities who have their shares can recover the secret using the Lagrange interpolation formula: $f(z) = \sum_{i=1}^{t} x_i\ \lambda_i(z) \pmod q$, where $\lambda_i(z) = \prod_{j=1,j\neq i}^{t} \frac{z-id_j}{id_i-id_j} \pmod q$. Since $f(0) = x$, the shared secret may be expressed as: $x = f(0) = \sum_{i=1}^{t} x_i\ \lambda_i(0) \pmod q$ Thus, the secret x can be recovered only if at least t shares are combined. In other words, no coalition of less than t entities yields any information about x.

The distributed share issuance protocol based on uni-variate polynomial interpolation (as proposed in [20], [19], [23]) is reviewed as follows. P_j, who does not yet have its share x_j, can become a bona fide member node, who can participate in future node admission, when it obtains t partial secret shares from neighboring node P_i. To do this, P_i provides P_j with its partial secret share $x_j^{(i)}$ as: $x_j^{(i)} = x_i \lambda_i(id_j) \pmod q$. By combining these $x_j^{(i)}$-s, P_j obtains its secret share x_j such that $x_j = \sum_{i=1}^{t} x_j^{(i)} \pmod q$. However, when each P_i issues P_j a partial share $x_j^{(i)}$, P_j (or an adversary who corrupts P_j) can easily recover x_i (which is P_i's secret) and in turn the network secret x; since $\lambda_i(id_j)$ is publicly known, P_j can obtain x_i by dividing $x_j^{(i)}$ by $\lambda_i(id_j)$. To remedy this, P_i-s must apply a technique specified in [20] to randomize x_j-s. We call it *random shuffling* in the rest of this paper. The detailed procedure is as follows: each pair of neighboring nodes (P_i, P_k) securely exchange a shuffling factor r_{ik}. One of the pair adds r_{ik} to its partial share and the other subtracts r_{ik} from its partial share. For P_i, there are total of $t-1$ shuffling factors and it must apply all of them, by either addition or subtraction, to its partial share $x_j^{(i)}$. The result is a completely-shuffled partial share $\tilde{x}_j^{(i)} = x_j^{(i)} + \sum_{k=1,k\neq i}^{t} sign(id_i - id_k) r_{ik}$ where $sign(x) = 1$ if $x > 0$ and $sign(x) = -1$ if $x < 0$. Therefore, once P_j receives t shuffled partial shares $\tilde{x}_j^{(i)}$-s, it recovers its own share as: $\sum_{i=1}^{t} \tilde{x}_j^{(i)} = \sum_{i=1}^{t} x_j^{(i)} + \sum_{i=1}^{t} \sum_{k=1,k\neq i}^{t} sign(id_i - id_k) r_{ik} = \sum_{i=1}^{t} x_j^{(i)} + 0 = x_j$. Note that the protocol requires "random shuffling", is interactive among sponsoring nodes and requires heavy communication.

B. Elliptic Curves

Our cryptographic protocol is based on the Discrete-Logarithm problems in Elliptic Curves (EC). We briefly review the underlying problems. For a prime $p > 3$, an elliptic curve $E(\mathbb{F}_p)$ over the field $\mathbb{F}_p$[3] consists of a set of points (x, y) with $x,\ y \in \mathbb{F}_p$ which satisfy the equation $y^2 = x^3 + ax + b$ where the discriminant $4a^3 + 27b^2 \neq 0$. $E(\mathbb{F}_p)$ constitutes an Abelian group under the point-addition [18] operation with the point infinity as the identity of the group. The order of this group is denoted by $\#E(\mathbb{F}_p)$. The domain parameters are represented by $(p, \mathbb{F}_p, a, b, P, q)$ where $P \in E(\mathbb{F}_p)$ has prime order q such that q divides $\#E(\mathbb{F}_p)$.

Let $\mathbb{G}$ be a cyclic group $\mathbb{G}$ which is a subgroup of the points generated by $P \in E(\mathbb{F}_p)$ of order q.

Definition 1 (EC-DL Problem): Given a pair of $\mathbb{G}$ elements (P, aP) for $a \in \mathbb{Z}_q^*$, find a. If this problem is hard, we say *the EC Discrete Logarithm (EC-DL) assumption holds in* $\mathbb{G}$.

Definition 2 (EC-CDH Problem): Given a triple (P, aP, bP) for $a, b \in \mathbb{Z}_q^*$, compute abP. If this problem

[3]Some elliptic curves are defined over extension fields $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$, where m is a positive exponent.

is hard, we say *the EC Computational Diffie-Hellman (EC-CDH) assumption holds in* $\mathbb{G}$.

Definition 3 (EC-DDH Problem): Given a quadruple (P, aP, bP, cP) for $a, b, c \in \mathbb{Z}_q^*$, decide whether $c = ab$. If this problem is hard, we say *the EC Decisional Diffie-Hellman (EC-DDH) assumption holds in* $\mathbb{G}$.

Definition 4 (EC-GDH Problem): Given a triple (P, aP, bP) for $a, b \in \mathbb{Z}_q^*$, find abP with the help of a EC-DDH oracle (which answers whether a given quadruple is a EC-DDH quadruple or not). If this problem is hard, we say *the EC Gap Diffie-Hellman (EC-GDH) assumption holds in* $\mathbb{G}$.

C. BLS Signature Scheme

Our self-certification technique is based on a threshold version [4] of the BLS signature scheme [6]. BLS signature scheme was proposed by Boneh, et al. [6]. It is a short signature scheme that works in a EC-GDH group $\mathbb{G}$ of order q and a generator P. In short, the scheme operates as follows:

- **Key Generation.** Pick random $x \in \mathbb{Z}_q^*$ and compute $Q = xP$. x is the private key and Q is the corresponding public key.
- **Signing.** To sign a message m, compute $\sigma = xH_1(m)$, where H_1 is a special hash function that maps binary strings onto points in $\mathbb{G}_1$. σ is the signature on m.
- **Verification.** Given (P, Q, m, σ), check if $\hat{e}(Q, H_1(m)) = \hat{e}(P, \sigma)$.

IV. System and Security Model

The basic operations in our self-certification protocol involve only a set of secret share holders. An admission threshold (t) is an important system parameter that needs to be carefully tuned. The self-certification protocol is composed of the following steps:

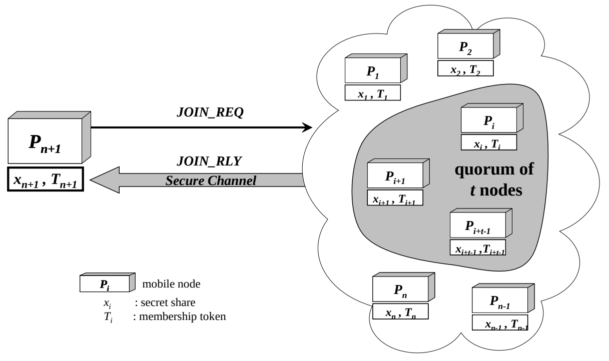

Fig. 1. High-level View of Self-certification Protocol. A prospective node P_{n+1} obtains its secret share x_{n+1} and membership token T_{n+1} from a quorum of t current nodes.

1) *Bootstrapping:* The network is initialized by either a trusted dealer or a set of founding nodes. The dealer or founding nodes initialize the network by choosing a network secret key, and computing and publishing the corresponding public parameters [17]. The network secret is shared among the founding node(s) and the share possessed by each node is referred to as its *secret share*.
2) *Self-Certification:* A prospective node P_{n+1}, who wishes to join the network, must be issued, over a secure channel[4], 1) its secret share for participating in future admission or certification of other nodes and 2) a membership token for authentication and secure communication. Figure 1 gives a high-level overview of the self-certification protocol. Note that, depending on the underlying cryptographic technique, this step may involve multiple rounds and/or co-ordination among the nodes, who sponsor self-certification of P_{n+1}.
3) *Pairwise Key Establishment:* Any pair of nodes, admitted via the self-certification protocol above, need to establish shared keys for secure communication with each other. This functionality can, for example, be applied to achieve secure routing in MANETs.

A secure self-certification protocol must satisfy the following properties:

1) **Completeness.** When the protocol completes (in polynomial time), P_{n+1} has a membership token if at least t out of n group members vote in favor of admission. In addition, P_{n+1} also possesses its own secret share that allows it to take part in future admission decisions
2) **Robustness.** During the self-certification process, a malicious adversary can easily preclude a prospective node from being admitted by inserting incorrect partial secret shares. To prevent this, P_{n+1} must be able to verify the validity of its reconstructed secret share and the membership token before using them. This feature is referred to as *verifiability* in the rest of the paper. In addition, if the verification fails, P_{n+1} must be able to trace the node(s) who sent the fake information. This functionality is provided by the so-called *traceability* feature.
   Note that *verifiability* is always required and thus must be included in self-certification process as a normal operation, whereas *traceability* is only necessary when a member detects (via verifiability) that its reconstructed secret and/or membership token are not valid.
3) **Security.** The self-certification process must not leak any information about either the secret share of any existing node (that takes part in the admission protocol) or the secret x, even to an adversary who has corrupted at most $t - 1$ existing nodes.

V. Non-Interactive Self-Certification

In this section, we describe our new self-certification technique suitable for long-lived MANETs. By coupling the bi-

[4]One way to set up a secure (secret and authenticated) channel between P_{n+1} and each sponsor is with device pairing techniques based on out-of-band (OOB) channels [24], [12], [33], [41]. Alternatively, if P_{n+1} and each sponsor have a common trusted CA, a secure channel can be trivially established using any secure authenticated key agreement protocol, e.g., [42]. Our self-certification protocol allows P_{n+1} to establish secure channels with any node, once it establishes secure channels with any a subset of t nodes. Since all communication between P_{n+1} and sponsors in the self-certification protocol flows over secure channels, “man-in-the-middle” attacks are prevented.

variate polynomial based secret share issuance technique with the non-interactive threshold BLS signature [4], we obtain a fully non-interactive self-certification protocol. We call the protocol *Bivariate polynomial based threshold BLS* or B-BLS in short.

A. Overview

The proposed self-configurable authentication mechanism avoids interaction among sponsors by using a *bi-variate* polynomial $f(z,y)$. To distribute shares among n nodes, a trusted dealer chooses a large prime q and selects a random symmetric bi-variate polynomial $f(z,y) = \sum_{\alpha=0}^{t-1}\sum_{\beta=0}^{t-1} f_{\alpha\beta}z^{\alpha}y^{\beta} \pmod q$ such that $f(0,0) = x$, where the constants $f_{\alpha\beta}$-s are the coefficients of the polynomial and x is the network secret. Since the polynomial is symmetric, $f_{\alpha\beta} = f_{\beta\alpha}$ for each α, β and $f(z,y) = f(y,z)$. For each node P_i, the dealer computes a uni-variate polynomial, called a *share polynomial*, $x_i(z)$ of degree $(t-1)$ such that $x_i(z) = f(z, id_i) \pmod q$, and securely transfers $x_i(z)$ to each node P_i.

In order to admit a new node P_{n+1}, a sponsor must issue to it a *share-polynomial* $x_{n+1}(z)$ in a distributed manner. This can be achieved if at least t nodes provide P_{n+1} with partial shares $x_j(id_{n+1})$ such that $x_j(id_{n+1}) = f(id_{n+1}, id_j)$ for $j \in [1,n]$. P_{n+1} can then use the standard Gaussian elimination procedure [32] to compute $f(id_{n+1}, z)$, which is the same as $f(z, id_{n+1})$ (since the polynomial $f(z,y)$ is symmetric) and thus obtain its share-polynomial $x_{n+1}(z) = f(z, id_{n+1})$ from t partial shares $x_j(id_{n+1})$. Unlike U-BLS, this scheme *does not* require any interaction among the admitting nodes.

In addition to the share polynomial, P_{n+1} also needs to be issued its membership token. This is achieved simply by executing the threshold BLS signing protocol.

B. Bootstrapping

The network can be initialized by either a single node called a trusted dealer, denoted by *TD*, or a set of nodes in distributed way. For the sake of simplicity, we describe only the centralized method in this section. In case of decentralized method, a set of t or more founding nodes agree on a random bi-variate polynomial $f(z,y)$ using the Joint Secret Sharing protocol. For more details, refer to [11].

The set-up first involves the following: elliptic curve parameters $(p, \mathbb{F}_p, a, b, P, q)$ are chosen, the curve being represented by a equation: $y^2 = x^3 + ax + b$. $\mathbb{G}_1$ is set to be a group of order q generated by P, $\mathbb{G}_2$ is a subgroup of $\mathbb{F}^*_{p^2}$ of order q, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is defined to be a public bilinear mapping [5], [9]. Also, $H_1 : \{0,1\}^* \to \mathbb{G}_1$ is the hash function that maps binary strings to non-zero points in $\mathbb{G}_1$. All of this information is published and all network nodes (as well as prospective nodes) are assumed to have access to it.

TD computes a two-dimensional sharing of the secret by choosing a random bi-variate polynomial:

$$f(z,y) = \sum_{\alpha=0}^{t-1}\sum_{\beta=0}^{t-1} f_{\alpha\beta}z^{\alpha}y^{\beta} \pmod q$$

such that $f(0,0) = x$, for the network secret x. *TD* computes $W_{\alpha\beta}$ ($\alpha, \beta \in [0, t-1]$), called *witnesses*: $W_{\alpha\beta} = f_{\alpha\beta}P$ and publishes these $W_{\alpha\beta}$-s.

Next, *TD* computes a *share-polynomial* $x_i(z)$ and a *membership token* T_i for each node P_i ($i \in [1,n]$). The $x_i(z)$ is simply computed with id_i in a way that $x_i(z) = f(z, id_i)$. The procedure to compute T_i is as follows: *TD* generates public and secret key pair for P_i and then computes $T_i = xH_1(id_i, PK_i, etc.)$ where PK_i is a P_i's public key. It then *securely* sends each node a distinct $x_i(z)$, T_i, and a secret key SK_i.[5]

Note that once the network is initialized, *TD* must securely erase the network secret x and all secret coefficients $f_{\alpha\beta}$ of the polynomial. After that, *TD* is no longer needed.

C. Self-Certification

To join the network, P_{n+1} must collect, over secure channels, at least t partial shares of the polynomial and partial membership tokens from the current nodes, respectively. Figure 2 shows the protocol message flow for the self-certification process.

1) A prospective node, P_{n+1}, broadcasts signed JOIN_REQ message m, which contains its public key PK_{n+1} and identity id_{n+1} in order to prove the knowledge of the corresponding private key[6].

2) After verifying the signature on the JOIN_REQ message, each receiving node, (P_i), willing to admit P_{n+1} computes a *partial share* $x_i(id_{n+1})$ using its own *share-polynomial* such that $x_i(id_{n+1}) = f(id_{n+1}, id_i)$. P_i also issues a membership token for P_{n+1} via the threshold BLS signing protocol (refer to Section III-C). It computes the partial membership token $T^{(i)}_{n+1}$ on the request message m such that $T^{(i)}_{n+1} = x_i H_1(m)$ where $x_i = x_i(0)$. (Note that $T^{(i)}_{n+1}$ is computed without Lagrange coefficient $\lambda_i(0)$ which means that the signing does not require any interaction among t sponsoring nodes.)
Each sponsor P_i then replies to P_{n+1} with a JOIN_RLY message. Each message is signed by the sender and contains encrypted $x_i(id_{n+1})$ and partial membership token $T^{(j)}_{n+1}$ along with the respective values of id_i and PK_i. The encryption key $K_{i,n+1}$ is computed using the technique described in Section V-D.
To compute their partial shares, sponsors do not need to be aware of each other, and, thus, no interaction is needed. This is in contrast with U-BLS, where each sponsor needs to be aware of all other sponsors in order to compute the Lagrange coefficient $\lambda_i(id_{n+1})$ in partial share issuance [35].
Note that, in U-BLS, since $\lambda_j(id_{n+1})$-s are publicly known, P_{n+1} can derive P_i's secret share x_i from partial share $x_i\lambda_j(id_{n+1})$. This is prevented using the

[5]Secure channel between the P_i and *TD* can be established using existing techniques. See footnote 4.

[6]We note that it is necessary to include timestamps, nonces and protocol message identifiers in order to secure the protocol against *replay* attacks [25]. However, we omit these values to keep our description simple.

Step 1 (JOIN_REQ). P_{n+1} broadcasts:

$$P_{n+1} \quad \xrightarrow{m=\{id_{n+1},PK_{n+1},etc.\},\ S_{n+1}(m)} \quad \{P_1,\cdots,P_n\}$$

Step 2 (JOIN_RLY). Each P_i $(i \in [1,t'])$ where $t \leq t' \leq n$:
- computes its partial secret share $x_i(id_{n+1}) = f(id_{n+1}, id_i)$,
- computes partial membership token $T_{n+1}^{(i)} = x_i H_1(m)$
- encrypts $x_i(id_{n+1})$ and $T_{n+1}^{(i)}$.

$$P_{n+1} \quad \xleftarrow{m'=\{id_i,PK_i,E_{K_{i,n+1}}\{x_i(id_{n+1})\},T_{n+1}^{(i)}\},\ S_i(m')} \quad \{P_1,\cdots,P_{t'}\}$$

Step 3. P_{n+1}:
- selects any t out of t' partial shares and constructs its share-polynomial $x_{n+1}(z)$ using Gaussian elimination,
- computes the membership token $T_{n+1} = \sum_{j=1}^{t} T_{n+1}^{(j)} \lambda_j(0)$.

Fig. 2. The message flow of B-BLS self-certification and distributed share issuance protocol. P_{n+1} must collect at least t partial shares of the polynomial and partial membership tokens from the current nodes, respectively.

random shuffling technique proposed in [20] by adding extra random value r_{ij} to each share. These r_{ij}-s are securely shared between sponsors P_i and P_j and sum up to zero by construction.

We note that, due to the random shuffling procedure [20], [19], [23], U-BLS protocol becomes heavily interactive among the t sponsoring nodes – it requires $O(t^2)$ point-to-point messages as well as extremely expensive $O(t)$ reliable broadcast messages [7]. All this makes it impractical for most MANET settings.

3) Upon receiving t' $(\geq t)$ JOIN_RLY messages, P_{n+1} selects *any* t of them and computes its own share-polynomial $x_{n+1}(z)$ and membership token T_{n+1}. First, the share-polynomial is constructed using standard Gaussian elimination [32]. Let us denote the share-polynomial $x_{n+1}(z)$ reconstructed by P_{n+1} as $\sum_{\alpha=0}^{t-1} A_\alpha z^\alpha$. Since $x_i(id_{n+1}) = x_{n+1}(id_i)$ due to the symmetry, the selected t partial shares $\{x_{n+1}(id_1), \cdots, x_{n+1}(id_t)\}$ can be represented as

$$\begin{aligned} A_0 + A_1 id_1 + A_2 {id_1}^2 + \cdots + A_{t-1} {id_1}^{t-1} &= x_{n+1}(id_1) \\ A_0 + A_1 id_2 + A_2 {id_2}^2 + \cdots + A_{t-1} {id_2}^{t-1} &= x_{n+1}(id_2) \\ &\vdots \\ A_0 + A_1 id_t + A_2 {id_t}^2 + \cdots + A_{t-1} {id_t}^{t-1} &= x_{n+1}(id_t). \end{aligned}$$

Thus, the problem of interpolating $x_{n+1}(z)$ using t $x_i(id_{n+1})$-s is equivalent to the problem of computing the matrix A such that $XA = B$:

$$\begin{bmatrix} (id_1)^0 & (id_1)^1 & \cdots & (id_1)^{t-1} \\ (id_2)^0 & (id_2)^1 & \cdots & (id_2)^{t-1} \\ & & \vdots & \\ (id_t)^0 & (id_t)^1 & \cdots & (id_t)^{t-1} \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \\ \vdots \\ A_{t-1} \end{bmatrix} = \begin{bmatrix} x_{n+1}(id_1) \\ x_{n+1}(id_2) \\ \vdots \\ x_{n+1}(id_t) \end{bmatrix}$$

The above system of linear equations yields a unique solution since the id_i values are distinct and the matrix $X = [x_{ij}]$, where $x_{ij} = (id_i)^{j-1}$ for all $i, j \in [0, t]$, is invertible.

In order to validate the acquired share-polynomial $x_{n+1}(z)$, P_{n+1} must perform the verifiability procedure: $A_\alpha = \sum_{\beta=0}^{t-1} f_{\alpha\beta}(id_{n+1})^\beta$ for $\alpha \in [0, t-1]$. Using the public witness values $W_{\alpha\beta} = f_{\alpha\beta}P$, the polynomial can be verified: $A_\alpha P = \sum_{\beta=0}^{t-1} (id_{n+1})^\beta W_{\alpha\beta}$ for $\alpha \in [0, t-1]$. The right-hand side in this equation can be pre-computed by P_{n+1} prior to starting the process.

Next, P_{n+1} also computes the threshold signature to construct its own membership token by simply multiplying the appropriate Lagrange coefficient with each partial signature and simply adding them, i.e., $T_{n+1} = \sum_{j=1}^{t} T_{n+1}^{(j)} \lambda_j(0) = \sum_{j=1}^{t} (x_j \lambda_j(0)) H_1(m) = x H_1(m)$. Similar to the share verification described above, P_{n+1} also verifies the acquired signature. The membership token T_{n+1} is verified by checking $\hat{e}(P, T_{n+1}) = \hat{e}(Q, H_1(m))$ where $Q = xP$.

D. Pairwise Key Establishment

Once every node has its share-polynomial, pairwise key establishment is the same as in [3] and [21]. Any pair of nodes P_i and P_j can establish shared keys as follows: P_i uses its share-polynomial $f(z, id_i)$ to compute K_{ij} such that $K_{ij} = f(id_j, id_i)$. Similarly, P_j uses its share-polynomial $f(z, id_j)$ to compute K_{ji} such that $K_{ji} = f(id_i, id_j)$. Since $f(z, y)$ is a symmetric polynomial, $K_{ij} = K_{ji}$. Thus, P_i and P_j now have a shared key that can be used for secure communication.

The security of above procedure is unconditional, i.e., not based on any computational assumption. Refer to [3] for details regarding the security arguments of this pairwise key establishment.

E. Security Considerations

In this section, we argue the security of the proposed scheme, based on the security model described in Section IV.

1) **Completeness.** This property follows by inspection. At the end of the protocol, P_{n+1} receives the membership token T_{n+1} which is verified as: $\hat{e}(P, T_{n+1}) = \hat{e}(Q, H_1(m))$. Using T_{n+1}, P_{n+1} can prove membership. P_{n+1} also receives a share-polynomial $x_{n+1}(z)$ which is verified as: $A_\alpha P = \sum_{\beta=0}^{t-1} (id_{n+1})^\beta W_{\alpha\beta}$ where A_α is a coefficient of $x_{n+1}(z)$ and $\alpha \in [0, t-1]$. Using $x_{n+1}(z)$, P_{n+1} can take part in future admission decisions and can also recover the group secret x in collaboration with any other $t-1$ members. Of course, P_{n+1} can obtain these credentials in polynomial time.
2) **Robustness.** P_{n+1} is able to identify (trace) malicious group members (if there are any) and does so as follows.
   a) **Partial Token Trace.** In case the verification of T_{n+1} fails, P_{n+1} can trace sponsors that sent invalid partial token(s). The correctness of each partial token T_{n+1}^i can be verified as $\hat{e}(P, T_{n+1}^{(i)}) = \hat{e}(\sum_{\beta=0}^{t-1}(id_i)^\beta W_{0\beta}, H_1(m))$.
   b) **Partial Share Trace.** If the verification of share polynomials fails, P_{n+1} must trace the faulty share providers by performing the traceability procedure. This involves verifying the validity of each partial share $x_i(id_{n+1}) = f(id_{n+1}, id_i)$ that P_{n+1} received. This can be achieved by checking: $x_i(id_{n+1})P = \sum_{\alpha=0}^{t-1} \sum_{\beta=0}^{t-1} (id_{n+1})^\alpha (id_i)^\beta W_{\alpha\beta}$. Note that $\sum_{\alpha=0}^{t-1} (id_{n+1})^\alpha W_{\alpha\beta}$ in the equation can be pre-computed since $W_{\alpha\beta}$-s and id_{n+1} are known to P_{n+1} in advance.

   If either of the above tracing functions fail, P_{n+1} concludes that P_i is cheating.
3) **Security.** The security of the secret-share polynomial acquisition part of our protocol is based on the computational hardness of the EC-DL assumption, as long as the adversary can not corrupt more than $(t-1)$ nodes. We briefly sketch out this argument. Basically, we show that an adversary, who corrupts at most $(t-1)$ nodes, learns nothing (other than the witness $W_{00} = xP$) about the secret x, during the initialization and admission procedures of the scheme. This is achieved by generating a simulator which, on input xP, produces public information and private information to the adversary which is statistically indistinguishable from the one produced in the actual execution of these procedures. Intuitively, only on receiving at least t partial shares $x_i(id_{n+1}) = f(id_{n+1}, id_i)$, the new node can compute its own share polynomial $x_{n+1}(z) = f(z, id_{n+1})$. From this acquired polynomial, the new node can not learn anything regarding the system secret x, the secret polynomials of any other nodes or the pairwise keys shared between any pair of nodes id_I and id_J such that $I \neq n+1$ and $J \neq n+1$. The security of the membership token acquisition part of our protocol simply reduces to the security of the threshold BLS signature scheme, which in turn is based on the computational hardness of the EC-GDH problem [4].

VI. Performance Analysis

In this section, we discuss the implementation of U-BLS and B-BLS and compare them in terms of self-certification, traceability and pair-wise key establishment costs. We also summarize and compare some salient features in Table II. As expected, B-BLS significantly outperforms U-BLS in our overall evaluation.

TABLE II
FEATURE COMPARISON

Key Features	U-BLS	B-BLS
Security Assumption (for self-certification)	EC-DL	EC-DL
Security Assumption (for key establishment)	EC-CDH	Unconditional
DoS Resistance (traceability)	Yes	Yes
Interaction among Sponsors Required	Yes	No
Random Shuffling Required	Yes	No
Reliable Broadcast Required	Yes	No

A. Complexity Analysis and Comparison

We summarize computation and communication complexities[7] in Table III.

TABLE III
COST COMPARISON

Category			U-BLS	B-BLS
Computation	Self-Certification	$\mathcal{M}$	t^2+1	$3t$
		$\mathcal{P}$	2	2
	Traceability	$\mathcal{M}$	t^3+t^2+2t	$2t^2$
		$\mathcal{P}$	$2t$	$2t$
	Key Establishment	$\mathcal{M}$	t	0
		$\mathcal{P}$	0	0
Communi-cation	Round	broadcast	1	1
		unicast	t^2+2t	t
	Bandwidth	log q-bit	$2t^2+2t$	$3t$
		log p-bit	$3t$	$3t$

$\mathcal{M}$: scalar-point-multiplication in ECC, $\mathcal{P}$: Tate pairing operation in ECC

More specifically, for self-certification, B-BLS requires each sponsoring node P_i to perform $O(t)$ scalar-point-multiplication ($\mathcal{M}$) operations over Elliptic Curves (ECC) and the joining node P_{n+1} to perform only two Tate pairing ($\mathcal{P}$) operations in ECC. On the other hand, U-BLS requires each P_i to perform $O(t^2)$ $\mathcal{M}$ operations, and P_{n+1} to perform two $\mathcal{P}$ operations. For traceability, U-BLS requires $O(t^3)$ $\mathcal{M}$-s and $O(t)$ $\mathcal{P}$-s with pre-computation, whereas B-BLS does $O(t^2)$ $\mathcal{M}$-s and $O(t)$ $\mathcal{P}$-s with pre-computation. B-BLS is significantly more efficient than U-BLS for computing pairwise keys, since the former requires only $O(t)$ 160-bit modular multiplications, while the latter needs $O(t)$ $\mathcal{M}$ ECC operations. Note that, pairwise key establishment is a very frequent operation in a MANET, thus, its efficiency is extremely important. As far as overall communication costs are concerned, B-BLS consumes $O(t \log q)$ and $O(t \log p)$ bits, while bandwidth consumption in U-BLS is $O(t^2 \log q)$ plus $O(t \log p)$ bits, due to the interactive random shuffling procedure.

[7]The costs required for protecting each protocol message are not taken into account since these costs vary with the specific signature scheme.

B. Basic Operations

To estimate the performance of B-BLS, we first present the costs of the primitive operations in Table IV. For measuring the costs of basic operations in B-BLS, we used a machine with an Intel P4 3.0GHz processor and 512MB memory.

TABLE IV
COSTS OF PRIMITIVE OPERATIONS (P4-3.0GHZ, 512MB RAM)

Function	modulus (bits)	exponent (bits)	average time (msec)
Map-to-point ($H_1(\cdot)$)	512	160	2.31
scalar-point multiplication	512	160	6.86
Tate pairing	512	160	20.74
BLS sign	512	160	9.71
BLS verify	512	160	36.92

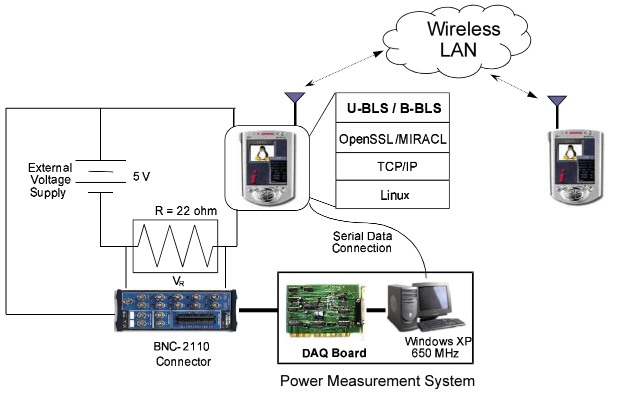

Fig. 3. Power Measurement Testbed. The test machine was an iPAQ running Linux on a 400 MHz Intel XScale with 48MB of flash memory and 128MB of SDRAM. The National Instruments PCI DAQ board was used to sample the voltage drops.

C. Experimental Setups

U-BLS and B-BLS protocols have been implemented over the popular OpenSSL library [30] and MIRACL library (optimized using Comba method) [26]. Currently, our implementation consists of approximately $20,000$ lines of C source code and supports Linux 2.4 and 2.6.

We now describe the experimental testbeds for measuring the performance of our proposed protocol. We ran experiments in a *real* wireless MANET environment and also measured energy consumption costs for each scheme with a power measurement system as described below.

Wireless Mobile Ad Hoc Networks. We used ten laptop computers with Intel Core-2 Duo 2.0 GHz CPU and 2 GB memory for our wireless experimental set-up. Each machine is configured with $802.11g$ in ad-hoc mode and runs the Optimized Link State Routing Protocol (OLSR) [29]. Each machine runs Linux kernel 2.6.

Power Measurement Systems. To measure consumption of battery power, we configured the following equipment, as shown in Figure 3. The test machine was an iPAQ (model H5555) running Linux (Familiar-0.7.2). The CPU on the iPAQ is a 400 MHz Intel XScale with 48MB of flash memory and 128MB of SDRAM. In order to obtain accurate power measurements, we removed the battery from the iPAQ during the experiment and placed a resistor in series with power supply. We used a National Instruments PCI DAQ (Data AcQuisition) board to sample the voltage drops across the resistor to calculate the current at 1000 samples per second.

D. Test Methodology

Parameter Selection. To perform fair comparisons, we consider the following parameters. The size of the parameter q was set to be 160-bit and p to be 1024-bit. For more details, we used the elliptic curve E defined by the equation: $y^2 = x^3 + 1$ over $\mathbb{F}_p$ with $p > 3$ a prime satisfying $p = 2\ (mod)\ 3$ and q being a prime factor[8] of $p + 1$. The parameter p is a 512-bit prime in order to make sure that the security of pairing $\hat{e}$ is equivalent to the security as in finite field of 1024 bits [9]. The measurements were performed with different threshold values t from 1 to 9. We used 1024-bit RSA signature algorithm with the fixed public exponent $65537(= 2^{16} + 1)$ for protocol message authentication. All experiments were repeated $1,000$ times for each measurement in order to get fairly accurate averaged results.

Test Cases. We measured the respective costs of self-certification, traceability, pairwise key establishment, and energy consumption.

1) **Self-Certification.** To measure the self-certification cost, nine laptops with same computing power were used as existing MANET nodes and one laptop was used as the joining/new node throughout the experiments. We then measured total processing time between sending of JOIN_REQ by the prospective node and receiving (plus verification) of acquired secret share and membership token[10]. The measurement results thus include the average computation time of the basic operations as well as communication costs, such as packet en/decoding time, network delay, etc.
2) **Traceability.** We measured the computation time for tracing partial shares and partial membership tokens that are received during the self-configuration protocol. We measured this cost with optimization using pre-computed values as much as possible, wherever applicable.
3) **Pairwise Key Establishment.** We measured the pro-

[8]By Euler's theorem, q must divide $\#E(\mathbb{F}_p)$. For the curve $y^2 = x^3 + 1$, $\#E(\mathbb{F}_p) = p + 1$.

[9]The $\mathbb{G}_1$ is a subgroup of points generated by P such that $P \in E(\mathbb{F}_p)$. The $\mathbb{G}_2$ is a subgroup of $\mathbb{F}^*_{p^2}$ of order q. The bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is the well-known Tate pairing. Computing discrete log in $\mathbb{F}_{p^2}$ is sufficient for computing discrete log in $\mathbb{G}_1$. Therefore, for proper security of discrete log in $\mathbb{F}_{p^2}$ the prime p should be at least 512-bits long (so that the group size is at least 1024-bits long)

[10]In our protocol, even if some faulty nodes are involved, this does not affect the performance results because the new node only gets the response from at least t non-faulty sponsoring nodes. This is based on the principle of threshold cryptography that there can be at most $t - 1$ faulty or corrupted nodes. Hence, our self-certification experiment based on admission threshold is equivalent to the experiment in presence of faulty nodes.

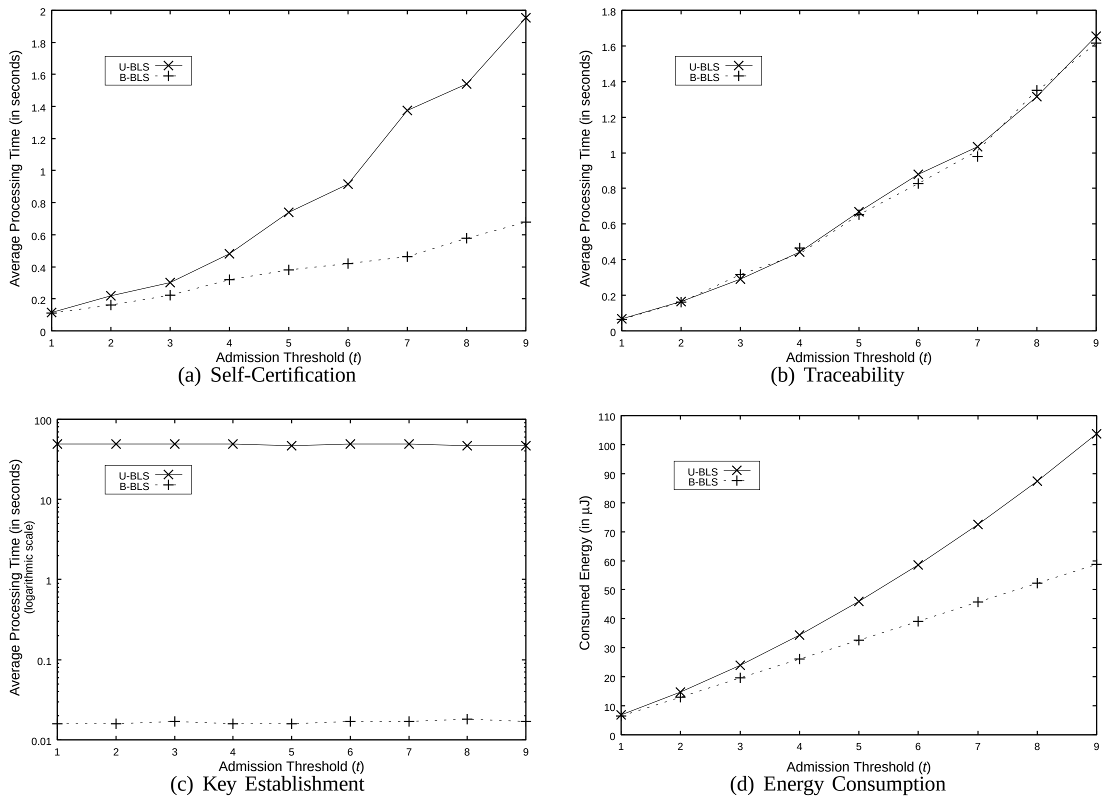

Fig. 4. Experimental Results. B-BLS performs much better than U-BLS in self-certification, pairwise key establishment, and energy consumption experiments and shows similar performance in traceability experiment.

cessing time for a node to compute a pairwise key on the high-end laptop. Note that no communication is involved in this measurement.

4) **Energy Consumption.** This experiment is quite tricky to perform fairly. It is meaningless to measure energy consumption with all the test cases above. However, it is well known that, in many small devices such as low-end MANET nodes or sensors, sending a single bit is roughly equivalent to performing 1,000 32-bit computations in terms of batter power consumption [1]. Therefore, we measured power consumption in terms of communication bandwidth required by each self-configuration protocol. For more details, we sent some bulk data (e.g., 100 MB) from a single iPAQ PDA (refer to Figure 3), measured power consumed while sending out this data, and then computed the average power consumption per bit. After that, we calculated power consumption of each protocol by multiplying this measurement result by the bit length of the transmitted data.

E. Experimental Results

We compare our experiment results in terms of self-certification, traceability, pairwise key computation, and energy consumption.

Self-Certification Results. As observed from Figure 4(a), the self-certification cost with B-BLS is much lower than that with U-BLS. The difference is even higher for higher threshold values. The reason is quite intuitive: not only is B-BLS computationally cheaper than U-BLS, but it also requires less communication.

Traceability Results. Figure 4(b) displays traceability costs for the two protocols. Even in the worst case, B-BLS is as good as U-BLS for performing the (very infrequent) operation of tracing malicious nodes.

Pairwise Key Establishment Results. Figure 4(c) shows that B-BLS is significantly more efficient than U-BLS for computing pairwise keys. This result was as expected because in B-BLS the pairwise key computation requires only $O(t)$ multiplications where the modular size is 160 bits. In contrast, U-BLS requires $O(t)$ exponentiations with a modular size of 1024 bits as well as $O(t)$ multiplications with 160-bit modulus.

Energy Consumption Results. Energy consumption results for self-certification operation are plotted in Figure 4(d). These results in Figure 4(d) clearly illustrate that B-BLS is much more energy-efficient than U-BLS.

VII. Conclusion

In this paper, we proposed B-BLS, a fully non-interactive self-certification protocol for long-lived MANETs via a novel combination of bi-variate polynomial secret sharing and threshold BLS signature scheme. We demonstrated, using theoretical and experimental evaluation, that B-BLS is more efficient than previous mechanisms, based on uni-variate polynomial secret sharing and threshold BLS signature, in terms of computation, communication, and energy consumption.

ACKNOWLEDGMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (Grant No. 2009-0067977). Portions of this paper appeared in [36], [43]. The authors would also like to thank Gene Tsudik for his collaboration on the conference version [36] related to this paper. Jeong Hyun Yi is the corresponding author.

REFERENCES

[1] K. Barr and K. Asanovic. Energy Aware Lossless Data Compression. In *ACM MobiSys'03*, pages 231–244, May 2003.

[2] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In *ACM STOC'88*, pages 1–10, May 1988.

[3] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-Secure Key Distribution for Dynamic Conferences. In *CRYPTO'92*, pages 471–48, Aug. 1999.

[4] A. Boldyreva. Efficient Threshold Signatures, Multisignatures and Blind Signatures based on the Gap-Diffie-Hellman-Group Signature Scheme. In *PKC'03*, pages 31–46, 2003.

[5] D. Boneh and M. Franklin. Identity-based Encryption from the Weil Pairing. In *CRYPTO'01*, pages 213–229, Aug. 2001.

[6] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. In *ASIACRYPT'01*, pages 514–532, Dec. 2001.

[7] G. Bracha. An Asynchronous $\lfloor (n-1)/3 \rfloor$-resilient Consensus Protocol. In *ACM PODC'84*, pages 154–162, Aug. 1984.

[8] Y. Desmedt and Y. Frankel. Threshold Cryptosystems. In *CRYPTO'89*, pages 307–315, Aug. 1990.

[9] G. Frey, M. Müller, and H.-G. Rück. The Tate Pairing and the Discrete Logarithm Applied to Elliptic Curve Cryptosystems. In *IEEE Transactions on Information Theory*, volume 45, pages 1717–1719, July 1999.

[10] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust Threshold DSS Signatures. In *CRYPTO'96*, pages 354–371, May 1996.

[11] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. In *EUROCRYPT'99*, pages 295–310, May 1999.

[12] M. T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun. Loud and Clear: Human-Verifiable Authentication Based on Audio. In *IEEE ICDCS'06*, 2006.

[13] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive Secret Sharing, Or How To Cope With Perpetual Leakage. In *CRYPTO'95*, pages 339–352, Aug. 1995.

[14] Y.-C. Hu, D. B. Johnson, and A. Perrig. SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. In *IEEE HotMobile'02*, pages 3–13, June 2002.

[15] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. In *ACM MobiCom'02*, pages 12–23, Sept. 2002.

[16] S. Jarecki, N. Saxena, and J. H. Yi. An Attack on the Proactive RSA Signature Scheme in the URSA Ad Hoc Network Access Control Protocol. In *ACM SASN'04*, pages 1–9, Oct. 2004.

[17] Y. Kim, D. Mazzocchi, and G. Tsudik. Admission Control in Peer Groups. In *IEEE NCA'03*, pages 131–139, Apr. 2003.

[18] N. I. Koblitz. *A Course in Number Theory and Cryptography*. Springer-Verlag, 1995. ISBN 0-387-94293-9.

[19] J. Kong, H. Luo, K. Xu, D. L. Gu, M. Gerla, and S. Lu. Adaptive Security for Multi-level Ad-hoc Networks. In *Wiley Journal of Wireless Communications and Mobile Computing*, volume 2, pages 533–547, Aug. 2002.

[20] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing Robust and Ubiquitous Security Support for MANET. In *IEEE ICNP'01*, pages 251–260, Nov. 2001.

[21] D. Liu and P. Ning. Establishing Pairwise Keys in Distributed Sensor Networks. In *ACM CCS'03*, pages 52–61, Oct. 2003.

[22] H. Luo, J. Kong, P. Zerfos, S. Lu, and L. Zhang. URSA: Ubiquitous and Robust Access Control for Mobile Ad Hoc Networks. In *IEEE/ACM Transactions on Networking*, volume 12, pages 1049–1063, Dec. 2004.

[23] H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang. Self-securing Ad Hoc Wireless Networks. In *IEEE ISCC'02*, pages 567–574, July 2002.

[24] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *IEEE SP'05*, 2005.

[25] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997. ISBN 0-8493-8523-7.

[26] MIRACL Library, http://indigo.ie/ mscott.

[27] M. Naor, B. Pinkas, and O. Reingold. Distibuted Pseudo-Random Functions and KDCs. In *EUROCRYPT'99*, pages 327–346, May 1999.

[28] M. Narasimha, G. Tsudik, and J. H. Yi. On the Utility of Distributed Cryptography in P2P and MANETs: The Case of Membership Control. In *IEEE ICNP'03*, pages 336–345, Nov. 2003.

[29] OLSR Protocol, http://menetou.inria.fr/olsr.

[30] OpenSSL Project, http://www.openssl.org.

[31] R. Ostrovsky and M. Yung. How to Withstand Mobile Virus Attacks. In *ACM PODC'91*, pages 51–61, Aug. 1991.

[32] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, 1992. ISBN 0-521-43108-5.

[33] N. Saxena, J.-E. Ekberg, K. Kostiainen, and N. Asokan. Secure device pairing based on a visual channel (short paper). In *IEEE SP'06*, May 2006.

[34] N. Saxena, G. Tsudik, and J. H. Yi. Admission Control in Peer-to-Peer: Design and Performance Evaluation. In *ACM SASN'03*, pages 104–114, October 2003.

[35] N. Saxena, G. Tsudik, and J. H. Yi. Identity-based Access Control for Ad-Hoc Groups. In *ICISC'04*, December 2004.

[36] N. Saxena, G. Tsudik, and J. H. Yi. Efficient Node Admission for Short-lived Mobile Ad Hoc Networks. In *IEEE ICNP'05*, pages 269–278, Nov. 2005.

[37] A. Shamir. How to Share a Secret. In *Communications of the ACM*, volume 22, pages 612–613, Nov. 1979.

[38] M. Steiner, G. Tsudik, and M. Waidner. CLIQUES: A New Approach to Group Key Agreement. In *IEEE ICDCS'98*, pages 380–387, May 1998.

[39] M. Steiner, G. Tsudik, and M. Waidner. Key Agreement in Dynamic Peer Groups. In *IEEE Transactions on Parallel and Distributed Systems*, volume 11, pages 769–780, July 2000.

[40] D. R. Stinson and R. Strobl. Provably Secure Distributed Schnorr Signatures and a (t, n) Threshold Scheme for Implicit Certificates. In *ACISP'01*, pages 417–434, July 2001.

[41] E. Uzun, K. Karvonen, and N. Asokan. Usability analysis of secure pairing methods. In *USEC'07*, 2007.

[42] P. van Oorschot. Extending Cryptographic Logics of Belief to Key Agreement Protocols. In *ACM CCS'93*, pages 232–243, 1993.

[43] J. H. Yi. Energy-Efficient and Non-interactive Self-certification in MANETs. In *SSS'06*, pages 533–547, 2006.

[44] L. Zhou and Z. J. Haas. Securing Ad Hoc Networks. In *IEEE Network Magazine*, volume 13, pages 24–30, Nov. 1999.

Nitesh Saxena is an Assistant Professor in the Department of Computer Science and Engineering at Polytechnic Institute of New York University (formerly Polytechnic University). He works in the areas of computer and network security, and applied cryptography. Nitesh obtained his Ph.D in Information and Computer Science from UC Irvine. He holds an M.S. in Computer Science from UC Santa Barbara, and a Bachelor's degree in Mathematics and Computing from the Indian Institute of Technology, Kharagpur, India. Nitesh's Ph.D. dissertation on "Decentralized Security Services" has been nominated for the ACM Dissertation Award 2006. He is the recipient of the Best Student Paper Award at Applied Cryptography and Network Security (ACNS) conference 2006.

Jeong Hyun Yi is an Assistant Professor in the School of Computing at Soongsil University, Seoul, Korea. He obtained a Ph.D. in Information and Computer Science from the University of California, Irvine in 2005. He received an M.S. and a B.S. in Computer Science at Soongsil University, Korea in 1995 and 1993, respectively. He was a Principal Researcher at Samsung Advanced Institute of Technology, Korea, from 2005 to 2008 and a member of research staff at Electronics and Telecommunication Research Institute (ETRI), Korea, from 1995 to 2001. Between 2000 and 2001, he was a guest researcher at National Institute of Standards and Technology (NIST), USA. His research interests include: network security, mobile security and privacy, ubiquitous computing, RFID, PKI, and applied cryptography. Some of his notable research contributions include Certificate Management Protocol (CMP) for Korean PKI Standards and integration of Korea PKI and US Federal PKI.